ISBN 978-5-901795-16-3 ПРОГРАММНЫЕ СИСТЕМЫ: ТЕОРИЯ И ПРИЛОЖЕНИЯ. Переславль-Залесский, 2009

УДК 517

Е. Ф. Сачкова

Реализация и анализ работы алгоритмов приближенного решения задачи управления

Аннотация. Рассматривается компьютерная реализация алгоритма приближенного решения задачи управления для трехмерных нелинейных систем, описываемых обыкновенными дифференциальными уравнениями, с двумя линейными управлениями. Алгоритм основан на методе нильпотентной аппроксимации. Он реализован в системе Maple и апробирован на примере управления ориентацией катящейся по плоскости сферы.

1. Введение

Настоящая работа посвящена описанию программной реализации алгоритма перемещения трехмерной нелинейной системы в двумя линейными управлениями из заданного начального состояния в малую окрестность заданного финального состояния. Алгоритм разработан в предыдущей статье [1]. Написанная в системе компьютерной математики Maple [2] программа осуществляет итерационный процесс последовательного приближения системы к цели с помощью управлений другой, более простой, управляемой системы, являющейся аппроксимацией исходной. Широкие возможности программы обусловлены эффективностью как алгоритма, так и численных и символьных расчетов, осуществляемых системой Maple, поэтому она может быть полезна при решении большого числа прикладных задач, возникающих при управлении мобильными роботами [3], спутниками (при неограниченных управлениях) и иными импульсными системами [4, 5], качением твердых тел [6, 7].

Приводится пример апробации программы при решении задачи управления ориентацией сферы, которая катится по плоскости без проскальзывания и прокручивания. Рассмотрены управления, ранее не апробированные: кусочно-постоянные с одним переключением (см. [8]); управления, построенные с помощью линейных векторных полей на плоскости, имеющих особенность типа центр и типа фокус

Работа поддержана Российским Фондом Фундаментальных Исследований, проект No. 09-01-00246.

(см. [9]). Проанализирована эффективность соответствующих алгоритмов.

2. Постановка задачи управления

Рассматривается управляемая система вида

$$\dot{x} = u_1 X_1(x) + u_2 X_2(x), \qquad x \in \mathbb{R}^3, \qquad u = (u_1, u_2) \in \mathbb{R}^2, \tag{1}$$

$$X_1,\ X_2,\ X_3 = [X_1, X_2] = \frac{\partial X_2}{\partial x} X_1 - \frac{\partial X_1}{\partial x} X_2, \tag{2}$$

— линейно независимые гладкие векторные поля в $\mathbb{R}^3$,

с заданными граничными условиями и точностью:

$$x(0) = x^0, \quad x(T) = x^1, \quad x^0,\, x^1 \in \mathbb{R}^3, \quad T > 0, \quad \varepsilon > 0. \tag{3}$$

Требуется переместить систему (1), удовлетворяющую условию (2), из начального состояния x^0 в ε-окрестность конечного состояния x^1 за время $T > 0$.

Задача (1)–(3) разрешима в $\mathbb{R}^3$, так как из условия (2) следует полная управляемость системы (1) в $\mathbb{R}^3$ ([6], с. 73, 78).

В предыдущих работах [1, 8, 9] рассмотрен подход к решению задачи (1)–(3), основанный на методе нильпотентной аппроксимации системы (1), (2) в окрестности точки x^1.

Напомним необходимые сведения. Система вида (1) называется *нильпотентной*, если ее алгебра Ли

$$\begin{aligned}&\mathrm{Lie}(X_1, X_2) = \\ &\quad = \mathrm{span}(X_1, X_2, [X_1, X_2], \dots, [X_{i_1}, [X_{i_2}, \dots, [X_{i_k}, X_{i_{k+1}}] \dots]], \dots)\end{aligned}$$

нильпотентна, то есть для некоторого номера N все скобки Ли длины N равны нулю:

$$[X_{i_1}, [X_{i_2}, \dots, [X_{i_N}, X_{i_{N+1}}] \dots]] = 0 \quad \forall\, i_1, \dots, i_N, i_{N+1} \in \{1, 2\}.$$

Для систем (1), (2) нильпотентные аппроксимации, согласно теореме Белаиша ([1, 10]), имеют $N = 2$. Алгоритм нильпотентизации систем (1), (2) разработан в [1] и реализован в виде процедуры NilpApprox() программы FindControlLoc.

При решении задачи (1)–(3) в алгоритме используются две управляемые системы: исходная система (1), (2) и ее нильпотентная аппроксимация в окрестности точки x^1 (ее коэффициенты c_{ij}, $i, j \in \{1, 2\}$ см. в [1]), вычисленная в некоторых специальных координатах,

связанных с исходной системой (называемых привилегированными координатами):

$$\begin{aligned} &\dot z_1 = u_1, \quad \dot z_2 = u_2, \\ &\dot z_3 = u_1(c_{11}z_1 + c_{12}z_2) + u_2(c_{21}z_1 + c_{22}z_2), \qquad c_{12} \neq c_{21}, \end{aligned} \tag{4}$$

которая сводится заменой переменных

$$G(z) = \left(z_1,\ z_2,\ \frac{1}{c_{21} - c_{12}} \left(z_3 - \frac{c_{21} + c_{12}}{2} z_1 z_2 - \frac{c_{11}}{2} z_1^2 - \frac{c_{22}}{2} z_2^2 \right) \right) \tag{5}$$

к симметричной нильпотентной системе

$$\dot y_1 = u_1, \qquad \dot y_2 = u_2, \qquad \dot y_3 = (u_2 y_1 - u_1 y_2)/2. \tag{6}$$

Заметим, что граничным состояниям $(x^0,\ x^1)$ исходной системы соответствуют граничные состояния $(y^0, 0)$ системы (6).

Метод приближенного решения задачи управления (1)–(3) заключается в последовательном приближении системы (1) к целевой точке x^1 с помощью управлений системы (6), вычисляемых на каждой итерации и точно переводящих систему (6) в целевую точку.

Решения задачи управления

$$y(0) = y^0, \quad y(T) = 0, \quad T > 0 \tag{7}$$

для системы (6) найдены в пяти классах управлений: в тригонометрическом, кусочно-постоянном с одним переключением, оптимальном в смысле минимума функционала субримановой длины ([8]); центральном и фокусном, построенных с помощью линейных векторных полей на плоскости, имеющих соответствующую особенность типа центр и типа фокус (см. [9]). Все эти управления реализованы в виде пяти процедур программы FindControlLoc и составляют библиотеку управлений NilpControls.

3. Вычислительный алгоритм

Основываясь на работе [1], напомним алгоритм приближенного решения задачи управления (1)–(3). Этот итерационный алгоритм основан на методе нильпотентной аппроксимации. Из общей теории [11] следует, что для любой точки x^1 существует радиус сходимости $\delta > 0$ этого алгоритма, то есть такое число $\delta = \delta(x^1) > 0$, что для всех точек x^0, $|x^0 - x^1| < \delta$, построенная далее в алгоритме последовательность

приближений q^n сходится к x^1. Будем далее решать задачу перемещения системы (1) из точки x^0 в точку x^1 при условии $|x^0 - x^1| < \delta$; такую задачу управления будем называть *локальной*.

Обозначим через $\mathcal{U}$ класс управлений, используемых в алгоритме для перемещения системы (например, оптимальных в смысле некоторого функционала, тригонометрических, кусочно-постоянных). Напомним, что используемая далее матрица $F(x)$ имеет вид:

$$F(x) = (X_1 \mid X_2 \mid X_3)\,, \tag{8}$$

т. е. составлена по столбцам из векторов X_1, X_2 правой части системы (1) и их коммутатора X_3.

Алгоритм приближенного решения локальной задачи управления (1)–(3):

(1) **Проверка условия достижения цели:** если $|x^0 - x^1| < \varepsilon$, то цель достигнута и алгоритм останавливается. Далее предполагается, что $|x^0 - x^1| \geqslant \varepsilon$.

(2) **Вычисление нильпотентной аппроксимации** исходной системы (1), (2) в окрестности точки x^1: вычисляются коммутатор $X_3 = [X_1,\, X_2]$, матрица $F(x) = (X_1, X_2, X_3)(x)$, коэффициенты c_{ij}, $i,\, j = 1,\, 2$ по формулам (8)–(11) статьи [1].

(3) **Итерационный процесс.** В качестве начального приближения на первой итерации берется $q^0 = x^0$. Пусть q^{n-1} — приближение к целевой точке x^1, полученное на $(n-1)$-ой итерации.

   (a) Выбирается класс управлений $\mathcal{U}$.

   (b) Вычисляются координаты начальной точки q^{n-1} в привилегированных координатах, центрированных в целевой точке x^1: $z^{n-1} = F^{-1}(q^{n-1})(q^{n-1} - x^1)$.

   (c) Вычисляются координаты y^{n-1} начальной точки q^{n-1} в системе координат (y_1, y_2, y_3) системы (6):

$$y^{n-1} = G(c_{ij}, z^{n-1}),$$

   где отображение G задано формулой (5).

   (d) По формулам работ [8,9] вычисляются управления $\widehat{u}^n \in \mathcal{U}$, переводящие систему (6) из точки y^{n-1} в точку $0 \in \mathbb{R}^3$ за время T.

(e) Решается задача Коши для исходной системы (1) с управлениями $\widehat{u}^n$:

$$\dot{x} = \widehat{u}_1^n(t)X_1(x) + \widehat{u}_2^n(t)X_2(x),$$
$$x(0) = q^{n-1}, \quad t \in [0,\, T];$$

обозначим ее решение через $x^n(t)$.

(f) В качестве следующего приближения берется точка $q^n = x^n(T)$.

(g) Проверяется условие достижения цели: если $|q^n - x^1| < \varepsilon$, то цель достигнута и алгоритм останавливается. Если $|q^n - x^1| \geqslant \varepsilon$, то совершается переход к следующей итерации, к пункту (3a), и в качестве начального приближения берется q^n. Из сходимости алгоритма при условии $|x^0 - x^1| < \delta$ следует, что на некоторой итерации N выполнится условие $|q^N - x^1| < \varepsilon$, и алгоритм остановится.

(4) **Приближенное решение локальной задачи управления** дается последовательным применением управлений $\widehat{u}^1$, …, $\widehat{u}^N$, вычисленных на каждой итерации и перепараметризованных соответствующим образом:

$$u(t) = \begin{cases} N\widehat{u}^1(Nt), & t \in [0, T/N], \\ N\widehat{u}^2(Nt - T), & t \in [T/N, 2T/N], \\ \dots & \\ N\widehat{u}^N(Nt - (N-1)T), & t \in [T(N-1)/N, T]. \end{cases} \tag{9}$$

Смысл этих формул в следующем: если, например, управление $\widehat{u}^1(t)$ переводит точку q^0 в точку q^1 на отрезке времени длины T, то управление $N\widehat{u}^1(Nt)$ переводит точку q^0 в точку q^1 на отрезке времени длины T/N, и т.д.; в результате управление $u(t)$ определено на отрезке $t \in [0, T]$. Управление $u(t) = u(x^0, x^1, T, \varepsilon; t)$, полученное с помощью формул (9), переводит систему (1) за время $T > 0$ из точки x^0 в точку x^1 с заданной точностью $\varepsilon > 0$, следовательно, является приближенным решением локальной задачи управления (1)–(3).

С помощью этого локального алгоритма можно построить и глобальный алгоритм, введя по некоторому правилу промежуточные узлы $x_1^1, \dots, x_k^1$ так, что $x_1^1 = x^0$, $x_k^1 = x^1$ и $|x_{i+1}^1 - x_i^1| < \delta(x_i^1)$.

4. Описание программы FindControlLoc

Локальный алгоритм решения задачи управления (1)–(3) реализован в виде компьютерной программы FindControlLoc, написанной на входном языке системы Maple.

Отметим существенные характеристики программы:

1) возможность рассматривать произвольные системы вида (1), (2);

2) возможность выбирать произвольные граничные условия (3) (близкие в смысле δ);

3) решать задачу управления (1)–(3) в нескольких классах управлений.

Гибкость программы FindControlLoc обусловлена, в частности, тем, что она использует средства Maple-языка — языка процедурного программирования. Фрагменты алгоритма реализованы в нескольких процедурах, которые вызываются из тела программы. Это позволяет использовать один и тот же код для различных классов управлений, экономно проводить вычисления, например, для фиксированного финального состояния вычислять нильпотентную аппроксимацию один раз.

4.1. Описание процедур программы FindControlLoc

Процедуры вычислительного алгоритма. Описываемые ниже две процедуры реализуют п. 2, п. 3, (b),(c) алгоритма, изложенного в разделе 3.

Процедура NilpApprox();

Вычисляется нильпотентная аппроксимация системы (1), (2) в точке $x \in \mathbb{R}^3$.

Входные данные: $X_1,\ X_2 \in \mathrm{Vec}(\mathbb{R}^3)$, $x \in \mathbb{R}^3$;

Выполняемые действия: с помощью операций символьного дифференцирования и функций пакета расширения Maple linalg в символьном виде

1) вычисляется матрица $F^{-1}(x)$, где $F(x)$ — матрица (8);

2) вычисляются коэффициенты $c_{ij}(x)$, $i, j = 1, 2$, (см. [1], формулы (8) – (11)) системы (4) в точке $x \in \mathbb{R}^3$.

Выходные данные: 1) символьная 3×3 матрица $F^{-1}(x)$; 2) символьная 2×2 матрица $C(x)$, $C_{ij} = c_{ij}$, $i, j = 1, 2$.

Процедура ChangeCoords();

Выполняется замена переменных в окрестности точки $q^1 \in \mathbb{R}^3_x$: $\phi(x) = y$, $\phi(q^1) = 0$, где (x_1, x_2, x_3) — координаты исходной системы (1), (y_1, y_2, y_3) — координаты системы (6).

Входные данные: q^0, q^1, $C(q^1)$, $F^{-1}(x)$,

где q^0, q^1 — векторы длины 3; $C(q^1)$ — 2×2 матрица коэффициентов системы (4), вычисленная в точке $x = q^1$; $F^{-1}(x)$ — символьная 3×3 матрица.

Выполняемые действия:

1) вычисляется матрица $F^{-1}(q^0)$ с помощью встроенной функции eval(); затем вычисляются привилегированные координаты z^0 точки q^0 (см. формулу п. 3, (b) алгоритма раздела 3);

2) с помощью формулы (5) вычисляются координаты y^0 точки z^0 в системе координат (y_1, y_2, y_3) системы (6).

Выходные данные: вектор y^0 длины 3.

Встроенные процедуры пакета Maple DEtools. Приведенные ниже две процедуры вызываются из программы FindControlLoc.

Процедура dsolve();

Входные данные: $X_1,\ X_2 \in \mathrm{Vec}(\mathbb{R}^3)$, $u(t)$, $q^0 \in \mathbb{R}^3$;

Выполняемые действия: численно решается задача Коши для системы $\dot{x} = u_1(t)X_1(x) + u_2(t)X_2(x)$ с начальным условием $x(0) = q^0$ с помощью метода Рунге-Кутта 4-5 порядков.

Выходные данные: traj — таблица чисел, задающая значения вычисленной траектории.

Процедура odeplot() ;

Входные данные: traj — таблица чисел, T;

Выполняемые действия: Выводит трехмерное графическое изображение, плоские проекции и графики компонент траектории traj.

Выходные данные: графическое изображение траектории, являющейся решением задачи Коши.

Библиотека NilpControls.
Все процедуры библиотеки NilpControls вычисляют управления, являющиеся решениями задачи (6), (7) ([8, 9]).

Далее везде $r_0 = \sqrt{(y_1^0)^2 + (y_2^0)^2}$ — полярный радиус, φ_0 — полярный угол точки (y_1^0, y_2^0).

Процедура Optimal();

Вычисляются программные управления, оптимальные в смысле минимума функционала субримановой длины $L = \int_0^T \sqrt{u_1^2 + u_2^2}\, dt \to \min$ ([8]).

Входные данные: $y^0 \in \mathbb{R}^3$, T;

Выполняемые действия:

1) Если $y_3^0 \neq 0$, то вычисляются функции

$$u_1(t) = -d/T \cos(\psi - ct), \qquad u_2(t) = -d/T \sin(\psi - ct),$$

где $d = r_0\bar{t}/(2|\sin(\bar{t}/2)|)$, $\psi = \varphi_0 + (\operatorname{sign} y_3^0)\bar{t}/2$, $c = (\operatorname{sign} y_3^0)\bar{t}/T$, $\bar{t} = \bar{t}(y^0) \in (0, 2\pi)$ — численное решение уравнения $\dfrac{\bar{t} - \sin\bar{t}}{\sin^2(\bar{t}/2)} = \dfrac{8|y_3^0|}{r_0^2}$, найденное с помощью встроенной функции fsolve().

2) Если $r_0 \neq 0$, $y_3^0 = 0$, то $u_1(t) = -y_1^0/T$, $u_2(t) = -y_2^0/T$.

Выходные данные: $u(t)$, $t \in [0, T]$.

Процедура Trig();

Вычисляются программные управления в классе тригонометрических функций ([8]).

Входные данные: $y^0 \in \mathbb{R}^3$, T, параметр $\gamma \in \mathbb{R}$, $\gamma \neq -2y_2^0/T$;

Выполняемые действия: вычисляются функции

$u_1 = -y_1^0/T + \beta(\gamma)\sin(2\pi t/T)$, $\quad u_2 = -y_2^0/T + \gamma\cos(2\pi t/T)$,

где $\beta(\gamma) = 4\pi y_3^0/(T(2y_2^0 + \gamma T))$ (см. [8]).

Выходные данные: $u(t, \gamma)$, $t \in [0, T]$.

Процедура PieceConst();

Вычисляются кусочно-постоянные с одним переключением программные управления ([8]).

Входные данные: $y^0 \in \mathbb{R}^3$: $r_0 \neq 0$; T, параметр $\nu \in \mathbb{R}$, $\nu \neq \varphi_0$;

Выполняемые действия:

1) Если $y_3^0 \neq 0$, то вычисляются функции

$$u_1 = \begin{cases} \mu(\nu)\cos(\nu), & t \in [0, T/2), \\ -\mu(\nu)\cos(\nu) - \eta_1, & t \in [T/2, T], \end{cases}$$

$$u_2 = \begin{cases} \mu(\nu)\sin(\nu), & t \in [0, T/2), \\ -\mu(\nu)\sin(\nu) - \eta_1, & t \in [T/2, T], \end{cases}$$

где $\mu(\nu) = 4y_3^0/(T(y_2^0\cos(\nu) - y_1^0\sin(\nu)))$, $\eta_1 = 2y_1^0/T$, $\eta_2 = 2y_2^0/T$.

2) Если $y_3^0 = 0$, то $u_1(t) = -y_1^0/T$, $u_2(t) = -y_2^0/T$.

Выходные данные: 1) $u(t, \nu)$, $t \in [0, T]$; 2) $u(t)$, $t \in [0, T]$.

Процедура Centre();

Вычисляются программные управления, полученные с помощью линейного поля $v = (v_1, v_2) = (ay_1 + by_2, \quad cy_1 - ay_2)$, имеющего особенность в точке $(0, 0)$ типа центр. Управления этого класса имеют одно переключение, постоянны на второй половине временного отрезка ([9]).

Входные данные: $y^0 \in \mathbb{R}^3$: $r_0 \neq 0$; T, параметры $a, b, c \in \mathbb{R}$, удовлетворяющие условиям: $a^2 + b^2 + c^2 \neq 0$, $bc < 0$, $\Delta = \begin{vmatrix} a & b \\ c & -a \end{vmatrix} > 0$;

Выполняемые действия:

1) Если $y_3^0 \neq 0$, то вычисляется проекция траектории системы (6), выходящая из точки (y_1^0, y_2^0):

$$y_1(t) = y_1^0\cos(t\sqrt{\Delta}) + \frac{v_1^0}{\sqrt{\Delta}}\sin(t\sqrt{\Delta}),$$

$$y_2(t) = y_2^0\cos(t\sqrt{\Delta}) + \frac{v_2^0}{\sqrt{\Delta}}\sin(t\sqrt{\Delta}),$$

где $v_1^0 = ay_1^0 + by_2^0$, $v_2^0 = cy_1^0 - ay_2^0$;
вычисляются функции:

$$u_1(t) = \begin{cases} k\,\mathrm{sign}(by_3^0)(ay_1(kt) + by_2(kt)), & t \in [0, T/2], \\ -2p_1/T, & t \in [T/2, T], \end{cases}$$

$$u_2(t) = \begin{cases} k\,\mathrm{sign}(by_3^0)(cy_1(kt) - ay_2(kt)), & t \in [0, T/2], \\ -2p_2/T, & t \in [T/2, T], \end{cases}$$

где точка переключения $p = (y_1(\mathrm{sign}(by_3^0)t_p), y_2(\mathrm{sign}(by_3^0)t_p), 0)$, $t_p = \frac{2|y_3^0|}{|\delta_v^0|}$, $\delta_v^0 = -b(y_2^0)^2 + c(y_1^0)^2 - 2ay_1^0y_2^0$, $k = 2tp/T$.

2) Если $y_3^0 = 0$, то $u_1(t) = -y_1^0/T$, $u_2 = -y_2^0/T$.

Выходные данные: 1) $u(t, a, b, c)$, $t \in [0, T]$; 2) $u(t)$, $t \in [0, T]$.

Процедура Focus();

Вычисляются программные управления, полученные с помощью линейного поля $v = (v_1, v_2) = (ay_1 + by_2, \quad cy_1 + dy_2)$, имеющего особенность в точке $(0,0)$ типа фокус (при $a + d > 0$ — неустойчивый). Управления этого класса имеют одно переключение, постоянны на второй половине временного отрезка ([9]).

Входные данные: $y^0 \in \mathbb{R}^3$: $r_0 \neq 0$; T, параметры $a, b, c, d \in \mathbb{R}$, удовлетворяющие условиям: $a^2 + b^2 + c^2 + d^2 \neq 0$, $\quad bc < 0$, $\quad S = a + d > 0$, $\quad \Delta = \begin{vmatrix} a & b \\ c & d \end{vmatrix} > S^2/4$;

Выполняемые действия:

1)Если $y_3^0 \neq 0$, то вычисляется проекция траектории системы (6), выходящая из точки (y_1^0, y_2^0):

$$y_1(t) = e^{St/2}(y_1^0 \cos(t\sqrt{|D|}/2) + \frac{(a-d)y_1^0 + \operatorname{sign}(by_3^0)2by_2^0}{\sqrt{|D|}} \sin(t\sqrt{|D|}/2),$$

$$y_2(t) = e^{St/2}(y_2^0 \cos(t\sqrt{|D|}/2) - \frac{(a-d)y_2^0 + \operatorname{sign}(by_3^0)2cy_1^0}{\sqrt{|D|}} \sin(t\sqrt{|D|}/2),$$

где $D = S^2 - 4\Delta$;
вычисляются функции:

$$u_1(t) = \begin{cases} k(ay_1(kt) + \operatorname{sign}(y_3^0)|b|y_2(kt)), & t \in [0, T/2], \\ -2p_1/T, & t \in [T/2, T], \end{cases}$$

$$u_2(t) = \begin{cases} k(-\operatorname{sign}(y_3^0)|c|y_1(kt) + dy_2(kt)), & t \in [0, T/2], \\ -2p_2/T, & t \in [T/2, T], \end{cases}$$

где точка переключения $p = (y_1(tp), y_2(tp), 0)$, $t_p = 1/S \ln(1 + 2Sy_3^0/\delta^0)$, $\delta^0 = \operatorname{sign}(by_3^0)(b(y_2^0)^2 - c(y_1^0)^2) + (a-d)y_1^0 y_2^0$, $k = 2t_p/T$.

2) Если $y_3^0 = 0$, то $u_1(t) = -y_1^0/T$, $u_2 = -y_2^0/T$.

Выходные данные: 1) $u(t, a, b, c, d)$, $t \in [0, T]$; 2) $u(t)$, $t \in [0, T]$.

Проанализируем описанную библиотеку. Управления Optimal и Trig решают задачу управления (6), (7) во всем пространстве состояний $\mathbb{R}^3$, а управления PieceConst, Centre, Focus, в отличие от первых двух, только в $\mathbb{R}^3 \setminus \{y_1^2 + y_2^2 = 0\}$, что означает, что они неприменимы в случае, если $y^0 \in \{y_1^2 + y_2^2 = 0\}$, и требуется искать другую стратегию перемещения. Все управления библиотеки, за исключением Trig, порождают управления с обратной связью, что обусловливает их устойчивость к небольшим погрешностям начального положения.

4.2. Схема программы FindControlLoc

Схема основной программы FindControlLoc отражает схему описанного выше алгоритма приближенного решения задачи управления (1)–(3).

Программа FindControlLoc.

Входные данные: $X_1,\ X_2 \in \mathrm{Vec}(\mathbb{R}^3)$, $x^0,\ x^1 \in \mathbb{R}^3$, $T > 0$, $\varepsilon > 0$, nc, par(nc);

Выполняемые действия:

(1) Инициализируются пакеты системы Maple linalg, DEtools, plottools, plots.

(2) Инициализируются процедуры NilpApprox, ChangeCoords, библиотека NilpControls.

(3) Считываются данные: X_1, X_2, x^1.

(4) Вычисляется матрица $C = \mathrm{NilpApprox}(X_1, X_2, x^1)$.

(5) Считываются данные x^0, T, ε, выбирается процедура-управление nc из библиотеки NilpControls, считываются параметры для этой процедуры par(nc).

(6) Входное начальное состояние x^0 запоминается в переменной q^0, счетчику итераций присваивается значение $i := 0$.

(7) Итерационный процесс реализуется с помощью условного цикла While(Dist(q^0, x^1)$\geqslant \varepsilon$), где Dist(q^0, x^1) — евклидово расстояние между точками q^0, x^1.

i- я итерация: пусть q^0 — приближение к x^1 на $(i-1)$-ой итерации (считаем $q^0 = x^0$ приближением к x^1 на нулевой итерации);

(a) счетчик итераций увеличивается на единицу: $i := i + 1$;

(b) вычисляются координаты начального состояния системы (6): $y^0 = \mathrm{ChangeCoords}(q^0, x^1, C)$;

(c) с помощью выбранной процедуры nc вычисляются управления и запоминаются в массиве переменной длины $u[i] = \mathrm{NilpControls(nc)}(y^0, T, \mathrm{par(nc)})$;

(d) численно решается задача Коши: traj = dsolve(X_1, X_2, $u[i](t)$, q^0);

(e) вычисляется следующее приближение к финальной точке x^1: $q^0 = \mathrm{traj(T)}$.

(8) Если Dist(q^0, x^1)$< \varepsilon$, то в переменную N запоминается количество итераций; если $N = 0$, то программа останавливается, если $N > 0$, то переходит к следующему пункту.

(9) Управления $u[i]$, $i = 1, ..., N$, перепараметризуются по правилу (9), умножаются на функции $\delta[i]$ с соответствующими номерами i, где

$$\delta[i] = \begin{cases} 0, & t \in [0, (i-1)T/N], \\ 1, & t \in [(i-1)T/N, iT/N], \\ 0, & t \in [iT/N, T], \end{cases}$$

и полученные функции, определенные на временном отрезке $[0, T]$, суммируются. Получаются искомые управления $u(t)$, соответствующие формулам (9).

(10) Полученный результат проверяется прямой подстановкой управления $u(t)$ в исходную систему (1) и численным решением задачи Коши: traj = dsolve(X_1, X_2, $u(t)$, x^0); вычислением состояния, в которое приходит система: q^1 = traj(T); вычислением Dist(q^1, x^1).

(11) *Вывод результата.*
   (a) Если Dist(q^1, x^1)$< \varepsilon$, то управления $u(t)$ выводятся аналитически и графически. Для визуализации результата с помощью функции odeplot выводится трехмерное изображение траектории движения исходной системы; для анализа результата выводятся ее двумерные проекции и графики компонент.
   (b) Если Dist(q^1, x^1)$\geqslant \varepsilon$, то выводится сообщение об ошибке.

Выходные данные: либо (a) графическое и аналитическое представление управления $u(t)$, $t \in [0, T]$, либо (b) — сообщение об ошибке.

Проанализируем возможные причины ошибки:

(1) недостаточная точность в итерационном процессе;
(2) наличие фазовых ограничений: траектория системы может выходить за пределы области;
(3) задача не локальна для выбранного класса управлений.

5. Управление ориентацией катящейся по плоскости сферы

5.1. Апробация программы FindControlLoc

Управляемая система, описывающая качение сферы по плоскости без прокручивания и проскальзывания, описана в книгах [6, 7].

Рассмотрим подсистему этой системы, описывающую изменение ориентации сферы. Переходя в этой подсистеме от ортогональных 3×3 матриц к кватернионам (см. [12]) и применяя проекцию на трехмерное пространство, получим следующую систему:

(10)
$$\dot{x_1} = -x_3 u_1 + x_2 u_2,$$
$$\dot{x_2} = -\sqrt{1 - x_1^2 - x_2^2 - x_3^2}\, u_1 - x_1 u_2,$$
$$\dot{x_3} = x_1 u_1 - \sqrt{1 - x_1^2 - x_2^2 - x_3^2}\, u_2,$$
$$q = (x_1, x_2, x_3) \in B^3 = \left\{x_1^2 + x_2^2 + x_3^2 < 1\right\}, \qquad u = (u_1, u_2) \in \mathbb{R}^2.$$

С помощью компьютерной программы FindControlLoc задача управления (1)–(3) для системы (10) решена в пяти классах управлений библиотеки NilpControls.

Пример работы программы FindControlLoc.

Входные данные: векторные поля системы (10)

$$X_1 = \left(-x_3,\ -\sqrt{1 - x_1^2 - x_2^2 - x_3^2}, x_1\right)^T,$$
$$X_2 = \left(x_2,\ -x_1,\ -\sqrt{1 - x_1^2 - x_2^2 - x_3^2}\right)^T;$$

граничные условия $x^0 = [.3555263893, .2583050417, .1359392951]$, $x^1 = [.1381591491, 0., .05841275134]$; время $T = 1.$, точность $\varepsilon = 10^{-6}$, nc $\in$ NilpControls, par(nc).

Выполняемые действия:
Dist(x^0, x^1)= .3463818366;
Коэффициенты нильпотентной аппроксимации (4) в точке x^1:
$$C = \begin{pmatrix} -.06987008492 & -0.5 \\ 0.5 & -.0698700849 \end{pmatrix}$$

Процедура nc	*Параметры par(nc)*	*Число итераций*
Optimal	—	$N = 6$
Trig	1.	$N = 5$
PieceConst	1.8849555922	$N = 5$
Centre	$(1, 4, -1)$	$N = 8$
Centre	$(0, 5, -3)$	$N = 4$
Focus	$(0, 5, -2, 0.3)$	$N = 3$

Выходные данные:
1) $u^{Opt}(t)$, $t \in [0,1]$, рис. 1, 2;
2) $u^{Trg}(t)$, $t \in [0,1]$, рис. 3, 4;
3) $u^{PC}(t)$, $t \in [0,1]$, рис. 5, 6;
4) $u^{Centre}(t)$, $t \in [0,1]$, рис. 7, 8;
5) $u^{Focus}(t)$, $t \in [0,1]$, рис. 9, 10.

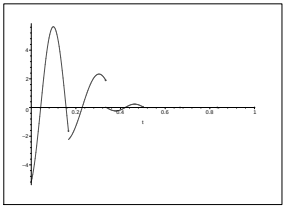

Рис. 1. $u_1^{Opt}(t)$

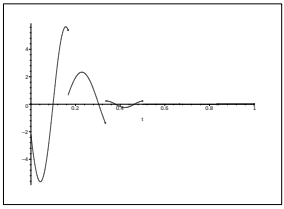

Рис. 2. $u_2^{Opt}(t)$

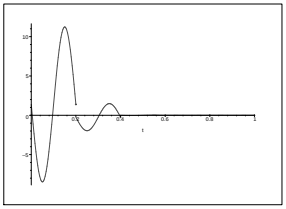

Рис. 3. $u_1^{Trg}(t)$

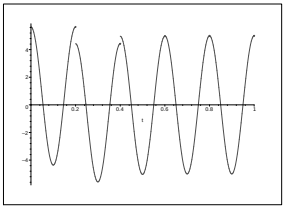

Рис. 4. $u_2^{Trg}(t)$

5.2. Анализ работы программы FindControlLoc

Дадим некоторые рекомендации по использованию библиотеки NilpControls на основании полученных практических наблюдений и анализа графиков управлений, вычисленных с помощью программы FindControlLoc.

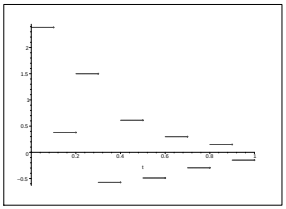

Рис. 5. $u_1^{PC}(t)$

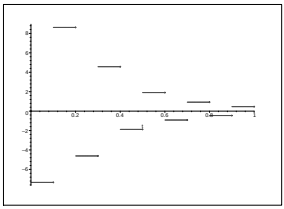

Рис. 6. $u_2^{PC}(t)$

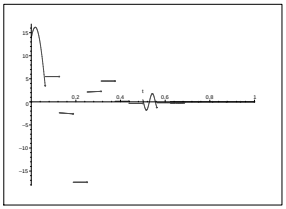

Рис. 7. $u_1^{Centre}(t)$

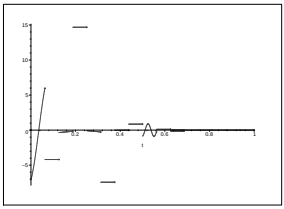

Рис. 8. $u_2^{Centre}(t)$

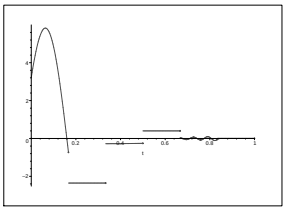

Рис. 9. $u_1^{Focus}(t)$

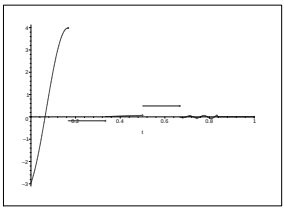

Рис. 10. $u_2^{Focus}(t)$

Если начальное расстояние $\mathrm{Dist}(x^0, x^1)$ мало в смысле константы δ (см. п. 3) и нет ограничений в пространстве состояний, то рекомендуется $\mathcal{U} = \mathrm{Optimal}$ (так как управления этого класса не зависят от параметров, «жесткие»). Из анализа графиков управлений (рис. 1, 2)

видно, что система локализуется в окрестности целевой точки за время $t \approx T/2$.

Если начальное расстояние $\mathrm{Dist}(x^0, x^1)$ достаточно велико и требуется большая скорость перемещения системы, как в задачах управления спутником (требуются неограниченные управления), то рекомендуется $\mathcal{U} = \mathrm{Trig}$. Тригонометрические управления зависят от параметра γ, настройка которого влияет на характер движения системы, например, в разобранном выше случае краевых условий при $\varepsilon = 10^{-3}$ и $\gamma = 5$. количество итераций $N = 9$, максимальная амплитуда управления $u_2(t)$ равна 90, скорость вхождения в ε-окрестность точки x^1 равна 40, в сравнении, при $\varepsilon = 10^{-6}$ и $\gamma = 1$. количество итераций $N = 5$, максимальная амплитуда управления $u_2(t)$ равна 20, скорость вхождения в ε-окрестность точки x^1 равна 5 (см. рис. 3, 4), тогда как во всех остальных случаях, кроме кусочно-постоянного, скорость равна нулю. Заметим, что при $\gamma = 0.5$ и $\varepsilon = 10^{-3}$ траектория системы выходит из области $B^3 = \left\{x_1^2 + x_2^2 + x_3^2 < 1\right\}$, что является одной из причин ошибок программы. В случае $\mathcal{U}=$ PieceConst движение происходит также с большой скоростью, но, несмотря на простоту управлений, характер движения сложен: из-за больших амплитуд и частой смены знака функций $u_1(t)$, $u_2(t)$ (рис. 5, 6) траектория системы игольчатая, поэтому этот тип управлений рекомендуется применять только в специальных случаях локальных задач управления.

Достаточно гибкими показали себя управления синтетического типа — управления классов $\mathcal{U}=$ Centre, Focus. В отличие от оптимальных, которые выражаются через тригонометрические функции и неэлементарную функцию, фокусные и центральные управления проще: они имеют один разрыв при $t = T/2$, на отрезке $[0, T/2]$ выражаются через тригонометрические, экспоненциальные функции, на отрезке $[T/2, T]$ управления постоянны. Амплитуды центрального и фокусного управлений монотонно убывают к нулю и в обоих этих случаях при $t < T$ система практически с нулевой скоростью, как и в оптимальном случае, входит в ε-окрестность точки x^1 (см. рис. 7, 10). Кроме того, управления этих типов зависят от нескольких параметров, что делает их гибкими. На рис. 7, 8 показаны управления класса Centre, вычисленные при двух различных наборах параметров. Управления классов $\mathcal{U}=$ Centre, Focus рекомендуется широко применять при умеренных начальных расстояниях $\mathrm{Dist}(x^0, x^1)$.

6. Заключение

В статье приводится компьютерная реализация вычислительного алгоритма и анализируется его работа. Программа FindControlLoc осуществляет пять стратегий управления системой в малой окрестности целевой точки. Чтобы эффективно решать глобальные задачи управления, потребуется строить эффективные смешанные стратегии, для чего потребуются методы параллельного программирования.

Список литературы

[1] Сачкова Е.Ф. *Приближенное решение задачи управления на основе нильпотентной аппроксимации* // Дифференциальные уравнения, 2009, № 10. ↑1, 2, 3, 2, 4.1

[2] Дьяконов В. Maple 6: учебный курс. — СПб.: Питер, 2001. — 608 с. ↑1

[3] Laumond J. P. // Lecture Notes in Control and Information Science, № 229: Springer, 1998. — 343 с. ↑1

[4] Гурман В. И. Принцип расширения в задачах оптимального управления.-2-ое изд., перераб. и доп. — М.: Наука, 1997. — 288 с. ↑1

[5] Дыхта В. А., Самсонюк О. Н. Оптимальное импульсное управление с приложениями. — М.: Физматлит, 2000. — 256 с. ↑1

[6] Аграчев А. А., Сачков Ю. Л. Геометрическая теория управления. — М.: Физматлит, 2005. — 392 с. ↑1, 2, 5.1

[7] Jurdjevic V. Geometric control theory. — Cambridge: University Press, 1997. ↑1, 5.1

[8] Сачкова Е.Ф. *Решение задачи управления для нильпотентной системы* // Дифференциальные уравнения, № 12, 2008, с. 1704–1707. ↑1, 2, 2, 3d, 4.1

[9] Сачкова Е.Ф. *Синтез управления для трехмерной нильпотентной системы* // Дифференциальные уравнения, принята к публикации. ↑1, 2, 2, 3d, 4.1

[10] Bellaiche A. *The tangent space in sub-Riemannian geometry.* // Sub-Riemannian Geometry, A. Bellaiche and J. J. Risler, Eds. — Basel, Swizerland: Birkhäuser, 1996, с. 1–78. ↑2

[11] Jean F. *Lectures on Dynamical and Contol Systems.* — Trieste, 2003. ↑3

[12] Уиттекер Э.Е. Аналитическая динамика. — М.: УРСС, 2004. ↑5.1

E. F. Sachkova. *Realization and analysis of algorithms for approximate solving the control problem* // Proceedings of Program Systems institute scientific conference "Program systems: Theory and applications". — Pereslavl-Zalesskij, v. **1**, 2009. — p. 59–76. — ISBN 978-5-901795-16-3 (*in Russian*).

Abstract. We consider a computer realization of an algorithm for approximate solving the control problem for three-dimensional nonlinear systems governed by ordinary differential equations, linear in two controls. The algorithm is based on the method of nilpotent approximation. It is realized in Maple system and tested on the problem of orientation control of a sphere rolling on a plane.