

<https://doi.org/10.26160/2572-4347-2022-17-13-27>

## МНОГОМЕТОДНАЯ ОПТИМИЗАЦИЯ УПРАВЛЕНИЯ В ЛИНЕЙНЫХ СИСТЕМАХ

*Тятюшкин А.И.*

*Институт динамики систем и теории управления СО РАН, Иркутск, Россия*

**Ключевые слова:** численные методы, метод выпуклых оболочек, оптимальное управление, задача быстродействия, адаптивные алгоритмы, линейное программирование, многометодная оптимизация.

**Аннотация.** Рассматриваются численные методы решения задач оптимального управления в линейных системах: задачи терминального управления с ограничениями на управление и задача оптимального быстродействия. Наряду с прямыми итерационными методами поиска оптимального управления предлагаются двойственные алгоритмы, состоящие в оптимизации начальных условий сопряженной системы. Предлагается многометодная технология, включающая одновременно несколько методов оптимизации и ориентированная на поиск оптимального управления в линейных системах с некоторыми особенностями, например, когда множество достижимости системы имеет плоские грани.

## MULTI-METHOD OPTIMIZATION OF CONTROL IN LINEAR SYSTEMS

*Tyatyushkin A.I.*

*Institute of System Dynamics and Control Theory SB RAS, Irkutsk, Russia*

**Keywords:** numerical methods, convex hull method, optimal control, speed problem, phase constraints, adaptive algorithms, linear programming, multi-method optimization.

**Abstract.** Numerical methods for solving optimal control problems in linear systems are considered: terminal control problems with constraints on the control and the optimal speed problem. Along with direct iterative methods for finding the optimal control, dual algorithms are proposed that consist in optimizing the initial conditions of the adjoint system. A multi-method technology is proposed that simultaneously includes several optimization methods and is focused on finding the optimal control in linear systems with some features, for example, when the system reachability set has flat faces.

### Введение

Задачи терминального управления и задачи быстродействия составляют важный класс прикладных задач оптимального управления. К ним относятся, например, следующие задачи из динамики полета: приведение движущихся объектов в заданное положение, ориентация и стабилизация летательных аппаратов, подъем ракеты на максимальную высоту и другие задачи управления движением. Наряду с нелинейной моделью объекта управления на стадии проектирования или на малом временном отрезке нередко используется линейная модель. При этом многие задачи терминального управления формируются в виде задачи минимизации квадратичной функции, которая служит мерой отклонения конечного состояния от некоторого заданного состояния. Кроме того, последовательность таких задач возникает и при решении задачи быстродействия прямыми методами. Поэтому построению эффективных вычислительных алгоритмов для решения этой задачи уделяется большое внимание.

Для решения задачи минимизации выпуклой функции конечного состояния конструируются прямые и двойственные алгоритмы градиентного типа. При этом поиск ведется по точкам множества достижимости, которые генерируются с помощью решений сопряженной системы, полученных для начальных условий – столбцов единичной матрицы до процесса оптимизации. Такой способ существенно снижает трудоемкость итераций градиентного метода.

Для решения важной в инженерной практике задачи быстродействия описывается алгоритмическое обеспечение, основанное на математических методах [1-5] и позволяющее реализовать многометодную технологию [6, 7] поиска оптимального управления.

### 1. Решение задач терминального управления

Будем исследовать задачи оптимального управления в линейных системах

$$\dot{x} = A(t)x + B(t)u(t) + f(t), \quad x(t_0) = x^0, \quad (1.1)$$

где  $x(t) \in R^n$ ,  $u(t) \in R^r$ ,  $A(t)$ ,  $B(t)$ ,  $f(t)$  –  $n \times n$ ,  $n \times r$ ,  $n \times 1$ -матрицы с кусочно-непрерывными элементами.

В теории линейных управляемых систем значительную роль играет переходная матрица [1]  $\Phi(t, \tau)$ , называемая также матрицей Грина. Она удовлетворяет следующему матричному дифференциальному уравнению:

$$\frac{d}{d\tau} \Phi(t, \tau) = -\Phi(t, \tau) A(\tau), \quad \Phi(t, t) = E. \quad (1.2)$$

С помощью этой матрицы решение (1.1) записывается по формуле Коши

$$x(t) = \Phi(t, t_0) x^0 + \int_{t_0}^t \Phi(t, \tau) [B(\tau)u(\tau) + f(\tau)] d\tau.$$

Введя обозначение  $c = \Phi(t_1, t_0) x^0 + \int_{t_0}^{t_1} \Phi(t_1, \tau) f(\tau) d\tau$ , вектор состояния в

момент  $t = t_1$  представим в виде

$$x(t_1) = c + \int_{t_0}^{t_1} \Phi(t_1, \tau) B(\tau) u(\tau) d\tau.$$

Заметим, что вектор  $c$  является состоянием системы (1.1) в момент  $t = t_1$ , в которое она переводится с помощью нулевого управления ( $u(t) \equiv 0$ ).

Множество конечных состояний  $x(t_1)$ , соответствующее заданному классу допустимых управлений, называется множеством достижимости системы (1.1).

Пусть задан некоторый вектор  $x^T \in R^n$ . Поставим вопрос о существовании управления, переводящего систему (1.1) в заданное состояние  $x^T$ . Ответ на данный вопрос дает следующий критерий управляемости

Калмана [2]: для системы (1.1) тогда и только тогда существует управление  $u(t)$ , которое переводит систему из состояния  $x^0$  при  $t = t_0$  в состояние  $x^T$  при  $t_1 \geq t_0$ , когда вектор  $x^T - c$  принадлежит области значений линейного преобразования

$$W = \int_{t_0}^{t_1} \Phi(t_1, \tau) B(\tau) B'(\tau) \Phi'(t_1, \tau) d\tau.$$

Для вычисления матрицы  $W$  по квадратурным формулам необходимо интегрировать в обратном времени уравнение (1.2). Вместо расчета по квадратурным формулам можно совместно с (1.2) интегрировать дополнительное матричное уравнение

$$\frac{d}{d\tau} V(t_1, \tau) = \Phi(t_1, \tau) B(\tau) B'(\tau) \Phi'(t_1, \tau), \quad V(t_1, t_1) = 0.$$

Тогда при  $t = t_0$  будем иметь

$$W = -V(t_1, t_0).$$

Менее трудоемкий способ вычисления матрицы  $W$  приведен в [1] и заключается в решении следующего матричного уравнения:

$$\frac{d}{d\tau} V(\tau) = A(\tau) V(\tau) + V(\tau) A(\tau)' + B(\tau) B(\tau)', \quad V(t_0) = 0, \quad (1.3)$$

интегрирование которого от  $t_0$  до  $t_1$  дает матрицу  $W = V(t_1)$ .

Важнейшим свойством линейных систем, которое позволяет добиться существенного ускорения численных методов оптимизации, является выпуклость множества достижимости. С использованием этого свойства многие алгоритмы дают возможность найти новое приближение для управления с помощью решения простой задачи выпуклого программирования. В отличие от нелинейных систем алгоритмы решения данной задачи не требуют дополнительных интегрирований системы (1.1).

Другое преимущество линейных систем состоит в том, что применение формулы Коши позволяет исключить переменные состояния и установить явную зависимость функционала от управления. В некоторых случаях это дает возможность получить решение задачи оптимального управления в явной форме. Например, управление с минимальной энергией [3] представляется с помощью решения уравнения (1.2) и вектора множителей Лагранжа, который является решением линейной системы алгебраических уравнений.

Рассмотрим алгоритмы, на основе которых строится многометодная технология, применяемая программными системами [6] для численного решения задач терминального управления.

### 1.1. Минимизация выпуклой функции конечного состояния

Пусть на правых концах траекторий системы (1.1) определена непрерывно дифференцируемая выпуклая скалярная функция  $\varphi(x(t_1))$  и

поставлена задача

$$\varphi(x(t_1)) \rightarrow \min. \quad (1.4)$$

При отсутствии ограничений на управление и фазовые координаты ее можно решить с помощью простого алгоритма: сначала найдем безусловный (без учета связей (1.1)) минимум функции  $\varphi(x)$ , т.е. точку  $x^T = \arg \min_{x \in R^n} \varphi(x)$ , а затем вычислим управление, с минимальной энергией переводящее систему (1.1) из  $x^0$  в  $x^T$ , или установим отсутствие такого управления в случае неуправляемости системы [5].

Более распространенный подход к решению задачи (1.1), (1.4) состоит в применении градиентных методов

$$u^{k+1}(t) = u^k(t) + \alpha_k B'(t) \psi^k(t), \quad k = 0, 1, \dots,$$

где  $\psi(t)$  – решение задачи Коши

$$\dot{\psi}(t) = -A'(t)\psi(t), \quad \psi(t_1) = -\nabla \varphi(x(t_1)), \quad (1.5)$$

$\alpha_k$  находится, например, из условия наискорейшего спуска.

Опишем вычислительную технологию решения задачи (1.1), (1.4) с ограничениями на управление

$$\alpha_j(t) \leq u_j(t) \leq \beta_j(t), \quad j = \overline{1, r}, \quad t \in [t_0, t_1], \quad (1.6)$$

методом условного градиента. В отличие от традиционной схемы метода, в которой точки множества достижимости получаются интегрированием системы (1.1), построим алгоритм, в котором эти точки генерируются с помощью базисных решений сопряженной системы.

Решением линейной вспомогательной задачи  $\nabla \varphi(x^k(t_1))' x \rightarrow \min$ , определяемой на каждом шаге метода, будет управление

$$\bar{u}_j^k(t) = \begin{cases} \alpha_j(t), & \text{если } b_j(t)' \psi^k(t) < 0, \\ \beta_j(t), & \text{если } b_j(t)' \psi^k(t) \geq 0, \end{cases} \quad j = \overline{1, r}, \quad (1.7)$$

где  $\psi^k(t)$  – решение задачи Коши (1.5) при  $\psi(t_1) = -\nabla \varphi(x^k(t_1))$ . Итерация метода условного градиента описывается формулами

$$\begin{aligned} u^{k+1}(t) &= u^k(t) + \alpha_k [\bar{u}^k(t) - u^k(t)], \\ \alpha_k &= \arg \max_{0 \leq \alpha \leq 1} \varphi(x^k(t_1) + \alpha [\bar{x}^k(t_1) - x^k(t_1)]). \end{aligned} \quad (1.8)$$

Для получения точек  $x^k(t_1)$  и  $\bar{x}^k(t_1)$  необходимо интегрировать систему (1.1) при управлениях  $u^k(t)$  и  $\bar{u}^k(t)$ . Используя по координатное представление точки  $x(t_1)$ :

$$x_i(t_1) = c_i + \int_{t_0}^{t_1} \psi^i(t)' B(t) u(t) dt, \quad i = \overline{1, n},$$

можно исключить операцию интегрирования системы (1.1), заменив ее интегрированием более простой системы. Для этого необходимо предварительно найти решения (1.5)  $\psi^i(t)$ , при  $\psi^i(t_1) = e^i$ ,  $i = \overline{1, n}$ ; с помощью этих решений посчитать и запомнить в узлах интегрирования строки  $n \times r$  матрицы  $M(t)$ , равные  $\psi^i(t)'B(t)$ ,  $i = \overline{1, n}$ . Проинтегрировав систему (1.1) при  $u(t) \equiv 0$ , вычислить вектор  $c$ .

Пусть теперь задано допустимое управление  $u^k(t)$ . Тогда итерация (1.8) состоит из следующих операций:

1) решается задача Коши  $\dot{x}^k = M(t)u^k(t)$ ,  $x^k(t_0) = c$ , в результате чего найдется вектор  $x^k(t_1)$ ;

2) решается задача Коши (1.5), и по формулам (1.7) вычисляется управление  $\bar{u}^k(t)$ ;

3) решается задача Коши  $\dot{\bar{x}}^k = M(t)\bar{u}^k(t)$ ,  $\bar{x}^k(t_0) = c$ , после чего будем иметь  $\bar{x}^k(t_1)$ ;

4) решается задача одномерного поиска

$$\varphi(x^k(t_1) + \alpha[\bar{x}^k(t_1) - x^k(t_1)]) \rightarrow \min_{0 \leq \alpha \leq 1};$$

5) по формуле (1.8) строится новое приближение.

Начиная со второй итерации операцию 1) можно не выполнять, так как значение  $x^{k+1}(t_1)$  будет найдено при выполнении операции 4). Однако в целях сохранения устойчивости процесса вектор  $x^{k+1}(t_1)$  через определенное число итераций следует получить с помощью операции 1). Итерационный процесс (1.8) прекращается при выполнении неравенства

$$\varphi(x^k(t_1)) - \varphi(x^k(t_1) + \alpha_k[\bar{x}^k(t_1) - x^k(t_1)]) \leq \varepsilon$$

или необходимого условия оптимальности

$$-\int_{t_0}^{t_1} B'(t)\psi^k(t)[u^k(t) - \bar{u}^k(t)]dt \leq \varepsilon_1,$$

где  $\psi^k(t)$  – решение задачи Коши (1.5). Значение левой части второго неравенства удобно находить при выполнении операции 2). Значения  $\varepsilon$  и  $\varepsilon_1$  определяют точность решения задачи (1.1), (1.5), (1.6) и должны задаваться с учетом погрешностей счета величин, стоящих в левых частях неравенств. Поскольку на практике задать “достижимое” значение  $\varepsilon_1$  не всегда удастся, во избежание заикливания обычно применяется первое неравенство, выполнение которого свидетельствует о прекращении сходимости релаксационного процесса.

Недостаток данного алгоритма состоит в том, что его реализация требует получения и хранения базисного набора решений сопряженной системы, составляющего фундаментальную матрицу. Однако, если размерность задачи такова, что имеющийся объем памяти позволяет реализовать описанный алгоритм, то такой подход будет вполне оправданным поскольку задачи Коши, решаемые в п. 1) и 3), значительно проще, чем задачи Коши (1.1), а число итераций, выполняемых методом условного градиента для получения решения с заданной точностью, обычно бывает достаточно большим. Если необходимый объем памяти  $rnN + \nu n > V$ , где  $V$  – допустимый объем,  $\nu \geq 2$ , то можно реализовать традиционный вариант метода условного градиента, в котором вместо операций 1) и 3) будет интегрироваться система (1.1) при управлениях  $u^k(t)$  и  $\bar{u}^k(t)$  соответственно. Тогда потребуется всего  $rN + \nu n$  слов памяти. Трудоемкость итерации эквивалентна интегрированию  $2n$  скалярных уравнений. Заметим, что учет этих факторов важен в программных системах реального времени, например, в бортовых системах управления летательными аппаратами [13-14].

## 1.2. Минимизация нормы отклонения от заданного состояния

Рассмотрим задачу

$$\|x(t_1) - x^T\| \rightarrow \min, \quad (1.9)$$

где  $x(t_1)$  – решение системы (1.1) при  $t = t_1$ , соответствующее управлению из класса допустимых (1.6). Решение этой задачи, очевидно, можно получить, применяя вместо операции одномерного поиска 4) следующую формулу:

$$\alpha_k = \begin{cases} \alpha^* = \frac{(x^T - x^k(t_1))'(\bar{x}^k(t_1) - x^k(t_1))}{\|\bar{x}^k(t_1) - x^k(t_1)\|^2}, & \text{если } \alpha^* \leq 1; \\ 1, & \text{если } \alpha^* > 1. \end{cases} \quad (1.10)$$

Известно, что метод условного градиента обладает медленной сходимостью, поэтому в его развитие предложены алгоритмы квадратичного программирования [8, 9], которые позволяют получить решение задачи (1.9) за существенно меньшее число итераций.

### 1.2.1. Метод выпуклых оболочек

Основная идея рассматриваемого алгоритма состоит в том, что для уменьшения целевой функции (1.9) максимально используются результаты наиболее трудоемкой операции – решения задачи Коши, полученные на нескольких итерациях. Вместо выпуклой оболочки двух точек  $x^k$  и  $\bar{x}^k$ , на которой отыскивается новое приближение в методе условного градиента, здесь применяется выпуклая оболочка  $s$ ,  $2 \leq s \leq n+1$ , точек множества достижимости. Для поиска минимума функции (1.9) на выпуклой оболочке  $s$  точек решается простейшая задача квадратичного программирования. Приближение, полученное в результате решения этой задачи, значительно лучше приближения, полученного на отрезке методом условного градиента.

Алгоритм начинает работу с допустимого управления  $u^1$ , и на его первой итерации, как и в методе условного градиента, новое приближение строится в виде выпуклой комбинации  $u^1$  и управления (1.7), вычисленного на решении системы (1.5) при

$$\psi(t_1) = g^1 = (x^1(t_1) - x^T) / \|x^1(t_1) - x^T\|.$$

По формуле (1.10) вычислим параметр. Если  $0 < \alpha_k < 1$ , то управления  $u^1$  и  $u^2 = \bar{u}^1$ , а также соответствующие им точки  $x^1 = x^1(t_1)$  и  $x^2 = \bar{x}^1$  дополняются управлением (1.7), найденным при

$$\psi(t_1) = g^2 = \frac{x^1 + \alpha_k(x^2 - x^1) - x^T}{\|x^1 + \alpha_k(x^2 - x^1) - x^T\|},$$

и соответствующей ему точкой  $x^3 = \bar{x}(t_1)$ . Следующее приближение отыскивается на выпуклой оболочке трех точек, содержащей отрезок, на котором ищется новое приближение методом условного градиента. Следовательно, приращение целевой функции (1.9) по модулю будет не меньше чем приращение, достигаемое методом условного градиента. Практика показывает, что метод выпуклых оболочек позволяет найти решение задачи (1.9) за существенно меньшее машинное время.

Опишем  $\nu$ -ю итерацию алгоритма. Пусть на его итерациях получены управления  $u^1, u^2, \dots, u^k$ ,  $k \leq n+1$ , и им соответствующие точки  $x^i$ ,  $i = \overline{1, k}$ , с помощью которых представляется приближение

$$u^\nu = \sum_{i=1}^k \beta_i u^i \text{ и } x^\nu = \sum_{i=1}^k \beta_i x^i,$$

где  $\beta_i \geq 0$ ,  $i = \overline{1, k}$ ,  $\sum_{i=1}^k \beta_i = 1$ .

Тогда шаг алгоритма состоит из следующих операций:

1) на плоскости, проходящей через точки  $x^i$ ,  $i = \overline{1, k}$ , отыскивается ближайшая к  $x^T$  точка  $y$ . Для этого решается задача безусловной минимизации

$$\left\| \left[ x^k + \sum_{i=1}^{k-1} \alpha_i (x^i - x^k) \right] - x^T \right\| \rightarrow \min,$$

решение которой удовлетворяет линейной системе алгебраических уравнений

$$\sum_{j=1}^{k-1} \alpha_j (x^i - x^k)' (x^j - x^k) + (x^i - x^k)' (x^k - x^T) = 0, \quad i = \overline{1, k-1}. \quad (1.11)$$

С помощью решения данной системы и  $\alpha_k = 1 - \sum_{i=1}^k \alpha_i$  легко строится

искомая точка на плоскости:  $y = \sum_{i=1}^k \alpha_i x^i$ ;

2) если среди  $\alpha_i$ ,  $i = \overline{1, k}$ , имеются отрицательные и, следовательно, точка  $y$  не принадлежит выпуклой оболочке точек  $x^i$ ,  $i = \overline{1, k}$ , то на отрезке, содержащем точки  $y$  и  $x^v$ , отыскивается точка пересечения отрезка с гранью выпуклой оболочки:

$$z = y + \sigma(x^v - y), \quad \text{где}$$

$$\sigma = \max_{\alpha_i < 0} \{ \alpha_i / (\alpha_i - \beta_i) \} = \alpha_{i_0} / (\alpha_{i_0} - \beta_{i_0}), \quad \text{или} \quad z = \sum_{i=1}^k \gamma_i x^i, \quad \gamma_i = \alpha_i + \sigma(\beta_i - \alpha_i).$$

Очевидно,  $\gamma_{i_0} = 0$ , поэтому точка  $x^{i_0}$  и управление  $u^{i_0}$  исключаются из дальнейшего рассмотрения;

3) если число оставшихся точек  $s \leq 2$ , то  $k = k - 1$  и повторяется операция 1), а при наличии  $\alpha_i < 0$  – операция 2). При  $s = 1$  выполняется операция 5;

4) если после выполнения операции 1) имеем  $\alpha_i > 0$ ,  $i = \overline{1, s}$ ,  $s \leq k$ , то точка  $y$  – решение задачи на выпуклой оболочке точек  $x^i$ ,  $i = \overline{1, k}$ , а управление, приводящее в эту точку, найдется по формуле

$$u(t) = \sum_{i=1}^s \alpha_i u^i(t), \quad t \in [t_0, t_1]; \quad (1.12)$$

5) оставшиеся точки  $x^i$ ,  $i = \overline{1, s}$ , дополняются новой точкой  $x^{s+1} = \bar{x}(t_1)$ , соответствующей управлению (1.7), посчитанному на решении задачи Коши (1.5) при  $\psi(t_1) = (y - x^T) / \|y - x^T\|$ , и затем повторяются операции 1)–5) при  $k = s + 1$ . Процесс прекращается при выполнении неравенства

$$\|y - x^T\|^2 - (g^{s+1})'(\bar{x}(t_1) - x^T) \leq \varepsilon,$$

которое следует проверять после окончания операции 5) ( $\varepsilon$  – заданная точность решения задачи (1.9)). Найденная при этом точка  $y$  является проекцией  $x^T$  на множество достижимости:  $y = \arg \min \|x - x^T\|$ . Управление (1.12), приводящее систему (1.1) в точку  $y$ , будет оптимальным для задачи (1.9).

Трудоемкость итерации описанного алгоритма незначительно больше, чем у метода условного градиента, поскольку вычисление проекции точки  $x^T$  на выпуклую оболочку заданных точек не содержит трудоемких операций интегрирования систем. Между тем число внешних итераций, выполняющих интегрирование прямой и сопряженной систем, существенно сокращается, и, следовательно, алгоритм требует значительно меньше времени центрального процессора для решения задачи (1.9) с заданной точностью.

Требования алгоритма к памяти определяются в основном числом управлений  $u^i$ ,  $i = \overline{1, s}$ , и их размерностью  $r$ . Учитывая релейную структуру управлений (1.7), для хранения их в памяти достаточно запомнить лишь точки

переключения каждой управляющей функции и ее начальных значений:  $u_j^i(t_0)$ ,  $j = \overline{1, r}$ ,  $i = \overline{1, s}$ , что требует  $(n+2)$  слов памяти. Кроме того, необходим объем памяти для хранения управления (1.12), точек  $x^i$ ,  $i = \overline{1, s}$ ,  $s \leq n+1$ , а также векторов, с помощью которых реализуются решения линейной системы алгебраических уравнений и численное интегрирование. Таким образом, необходимый объем памяти составляет  $rN + \mu n + (n+1)n + (n+2)r$ ,  $\mu \geq 5$ , слов.

### 1.2.2. Модифицированные алгоритмы выпуклых оболочек

Рассмотрим модификации метода выпуклых оболочек, реализация которых не требует запоминания управлений и вследствие этого допускает решение задач с непрерывными  $u^i$ ,  $i = \overline{1, s}$ . Такие управления получаются, например, при решении задач с ограничениями вида

$$\|u(t)\| \leq l, \quad t \in [t_0, t_1]. \quad (1.13)$$

Вместо управлений  $u^i$ ,  $i = \overline{1, s}$ , на итерациях модифицированных алгоритмов запоминаются порождающие их векторы  $g^i$ ,  $i = \overline{1, s}$ , – начальные условия для сопряженной системы (1.5). По первой модификации после окончания итерационного процесса путем интегрирования системы (1.5) с  $\psi(t_1) = g^i$ ,  $i = \overline{1, s}$ , восстанавливается управление (1.7) и строится оптимальное управление (1.12).

Согласно второй модификации до начала итерационного процесса находятся решения  $\psi^i(t)$ ,  $i = \overline{1, n}$ , с условиями  $\psi^i(t_1) = e^i$ . Тогда для генерирования новой точки  $x^{s+1}$  (операция 5)) можно использовать решение задачи Коши

$$\dot{x}_i = \psi^i(t)' B(t) \bar{u}(t), \quad x_i(t_0) = c_i, \quad i = \overline{1, n},$$

где при ограничениях (1.13)

$$\bar{u}(t) = (l / \|B'(t)\psi(t)\|) B'(t)\psi(t). \quad (1.14)$$

После завершения итерационного процесса с помощью векторов  $g^j$ ,  $j = \overline{1, s}$ , и фундаментального набора решений  $\psi^i(t)$ ,  $i = \overline{1, n}$ , находятся  $\psi^j(t) = \sum_{i=1}^n g_i^j \psi^i(t)$ ,  $j = \overline{1, s}$ , восстанавливаются управления (1.7) или (1.14) и строится оптимальное управление (1.12). Реализация данного алгоритма требует  $rN(n+1) + \nu n + 2n^2$ ,  $\nu \geq 3$ , слов памяти. Первая модификация метода выпуклых оболочек наименее требовательна к объему памяти – для ее реализации достаточно  $rN + \nu n + 2n^2$  слов.

### 1.2.3. Алгоритм, основанный на теореме о минимаксе

Для решения задачи (1.9) с ограничениями на управление (1.13)

построим алгоритм, основанный на теоремах об отделимости и о минимаксе [10]. Если множество достижимости  $X(l)$  строго выпукло и содержит точку  $x^T$ , то данный алгоритм позволяет получить управление, которое с минимальной интенсивностью  $l^* \leq l$  переводит систему (1.1) в точку  $x^T$ , и следовательно, на этом управлении функционал (1.9) достигает абсолютного минимума. В случае, когда  $x^T \notin X(l)$ , данный алгоритм найдет управление, которое переводит систему (1.1) из  $x^0$  в точку множества  $X(l)$ , ближайшую к точке  $x^T$ .

Пусть при  $u(t) \equiv 0$  решена задача Коши (1.1), найден вектор  $c = x(t_1)$  и задано первое приближение

$$g^1 = (x(t_1) - x^T) / \|x(t_1) - x^T\|.$$

Тогда  $\lambda(0, g^1) = g^{1'}(x(t_1) - x^T) > 0$  и итерация алгоритма будет состоять из следующих операций:

1) при заданном  $l$  вычисляется значение функции

$$\lambda(l, g^1) = g^{1'}(x(g^1) - x^T);$$

2) если  $\lambda(l, g^1) < 0$ , то находится  $l_1 < l$ , при котором  $\lambda(l_1, g^1) = 0$ :

$$l_1 = l \frac{\lambda(0, g^1)}{\lambda(0, g^1) - \lambda(l, g^1)},$$

полагаем  $k = 1$ ;

3) Выполнив операцию максимизации, найдем вектор  $g^{k+1} = g(\alpha_k)$  такой, что  $\lambda(l_k, g^{k+1}) > 0$ ;

4) находится новое значение

$$l_{k+1} = \frac{g^{k'}(c - x^T)}{g^{k'}(c - x^T) - \lambda(l_k, g^k)} l_k,$$

для которого  $\lambda(l_{k+1}, g^{k+1}) = 0$ ;

5) если  $l_{k+1} < l$  и  $l_{k+1} - l_k > \varepsilon_l$ , то повторяются операции 3) и 4). При  $l_{k+1} - l_k \leq \varepsilon_l$  решение задачи прекращается, и минимальная интенсивность управления полагается равной  $l_{k+1}$ ;

6) если  $l_{k+1} > l$  и, следовательно,  $\lambda(l, g^k) > 0$ , то процесс расширения множества достижимости  $X(l)$  прекращается и в дальнейшем при заданном  $l$  повторяются итерации основного алгоритма до выполнения неравенства

$$\|x(g^k) - x^T\| - \lambda(l, g^k) \leq \varepsilon. \quad (1.15)$$

Управление, с которым на последней итерации найдена точка  $x(g^k)$ , будет оптимальным.

Сходимость данного алгоритма имеет место лишь в случае строго выпуклого множества  $X(l)$ . Поэтому в общем случае возможно заикливание итерационного процесса. Чтобы избежать заикливания, следует ввести дополнительное условие прекращения процесса

$$\lambda(l, g^{k+1}) - \lambda(l, g^k) \leq \varepsilon_\lambda,$$

где  $\varepsilon_\lambda$  – оценка погрешностей вычисления функции  $\lambda(l, g)$ . Если это условие выполняется при нарушении неравенства (1.15), то данный алгоритм не может обеспечить решение задачи (1.9) и для дальнейшего поиска оптимального управления необходимо использовать изложенные выше алгоритмы.

По сравнению с рассмотренными ранее алгоритмами, преимуществом данного алгоритма является небольшие требования к объему памяти, пропорциональные размерности системы. Следовательно, он применим к системам (1.1) высокого порядка.

## 2. Алгоритмы решения задачи быстродействия

Рассмотрим задачу линейного быстродействия: перевод системы (1.1) из точки  $x^0$  в точку  $x^T$  при ограничениях на управление (1.6) за наименьшее время.

### 2.1. Минимаксный алгоритм

Один из первых алгоритмов [4], предложенных для решения задачи линейного быстродействия, основан на теореме об отделимости выпуклых множеств. По своей конструкции он близок к методу минимизации интенсивности управления. Вместо наименьшего значения параметра  $l = l^*$  алгоритмы быстродействия отыскивают минимальный отрезок времени  $t^* - t_0$ , за который переводима система (1.1) из  $x^0$  в  $x^T$  при заданных ограничениях на управление. Предполагая управляемость системы (1.1), построим алгоритм, который генерирует сходящуюся возрастающую последовательность моментов времени  $t_1, t_2, \dots$  с двумя основными свойствами:

1) вдоль последовательности  $\{t_k\}$  множество достижимости

$$X(t_k) = \left\{ x \in R^n : x = c + \int_{t_0}^{t_k} \Phi(t_k, t) B(t) u(t) dt, \quad |u_j(t)| \leq l, \quad j = \overline{1, r}, \quad t \in [t_0, t_k] \right\}$$

расширяется  $(X(t_k) \subset X(t_{k+1}))$ ,

2) в момент  $t^* = \lim_{k \rightarrow \infty} t_k$  происходит первое касание точки  $x^T$ .

Для определения значений  $t_k, k = 1, 2, \dots$ , будем применять следующее условие:

$$\psi(t_k)'(x(t_k) - x^T) = 0,$$

проверка которого требует одновременного интегрирования систем (1.1) и (1.5) в прямом времени.

При каждом фиксированном значении  $t_k$  корректировка вектора  $g$  – нормали опорной плоскости – осуществляется посредством решения задачи  $\lambda(g, t_k) \rightarrow \max_{\|g\| \leq 1}$ .

Иными словами, полагается

$$g^{k+1} = \arg \max_{\|g\| \leq 1} \lambda(g, t_k).$$

Введем множества:

$$G_+^k = \{g \in R^n : \lambda(g, t_k) > 0, \|g\| = 1\}, G_0^k = \{g \in R^n : \lambda(g, t_k) = 0, \|g\| = 1\}.$$

Пусть  $g^0 \in G_0^0$ . Тогда  $k$ -й шаг минимаксного алгоритма состоит из следующих операций:

1) при управлении (1.7) в прямом времени интегрируется система

$$\dot{x} = A(t)x + \sum_{j=1}^r b_j(t)\bar{u}_j(t) + f(t), \quad x(t_k) = \bar{x}(t_k) = x(g_k, t_k), \quad (2.1)$$

$$\dot{\psi} = -A(t)'\psi, \quad \psi(t_k) = g^k, \quad t \geq t_k,$$

до первого корня  $t_{k+1}$  функции  $\psi(t)'(x(t) - x^T) : \psi(t_{k+1})'(x(t_{k+1}) - x^T) = 0$ ;

2) при фиксированном  $t_{k+1}$ , начиная с  $g = \psi(t_{k+1})$ , выполняется несколько шагов градиентного метода. В результате находится  $g^{k+1} \in G_+^{k+1}$ .

Операции 1), 2) повторяются, пока градиентный метод обеспечивает выполнение неравенства  $\lambda(g^{k+1}, t_{k+1}) > 0$ , а интегрирование системы (2.1) позволяет получить  $t_{k+1} > t_k$ . Для управляемой системы данный итерационный процесс сходится к  $(g^*, t^*)$ :

$$\lambda(g^*, t^*) = \min_{x(t^*) \in X(t^*)} \|x(t^*) - x^T\| = 0, \quad \lambda(g^*, t^*) = \max_{g \in G_+^k} \lambda(g, t^*).$$

При этом управление (1.7), посчитанное на решении системы (1.5) при  $\psi(t_1) = g$ , будет оптимальным по быстродействию.

Недостатками минимаксного алгоритма являются, с одной стороны, необходимость интегрирования сопряженной системы в прямом времени и, с другой – гарантированная вычислительная устойчивость только для строго выпуклых множеств достижимости.

Трудоемкость итерации алгоритма оценивается трудоемкостью интегрирования  $2n(\nu+1)$  скалярных уравнений на отрезке  $[t_0, t_k]$ , где  $\nu$  – число пересчетов  $\lambda(g, t_k)$ , при вычислении параметра  $\alpha_k$ . Требуемый объем памяти –  $\gamma n$  слов,  $\gamma \leq 5$ .

## 2.2. Алгоритм выпуклых оболочек

Рассмотрим алгоритм, в котором вместо градиентных процедур для попадания в область  $G_+^k$  применяется итерационная процедура из раздела 1.2.1. Тогда в качестве вектора  $g$  берется приближенное решение задачи (1.9):

$$g^k = (x^k(t_k) - x^T) / \|x^k(t_k) - x^T\|. \quad (2.2)$$

Поскольку  $\max_{\|g\| \leq 1} \lambda(g, t_k) = \min_{x \in X(t_k)} \|x - x^T\|$ , при  $x^k(t_k) \neq x^T$  найдется такой номер  $k$ , для которого  $g^k$  вида (2.2) будет принадлежать  $G_+^k$ . В противном случае  $x^T \in X(t_k)$  и при  $g^k \in G_0^k$  вычисляется оптимальное управление (1.7).

Пусть  $g^0 = (x^0 - x^T) / \|x^0 - x^T\|$ ,  $k = 0$ . Тогда итерация алгоритма состоит из следующих операций:

1) начиная с  $x(t_k) = \bar{x}(t_k)$ ,  $\psi(t_k) = g^k$  интегрируется  $2n$ -мерная система (2.1) от  $t_k$  до ближайшего к нему корня  $t_{k+1}$  уравнения

$$\psi(t)'(x(t) - x^T) = 0;$$

2) задается входная информация для алгоритма 1.2.1:  $x^1 = \bar{x}(t_k)$ ,  $u^1 = \bar{u}(t)$  – управление (1.7), приводящее (1.1) в  $\bar{x}(t_k)$ .

Затем при  $\psi(t_k) = (x^1 - x^T) / \|x^1 - x^T\|$  интегрируется система (1.5) и по формуле (1.7) считается управление  $u^2 = u(t)$ . Решение системы, соответствующее  $u^2(t)$ , при  $t = t_1$  берется в качестве  $x^2$ ;

3) начиная с  $k = 2$  реализуются итерации алгоритма 1.2.1, пока не выполнится неравенство

$$(g^{k+1})'(\bar{x}(t_k) - x^T) = \lambda(g^{k+1}, t_k) > 0,$$

где  $g^{k+1} = (y^v - x^T) / \|y^v - x^T\|$ ,  $y^v$  – полученное приближение для задачи (1.9). Затем  $k$  меняется на  $k + 1$  и повторяются операции 1)-3).

Сходимость этого процесса следует из сходимости метода условного градиента и имеет место для произвольного выпуклого множества.

Итерационный процесс прекращается при выполнении неравенства  $\|y^v(t_k) - x^T\| < \varepsilon$ . Тогда оптимальное управление считается по формуле (1.12).

Решение практических задач показывает также, что данный алгоритм требует меньше машинного времени для получения оптимального по быстродействию управления, переводящего систему (1.1) из  $x^0$  в  $x^T$  с заданной точностью. Одним из недостатков алгоритма является неустойчивость операции прямого интегрирования сопряженной системы при получении значений  $t_k$ ,  $k = 1, 2, \dots$ . Для некоторых систем этот недостаток может оказаться существенным и требует специальных способов интегрирования системы (2.1).

## **Заключение**

Получение численного решения задачи оптимального управления при наименьших затратах вычислительных ресурсов особенно актуально в управляемых системах реального времени, например в бортовых системах управления летательными аппаратами. Путем многократного решения задачи оптимального программного управления в некоторых случаях можно получить решение задачи синтеза оптимального управления. Так, с помощью оптимальных по быстродействию программных управлений, полученных для разных начальных состояний, в одной задаче преследования [13, 14] построена линия переключения управления в фазовом пространстве.

Для поиска оптимального по быстродействию управления построены достаточно эффективные алгоритмы, итерации которых ведутся в конечномерном пространстве, и с помощью их находятся базисные точки множества достижимости. По окончании итерационного процесса восстанавливаются управления, соответствующие найденным базисным точкам, и строится их выпуклая комбинация, которая и является оптимальным управлением. В отличие от прямых методов эти алгоритмы не требуют большого объема памяти и могут применяться в задачах с большим числом управлений. В сравнении с “чисто” двойственными методами, эффективность которых теряется, если множество достижимости не является строго выпуклым, данные алгоритмы обладают более высокой вычислительной устойчивостью. Практика показывает также, что по скорости сходимости эти алгоритмы превосходят прямые и двойственные градиентные методы.

Основой алгоритмического обеспечения задач оптимального управления с фазовыми ограничениями являются специальные методы решения больших задач линейного программирования. Для решения задач линейного программирования, полученных дискретизацией линейных задач оптимального управления с фазовыми ограничениями, применяются различные по трудоемкости и по требованиям к объему памяти алгоритмы [11], учитывающие специфические свойства этих задач.

## **Список литературы**

1. Андреев Ю.Н. Управление конечномерными линейными объектами. – М.: Наука, 1976. – 424 с.
2. Калман Р., Фалб П., Арбиб М. Очерки по математической теории систем. – М.: Мир, 1971. – 400 с.
3. Красовский Н.Н. Теория управления движением. – М.: Наука, 1968. – 476 с.
4. Eaton J.H. An iterative solution to time-optimal control // J. Math. Anal. and Appl. – 1962. – Vol. 5, No. 2. – P. 329-344.
5. Васильев О.В., Тятюшкин А.И. К численному решению задач линейного быстродействия // Дифференциальные и интегральные уравнения. – Иркутск. – 1973. – Вып. 2. – С. 57-69.
6. Тятюшкин А.И. Многометодная технология оптимизации управляемых систем. – Новосибирск: Наука, 2006. – 343 с.
7. Tyatyushkin A.I. A multimethod technique for solving optimal control problem // Optimization Letters. – 2012. – Vol. 7. – P. 1335-1347.
8. Barr R.O. An efficient computational procedure for a generalized quadratic

- programming problem // J. SIAM Control. – 1969. – Vol. 7, No. 3. – P. 415-429.
9. Gibert E.G. An iterative procedure for computing the minimum of a quadratic form on a convex set // J. SIAM Control. – 1966. – Vol. 4, No. 1. – P. 61-80.
  10. Габасов Р., Кириллова Ф.М. Качественная теория оптимальных процессов. – М.: Наука, 1971. – 508 с.
  11. Габасов Р., Кириллова Ф.М., Тятюшкин А.И. Конструктивные методы оптимизации. Ч. 1: Линейные задачи. – Минск: Университетское, 1984. – 214 с.
  12. Табак Д., Куо Б. Оптимальное управление и математическое программирование. – М.: Наука, 1975. – 280 с.
  13. Тятюшкин А.И., Федун Б.Е. Численное исследование свойств оптимального управления в одной задаче преследования // Изв. РАН, ТиСУ. – 2005. – № 3. – С. 104-113.
  14. Тятюшкин А.И., Федун Б.Е. Возможности защиты от атакующей ракеты задней полусферы самолета вертикальным маневром // Изв. РАН, ТиСУ. – 2006. – № 1. – С. 111-125.

### References

1. Andreev Yu.N. Control of finite-dimensional linear objects. – M.: Science, 1976. – 424p.
2. Kalman R., Falb P., Arbib M. Essays on Mathematical Theory of Systems. – M.: World, 1971. – 400 p.
3. Krasovsky N.N. Theory of motion control. – M.: Science, 1968. – 476 p.
4. Eaton J.H. An iterative solution to time-optimal control // J. Math. Anal. and Appl. – 1962. – Vol. 5, No. 2. – P. 329-344.
5. Vasiliev O.V., Tyatyushkin A.I. To the numerical solution of problems of linear speed // Differential and integral equations. – Irkutsk. – 1973. – Issue 2. – P. 57-69.
6. Tyatyushkin A.I. Multimethod technology for optimization of controlled systems. – Novosibirsk: Science, 2006. – 343 p.
7. Tyatyushkin A.I. A multimethod technique for solving optimal control problem // Optimization Letters. – 2012. – Vol. 7. – P. 1335-1347.
8. Barr R.O. An efficient computational procedure for a generalized quadratic programming problem // J. SIAM Control. – 1969. – Vol. 7, No. 3. – P. 415-429.
9. Gibert E.G. An iterative procedure for computing the minimum of a quadratic form on a convex set // J. SIAM Control. – 1966. – Vol. 4, No. 1. – P. 61-80.
10. Gabasov R., Kirillova F.M. Qualitative theory of optimal processes. – M.: Science, 1971. – 508 p.
11. Gabasov R., Kirillova F.M., Tyatyushkin A.I. Constructive optimization methods. Part 1: Linear problems. – Minsk: University, 1984. – 214 p.
12. Tabak D., Kuo B. Optimal control and mathematical programming. – M.: Science, 1975. – 280 p.
13. Tyatyushkin A.I., Fedunov B.E. Numerical study of the properties of optimal control in one problem of pursuit // News of RAS, TiSU. – 2005. – No. 3. – P. 104-113.
14. Tyatyushkin A.I., Fedunov B.E. Possibilities of defense against attacking missiles of the rear hemisphere of the aircraft by vertical maneuver. // News of RAS, TiSU. – 2006. – No. 1. – P. 111-125.

|                                                                                                                   |                                                                                                      |
|-------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------|
| <b>Тятюшкин Александр Иванович</b> – доктор технических наук, профессор, ведущий научный сотрудник<br>tjat@icc.ru | <b>Tyatyushkin Alexander Ivanovich</b> – doctor of technical sciences, professor, leading researcher |
|-------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------|

Received 27.07.2022