Санкт-Петербургский государственный университет

Цзун Цяньвэнь

Выпускная квалификационная работа

Исследование метода ортогональных коллокаций для решения задач оптимального управления

Уровень образования: магистратура

Направление 02.04.02 «Фундаментальная информатика и информационные технологии»

Основная образовательная программа

ВМ.5786.2020 «Цифровые технологии и системы»

Научный руководитель:

доктор физ.-мат. наук, профессор кафедры компьютерных технологий и систем

Сотникова Маргарита Викторовна

Санкт-Петербург

2022

Содержание

1 Введение

В основе метода решения задач оптимального управления лежат следующие три основных компонента: методы решения дифференциальных уравнений и интегрирования функций; метод решения системы нелинейных алгебраических уравнений; и метод решения задачи нелинейной оптимизации. Существует два подхода к численному решению задач оптимального управления: прямой и непрямой.

Методы решения дифференциальных уравнений и интегрирования функций необходимы для всех численных методов оптимального управления. В непрямом методе численное решение дифференциальных уравнений сочетается с численным решением систем нелинейных уравнений, а в прямом методе численное решение дифференциальных уравнений сочетается с нелинейной оптимизацией.

Ортогональная коллокация – это прямой метод численного решения дифференциальных уравнений. Он использует коллокацию в нулях некоторых ортогональных полиномов для преобразования дифференциального уравнения в набор обыкновенных дифференциальных уравнений. Затем эти ОДУ могут быть решены любым методом. Было показано, что обычно выгодно выбирать точки коллокации как нули соответствующего полинома Якоби. Метод особенно подходит для решения нелинейных задач и имеет преимущества высокой вычислительной точности и устойчивости по сравнению с традиционными разностными методами.

2 Постановка задачи

Целью данной работы является исследование метода ортогональных коллокаций для решения задач оптимального управления.

Для достижения поставленной цели были решены следующие задачи:

1. Построены интерполяционные полиномы Лагранжа. Исследован феномен Рунге. Реализована программа для вычисления полиномов Лагранжа в среде Matlab.

2. Проведено сравнение точности интерполяции полиномами Лагранжа на равномерной и неравномерной сетках. Показано, что неравномерная сетка позволяет избежать появления эффекта Рунге.

3. Рассмотрен метод приближенного численного решения дифференциальных уравнений с использований интерполяционных полиномов Лагранжа.

4. Приведен пример решения дифференциальных уравнений.

3 Обзор литературы

Существует два подхода к численному решению задач оптимального управления: прямой и косвенный. Типичным примером косвенного подхода является принцип экстремальных значений Понтрягина [1], а типичным примером прямого подхода является метод ортогональных конфигураций. По сравнению с косвенным методом, прямой метод не требует выведения необходимых условий первого порядка и имеет преимущества малой чувствительности к начальному значению и большого радиуса сходимости [2, 3].Elnagar и другие [4] впервые применили преобразование псевдоспектрального метода теории оптимального управления динамическими системами (OCP) и продемонстрировал, что этот метод может эффективно преобразовывать задачи оптимального управления в задачи нелинейного программирования.

Теория ортогональной коллокации в этой работе основана на использовании интерполяции Лагранжа на неравномерной сетке. Некоторые математические факты, использованные в работе, представлены в [5] и [6]. В статье [5] описана барицентрическая интерполяция, которая является вариантом полиномиальной интерполяции Лагранжа. В книге [6] описаны свойства для различных ортогональных полиномов, используемых для нахождения точек интерполяции узлов. Эффект Рунге, возникающий при интерполяции на равномерной сетке, можно избежать с помощью интерполяции на неравномерной сетке, которая впервые описана в [7]. Обзор псевдоспектральных методов, основанных на коллокации в точках Лежандра-Гаусса, Лежандра-Гаусса-Радо и Лежандра-Гаусса-Лобатто, приведен в [8].

4 Интерполяционный полином Лагранжа на равномерной и неравномерной сетках

4.1 Интерполяционный полином Лагранжа на равномерной сетке. Эффект Рунге

Пусть задана $n+1$ пара чисел $(x_0, y_0), \ldots, (x_n, y_n)$, где все x_j различны и расстояния между всеми точками x_j и x_{j+1} одинаково, то есть сетка равномерная. Требуется построить полином $L(x)$ степени не более n, для которого $L(x_j) = y_j$.

Лагранж предложил следующий способ вычисления таких полиномов:

$$L(x) = \sum_{i=0}^{n} y_i \ell_i(x), \tag{1}$$

где базисные полиномы l_i определяются по формуле

$$\ell_i(x) = \prod_{j=0, j\neq i}^{n} \frac{x - x_j}{x_i - x_j} = \frac{(x - x_0)\cdots(x - x_{i-1})(x - x_{i+1})\cdots(x - x_n)}{(x_i - x_0)\cdots(x_i - x_{i-1})(x_i - x_{i+1})\cdots(x_i - x_n)}. \tag{2}$$

Для любого $i = 0, \ldots, n$ многочлен l_i имеет степень n и

$$\ell_i(x_j) = \begin{cases} 0 & , j \neq i, \\ 1 & , j = i. \end{cases} \tag{3}$$

Отсюда следует, что $L(x)$, являющийся линейной комбинацией многочленов $l_i(x)$, имеет степень не больше n и $L(x_i) = y_i$.

В приложении приведена программа для вычисления полинома Лагранжа, реализованная в Matlab.

Пример. Мы хотим интерполировать функцию $f(x) = x^2$ на интервале

$$1 \leq x \leq 3, \tag{4}$$

по трем точкам: $f(x_1 = 1) = 1, f(x_2 = 2) = 4, f(x_3 = 3) = 9$.

В соответствии с формулами (1) и (2), интерполирующий полином равен:

$$L(x) = 1 \cdot \frac{x-2}{1-2} \cdot \frac{x-3}{1-3} + 4 \cdot \frac{x-1}{2-1} \cdot \frac{x-3}{2-3} + 9 \cdot \frac{x-1}{3-1} \cdot \frac{x-2}{3-2} = x^2 \tag{5}$$

Рассмотрим функцию

$$f_1(x) = \frac{1}{1 + 25x^2}. \tag{6}$$

На рис. 1 приведены полиномы Лагранжа, построенные для этой функции на интервале $[-1, 1]$ с равномерной сеткой, при различном числе пар точек N.

Аналогично, на рис. 2 построены полиномы Лагранжа для функции

$$f_2(x) = \operatorname{sech}(5x) \tag{7}$$

на интервале $[-1, 1]$ с равномерной сеткой.

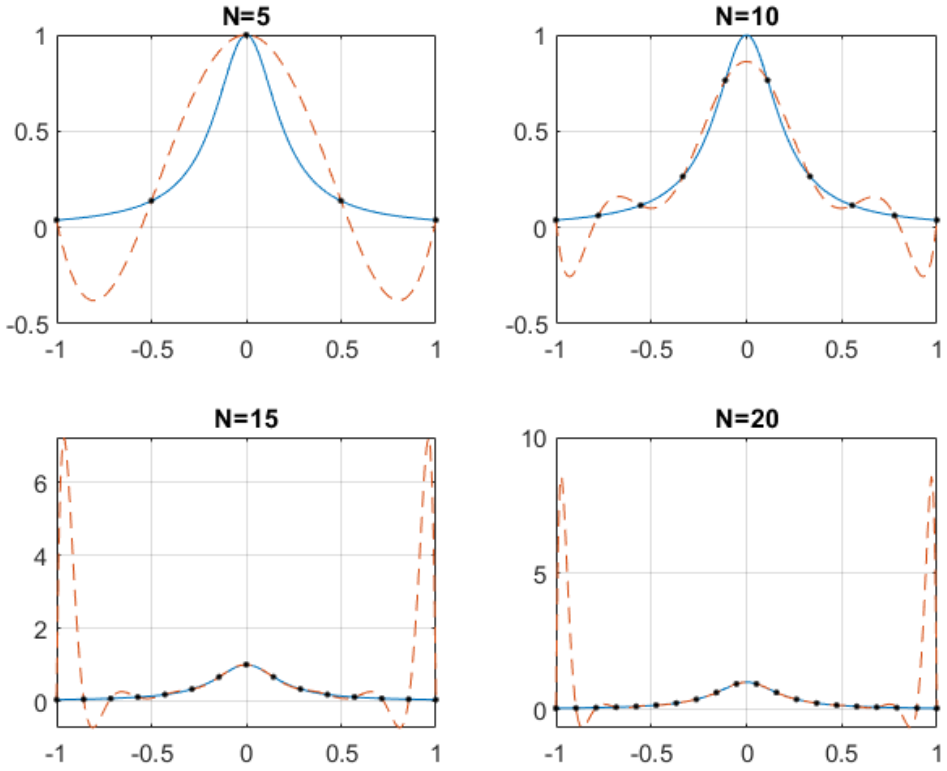


Рисунок 1 – Полиномы Лагранжа для функции (6).

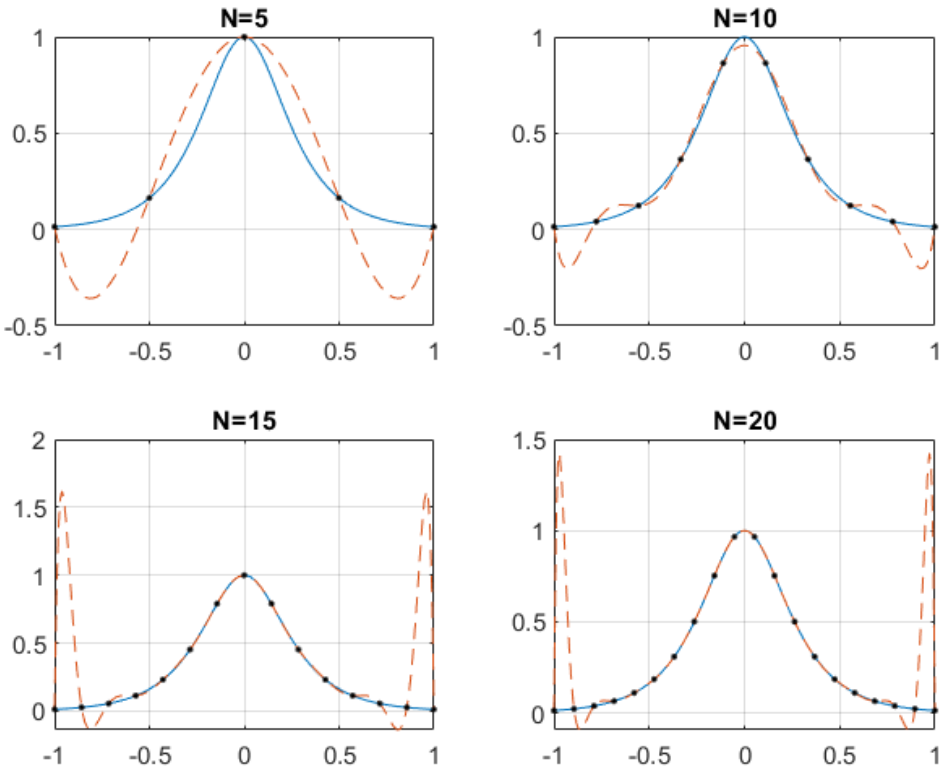


Рисунок 2 – Полиномы Лагранжа для функции (7).

Из рисунков 1 и 2 видно, что по мере увеличения числа узловых точек N точность аппроксимации в центре интервала будет увеличиваться, но на краях интервала появятся большие отклонения от исходной функции, что называется феноменом Рунге.

Теорема о приближении Вейерштрасса гласит, что для каждой непрерывной функции $f(x)$, определенной на интервале $[a, b]$, существует множество полиномиальных функций $P_n(x)$ для $n = 0, 1, 2, ...$, каждая степени не более n, которые приближают $f(x)$ с равномерной сходимостью на $[a, b]$ по мере приближения n к бесконечности, то есть,

$$\lim_{n\to\infty} (\max_{a\leq x\leq b} |f(x) - P_n(x)|) = 0. \tag{8}$$

Феномен Рунге в численном анализе — эффект нежелательных осцилляций, возникающий при интерполяции полиномами высоких степеней. Был открыт Карлом Рунге при изучении ошибок полиномиальной интерполяции для

приближения некоторых функций.

Возникновение этого эффекта зависит от выбора узлов интерполяции. Теоретически, согласно теореме Вейерштрасса, существует такая последовательность узлов, для которой последовательность интерполяционных полиномов равномерно сходится к заданной непрерывной функции на отрезке.

В связи с этим для борьбы с эффектом Рунге будем использовать неравномерную сетку.

4.2 Интерполяционный полином Лагранжа на неравномерной сетке.

Для борьбы с эффектом Рунге используется неравномерная сетка. В данной работе используется неравномерная сетка Гаусса-Лежандра. Существует три типа таких сеток: Гаусса-Лежандра, Гаусса-Лежандра-Лобато и Гаусса-Лежандра-Радао.

В листингах 3, 4 и 5 приведены программы для вычисления узлов на неравномерной сетке методами Гаусса-Лежандра, Гаусса-Лежандра-Лобато и Гаусса-Лежандра-Радао соответственно.

Рассмотрим примеры построения интерполяционного полинома Лагранжа на неравномерной сетке для функции $f_1(x)$ (6). На рисунках 3, 4 и 5 показаны результаты интерполяции указанной функции полиномами Лагранжа на неравномерной сетке, построенной методами Гаусса-Лежандра, Гаусса-Лежандра-Лобато и Гаусса-Лежандра-Радао соответственно. В каждом случае рассматривается количество узлов $N = 5$, $N = 10$, $N = 15$ и $N = 20$.

Аналогично, на рисунках 6, 7 и 8 показаны результаты интерполяции функции $f_2(x)$ (7) полиномами Лагранжа на неравномерной сетке, построенной методами Гаусса-Лежандра, Гаусса-Лежандра-Лобато и Гаусса-Лежандра-Радао соответственно для количества узлов $N = 5$, $N = 10$, $N = 15$ и $N = 20$.

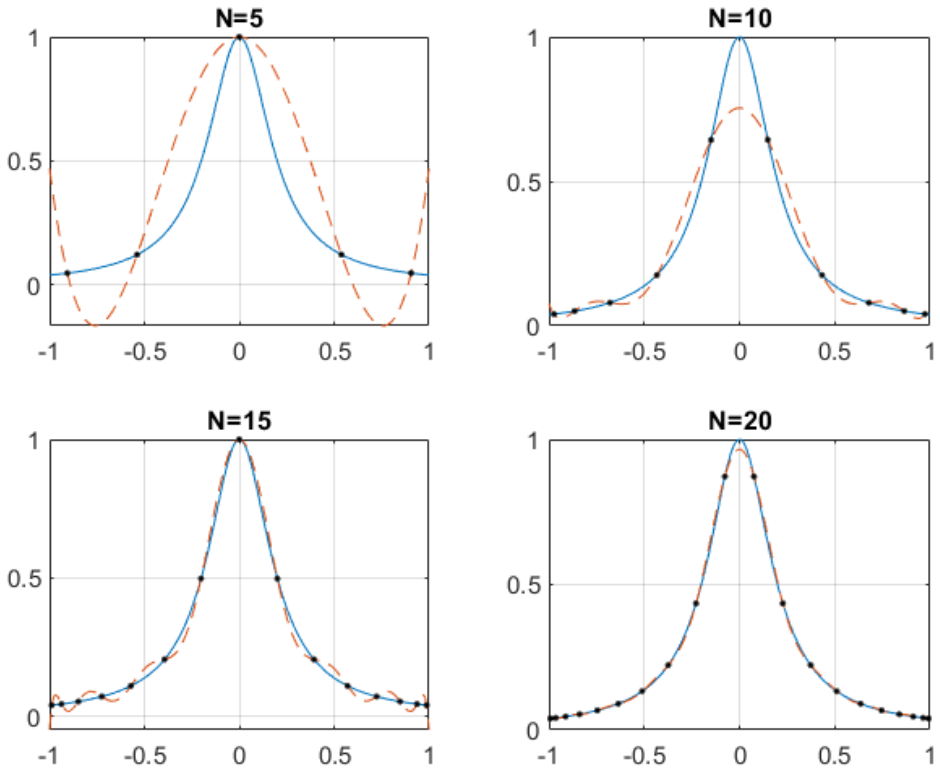


Рисунок 3 – Полиномы Лагранжа для функции $f_1(x)$ (6) для сетки Гаусса-Лежандра.

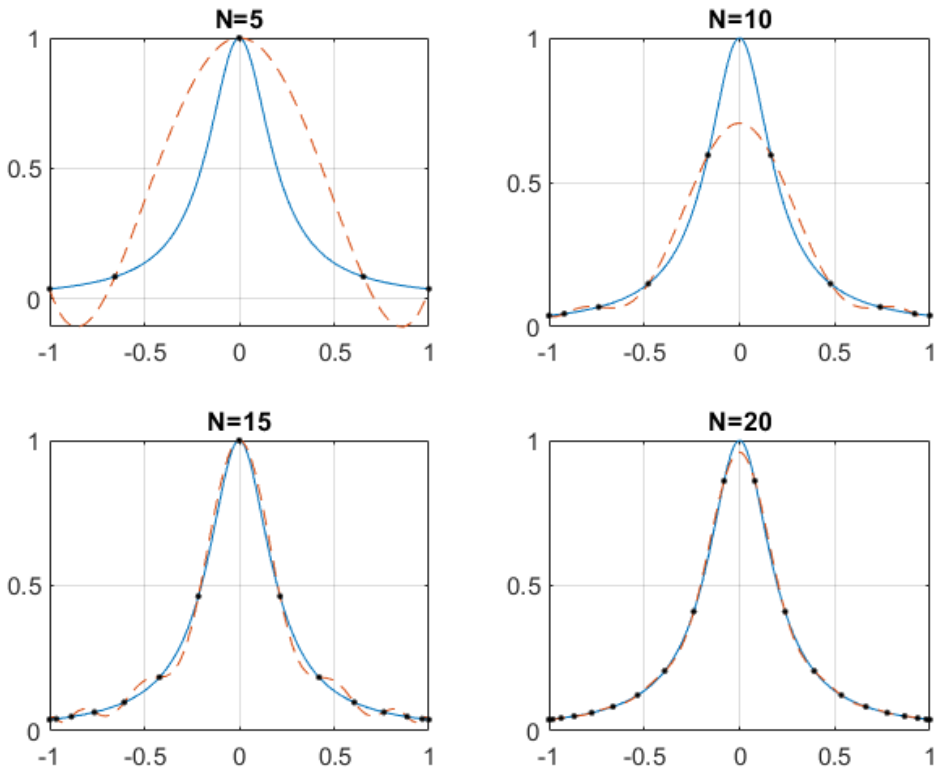


Рисунок 4 – Полиномы Лагранжа для функции $f_1(x)$ (6) для сетки Гаусса-Лежандра-Лобато.

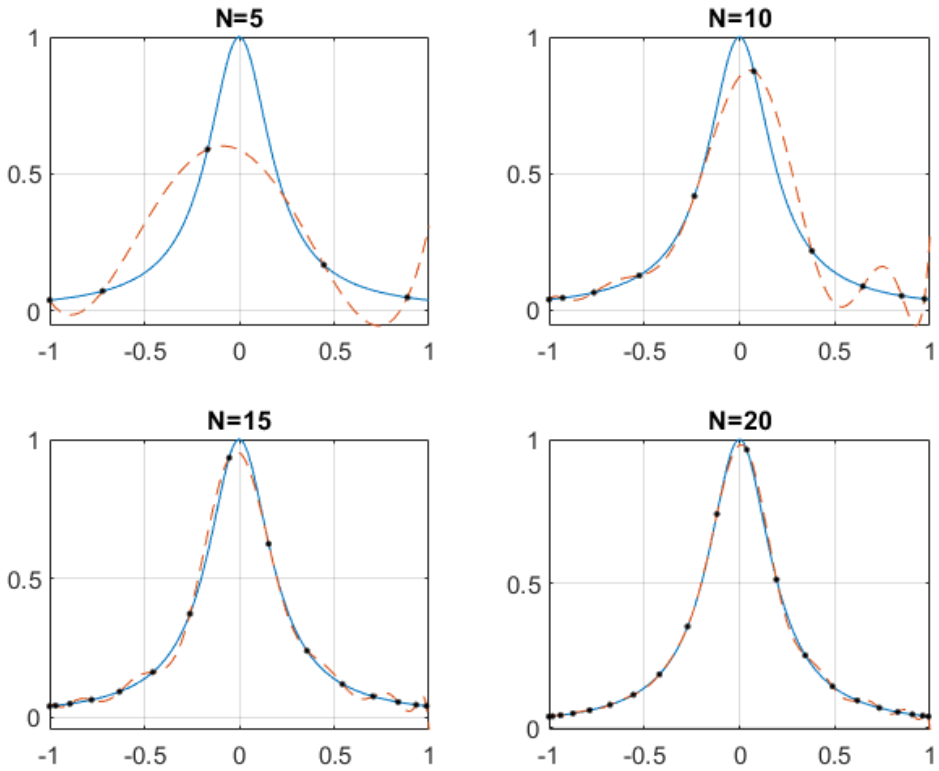


Рисунок 5 – Полиномы Лагранжа для функции $f_1(x)$ (6) для сетки Гаусса-Лежандра-Радао.

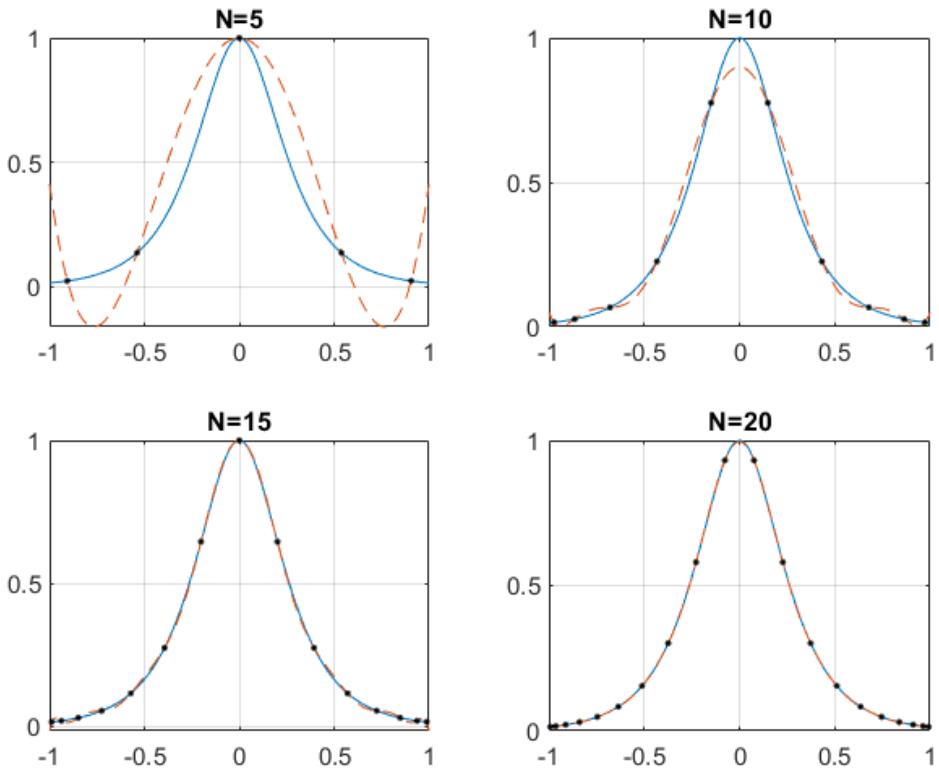


Рисунок 6 – Полиномы Лагранжа для функции $f_2(x)$ (7) для сетки Гаусса-Лежандра.

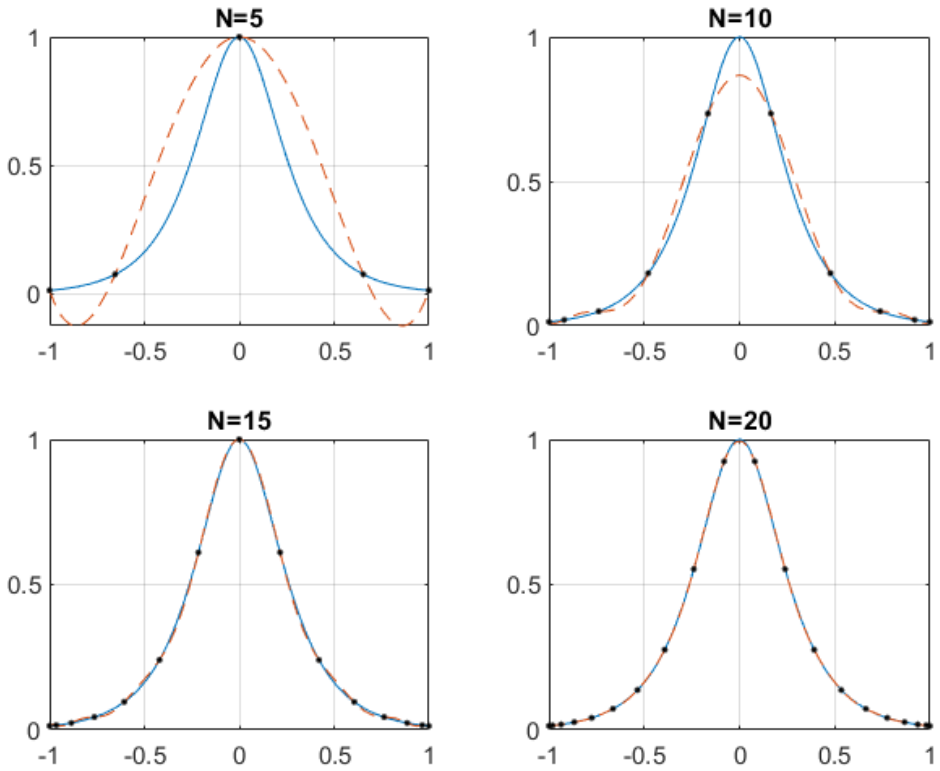


Рисунок 7 – Полиномы Лагранжа для функции $f_2(x)$ (7) для сетки Гаусса-Лежандра-Лобато.

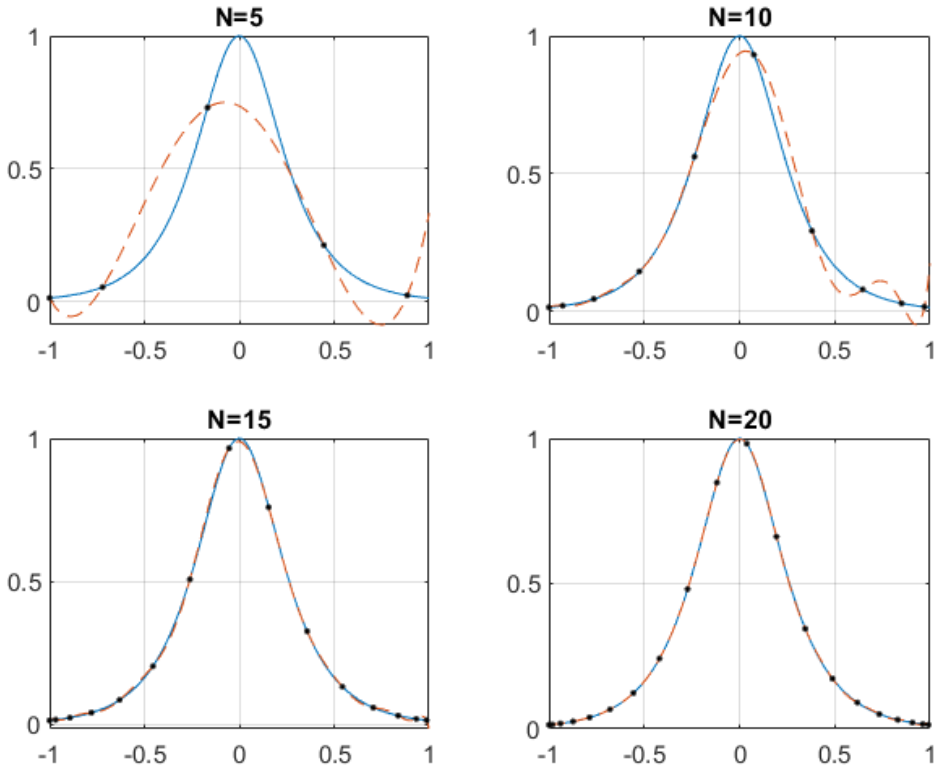


Рисунок 8 – Полиномы Лагранжа для функции $f_2(x)$ (7) для сетки Гаусса-Лежандра-Радао.

Из рисунков можно сделать вывод, что точность аппроксимации будет становиться все лучше и лучше по мере увеличения числа узлов N и феномен Рунге не появляется, поэтому изменение типа сетки на неравномерную помогает избежать появления феномена Рунге.

В целом, псевдоспектральные методы Гаусса и Радау близки друг к другу по точности решения и немного лучше, чем псевдоспектральный метод Лежандра, по оценке сопутствующего состояния. На самом деле, каждый из трех методов имеет свои преимущества и недостатки при решении различных проблем, а результаты тесно связаны с характеристиками конкретной проблемы. Псевдоспектральный метод Лежандра имеет лучшую сходимость для задач оптимального управления с несовершенно свободными концами, особенно когда ограничение на конец является функцией начального или конечного состояния, а псевдоспектральные методы Гаусса и Радау могут не сходиться.

4.3 Сравнение точности интерполяционных полиномов для различных типов сеток

Сравним точность интерполяционных полиномов Лагранжа, построенных для двух функций $f_1(x)$ и $f_2(x)$, заданных в п. 4.1 формулами (6) и (7), с использованием трех типов сеток (Гаусса-Лежандра, Гаусса-Лежандра-Лобато и Гаусса-Лежандра-Радао), а также равномерной сетки. Для этого вычислим значение интерполяционного полинома и исходной функции в $N = 100$ точках, заданных равномерно на интервале $[-1, 1]$. Изменяя количество точек в сетке от $k = 5$ до 20 посчитаем величину ошибки

$$\epsilon_k = \frac{1}{N}\left[\sum_{i=1}^{N}(f(x_i) - L_i(x_i))^2\right]^{1/2},$$

где x_i – это точки равномерной сетки, $x_1 = -1$, $x_N = 1$, а L_i – это интерполяционный полином Лагранжа на k точках, вычисленный для определенного набора точек (равномерная сетка, неравномерная сетка Гаусса-Лежандра (Г.-Л.), Гаусса-Лежандра-Лобато(Г.-Л.-Л.) и Гаусса-Лежандра-Радао (Г.-Л.-Р.)).

В таблицах 1 и 2 приведены результаты вычисления величины ошибки ϵ_k для функций $f_1(x)$ и $f_2(x)$. Результаты вычислений также предствлены графически на рис. 9.

Таблица 1 – Величины ошибки ϵ_k для функции $f_1(x)$

	Равномерная сетка	Л.-Г.	Л.-Г.-Л.	Л.-Г.-Р.
k=5	0.014534	0.016180	0.019417	0.016592
k=10	0.057516	0.006030	0.005877	0.005520
k=15	0.049858	0.002145	0.002528	0.002628
k=20	1.168141	0.000871	0.000763	0.000815

Таблица 2 – Величины ошибки ϵ_k для функции $f_2(x)$

	Равномерная сетка	Л.-Г.	Л.-Г.-Л.	Л.-Г.-Р.
k=5	0.012015	0.012410	0.016063	0.014894
k=10	0.023507	0.002635	0.002463	0.002509
k=15	0.015228	0.000569	0.000706	0.000758
k=20	0.105770	0.000133	0.000110	0.000122

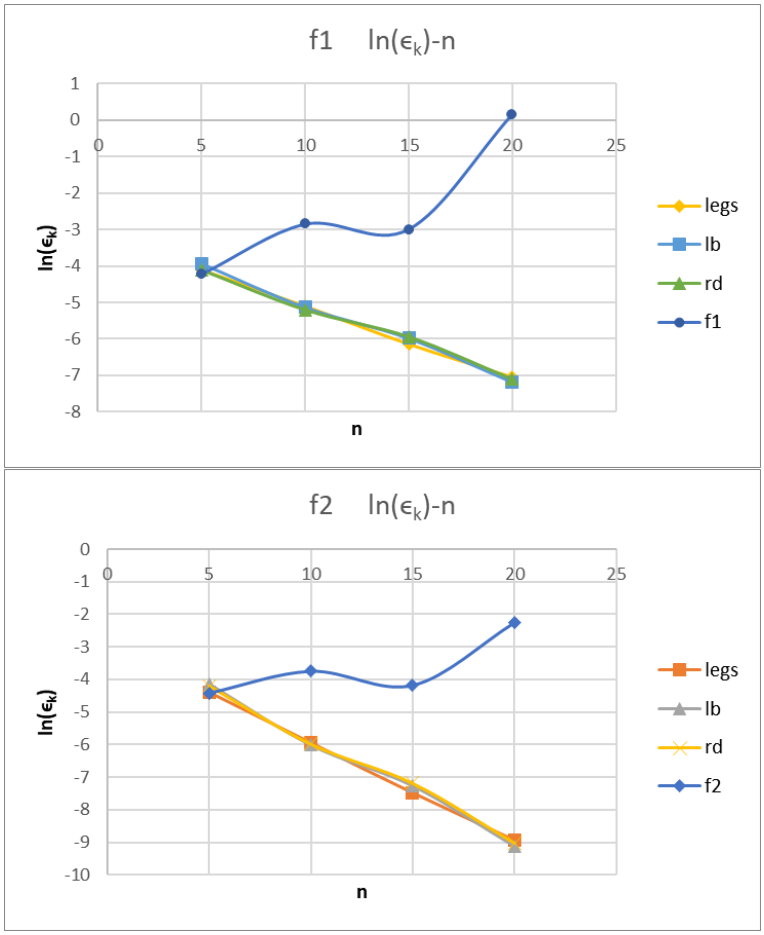


Рисунок 9 – зависимость величины ошибки ϵ_k от числа точек k.

Из полученных данных следует, что чем больше количество точек k в неравномерной сетке, тем меньше величина ошибки ϵ_k. Это связано с тем, что в интервале, где $f(x)$ быстро изменяется (больший наклон), количество узлов x_k для неравномерной сетки больше, чем для равномерной, и эффект интерполяции лучше.

Для равномерной сетки по мере увеличения числа узлов интерполяции k ошибка ϵ_k возрастает, что связано с появлением эффекта Рунге.

Из полученных результатов также следует, что точность интерполяции для трех типов сеток (Гаусса-Лежандра, Гаусса-Лежандра-Лобато и Гаусса-Лежандра-Радао) прктически одинаковая.

5 Использование полиномов Лагранжа для решения дифференциальных уравнений.

Основной принцип метода ортогональной коллокации заключается в следующем: OCP(optimal control problem) преобразуется в задачу нелинейного программирования (NLP-nonlinear programming problem) путем дискретизации непрерывной задачи оптимального управления в точке ортогонального согласования и аппроксимации переменных состояния и управления интерполирующими полиномами [9–11]. Обычно используемыми методами ортогональных коллокаций являются псевдоспектральный метод Гаусса, псевдоспектральный метод Лежандра и псевдоспектральный метод Радау [12], которые соответствуют интегралу LG (Лежандр-Гаусс), интегралу LGL (Лежандр-Гаусс-Лобатто) и интегралу LGR (Лежандр-Гаусс-Радау) соответственно. Ross и др. [13] показали, что NLP, полученные с помощью псевдоспектрального преобразования, сходятся к оптимальному решению OCP со спектральной точностью.

5.1 Метод численного решения дифференциальных уравнений с помощью полиномов Лагранжа

Предположим, что мы выбрали некоторый набор узловых точек $\mathrm{T} = \{\tau_i\}$, где $i = 1, \ldots, n$. Предположим также, что нам известны значения функции $f(t)$ в точках : $\phi_i = f(\tau_i)$. Тогда функция $f(t)$ может быть интерполирована с помощью полинома Лагранжа

$$f(t) = \sum_{i=1}^{n} \phi_i \ell_i(t), \tag{9}$$

где базисные полиномы Лагранжа $\ell_i(t)$ имеют вид

$$\ell_i(t) = \frac{(t-\tau_1)(t-\tau_2)\cdots(t-\tau_{i-1})(t-\tau_{i+1})\cdots(t-\tau_n)}{(\tau_i-\tau_1)(\tau_i-\tau_2)\cdots(\tau_i-\tau_{i-1})(\tau_i-\tau_{i+1})\cdots(\tau_i-\tau_n)}. \tag{10}$$

Используя интерполяционную формулу (9), мы можем вычислить производную функции $f(t)$ в некоторой точке t. В общем случае эта производная

имеет достаточно сложный вид, но ее запись сильно упростится, если мы будем вычислять производную в узловых точках $t = \tau_i$.

Для простоты запишем n-членное умножение как

$$\omega_n(t) = (t - \tau_1)(t - \tau_2) \cdots (t - \tau_n). \tag{11}$$

Тогда значение производной функции $\omega_n(t)$ в узле интерполяции равно

$$\omega'_n(\tau_i) = (\tau_i - \tau_1)(\tau_i - \tau_2) \cdots (\tau_i - \tau_{i-1})(\tau_i - \tau_{i+1}) \cdots (\tau_i - \tau_n). \tag{12}$$

С учетом формулы (10), можно заметить, что

$$\ell_i(t) = \frac{\omega_n(t)}{(t - \tau_i)\omega'_n(\tau_i)}. \tag{13}$$

Тогда производная функции $l_i(t)$ равна

$$\ell'_i(t) = \frac{\omega'_n(t)(t - \tau_i)\omega'_n(\tau_i) - \omega_n(t)\omega'_n(\tau_i)}{(t - \tau_i)^2(\omega'_n(\tau_i))^2}. \tag{14}$$

В результате получаем следующее выражение для производной функции $\ell'_i(t)$ в точке $t = \tau_j$, $i \neq j$:

$$\ell'_i(\tau_j) = \frac{\omega'_n(\tau_j)}{(\tau_j - \tau_i)\omega'_n(\tau_i)}, \tag{15}$$

так как $\omega_n(\tau_j) = 0$.

Если попытаться вычислить производную $\ell'_i(t)$ в точке $t = \tau_i$, то в знаменателе получится 0. В связи с этим, с учетом (10), для вычисления производной $\ell'_i(t)$ в точке $t = \tau_i$ можно использовать равенство

$$\ell'_i(\tau_i) = \sum_{j \neq i} \ell'_j(\tau_i). \tag{16}$$

Итак, формулы (15) и (16) позволяют вычислить производную полинома Лагранжа $\ell'_i(t)$ в узлах $\mathrm{T} = \{\tau_i\}$, где $i = 1, \ldots, n$.

Поскольку выражение (9) для $f(t)$ представляет собой взвешенную сумму базисных полиномов $\ell_i(t)$, то производная $f(t)$ в точке $t = \tau_j$ будет иметь вид

$$f'(t)\big|_{t=\tau_j} = \sum_{i=1}^{n} \phi_i \ell'_i(\tau_j). \tag{17}$$

Если мы обозначим вектор производных функции $f(t)$ в точках как $df = [f'(\tau_1),\ f'(\tau_2), \ldots, f'(\tau_n)]$, а вектор значений функции $f(t)$ в тех же точках как $\Phi = [f(\tau_1),\ f(\tau_2), \ldots, f(\tau_n)] = [\phi_1,\ \phi_2, \ldots, \phi_n]$, то формулу (17) можно переписать в векторно-матричном виде как

$$df = D \cdot \Phi, \tag{18}$$

где D – это матрица дифференцирования:

$$D = \begin{bmatrix} \ell'_1(\tau_1) & \ldots & \ell'_n(\tau_1) \\ \vdots & \ddots & \vdots \\ \ell'_1(\tau_n) & \ldots & \ell'_n(\tau_n) \end{bmatrix}.$$

В листинге 6 приведена программа для вычисления этой матрицы.

Использование полиномов Лагранжа для решения дифференциальных уравнений. Предположим, что мы хотим найти решение дифференциального уравнения

$$\dot{x} = G(x). \tag{19}$$

Некоторая функция $x^*(t)$ будет решением (19), если ее производная в любой точке t будет равна правой части ДУ, вычисленной в этой точке. Потребуем, чтобы это равенство выполнялось в точках T, определенных выше.

Пусть $X = [x^*(\tau_1), \ldots, x^*(\tau_n)]$ – это (пока неизвестный) вектор значений решения $x^*(t)$. Мы будем искать его как решение нелинейного алгебраического уравнения

$$DX = G(X), \tag{20}$$

где слева стоят производные $x^*(t)$, вычисленные с помощью (18), а справа – значения правых частей (19), вычисленные в этих точках.

Уравнение (20) можно решить с использованием функций пакета MATLAB. После того, как найден вектор X, искомое решение дифференциального уравнения (19) строится с помощью интерполяционного полинома Лагранжа по формуле (9).

5.2 Описание вспомогательных функций

1. Функция **fmincon** пакета MATLAB находит минимум нелинейной многомерной функции с ограничениями.

Функция fmincon решает задачу нелинейного программирования, а именно находит минимум целевой функции $\min\limits_{x} f(x)$ таким образом, что

$$\begin{cases} c(x) \leq 0 \\ ceq = 0 \\ A \cdot x \leq b \\ Aeq \cdot x = beq \\ lb \leq x \leq ub \end{cases} \tag{21}$$

где b и beq являются векторами, A и Aeq являются матрицами, c(x) и ceq(x) являются функциями, которые возвращают векторы, и f(x) является функцией, которая возвращает скаляр. f(x), c(x) и ceq(x) могут быть нелинейными функциями. x, lb и ub могут быть переданы как векторы или матрицы.

Функция fmincon применяет четыре различных алгоритма: метод внутренней точки (interior point), алгоритм последовательного квадратичного программирования (sequential quadratic programming algorithm, SQP), метод активного множества (active set), алгоритм отражения доверительной области (trust region reflective).

Метод fmincon предназначен для поиска минимального значения ограниченной нелинейной многомерной функции. Это градиентный метод, который предполагает, что и целевая функция, и ограничения непрерывны и имеют производные первого порядка. Если эти условия отсутствуют, рекомендуется использовать fminsearch.

2. Функция **ode23** – решение нежестких дифференциальных уравнений. Метод низкого порядка точности.

[t,y] = ode23(odefun, tspan,y0), где tspan = [t0 tfl.

Метод интегрирует систему дифференциальных уравнений $y' = f(t, y)$ от

t0 до tf с начальными условиями y0. Каждая строка в массиве решения y соответствует значению, возвращенному в вектор-столбце t. Решатели ОДУ могут решить системы уравнений формы $y' = f(t, y)$, или проблемы, которые включают матрицу $M(t, y)y' = f(t, y)$.

Все решатели используют подобный синтаксис. Решатель ode23s может решить задачи с матрицей, если эта матрица является постоянной. ode15s и ode23t могут решить задачи с матрицей, которая сингулярна. Эта задача известна как дифференциально-алгебраические уравнения (ДАУ),

ode23 – это реализация явной пары Рунге-Кутта Богацкого и Шампине. Она может быть более эффективной, чем ode45 при грубых допусках и при наличии умеренной жесткости.

5.3 Решение дифференциального уравнения

В качестве примера рассмотрим дифференциальное уравнение

$$\dot{x} = e^{-x} sin(x), \quad x(0) = 1$$

на интервале $t \in [0, 1]$.

Нам нужно использовать численную схему для решения системы уравнений (20):

$$DX = G(X),$$

где X – вектор из n+1 переменных, $G(x) = e^{-x} sin(x)$. Мы определяем функцию, которая принимает вектор X и точки сетки t (на неравномерной сетке, построенной методами Гаусса-Лежандра-Радао) и возвращает норму вектора расхождения. Затем мы будем использовать это значение для поиска решения.

Мы можем вычислить правую часть G(x), используя x. Затем нам нужно вычислить расхождение dF = ||DX-G(X)||. После этого наша цель состоит в определении такого X, чтобы разность ||DX-G(X)|| была как можно меньше. Для этого мы будем использовать функцию fmincon.

Наконец, мы сравним результаты, полученные с помощью ode23, с результатами, полученными с помощью метода полиномов Лагранжа. На рис. 10 показаны решения, полученные этими методами. Из рисунка видно, что траектории достаточно близки.

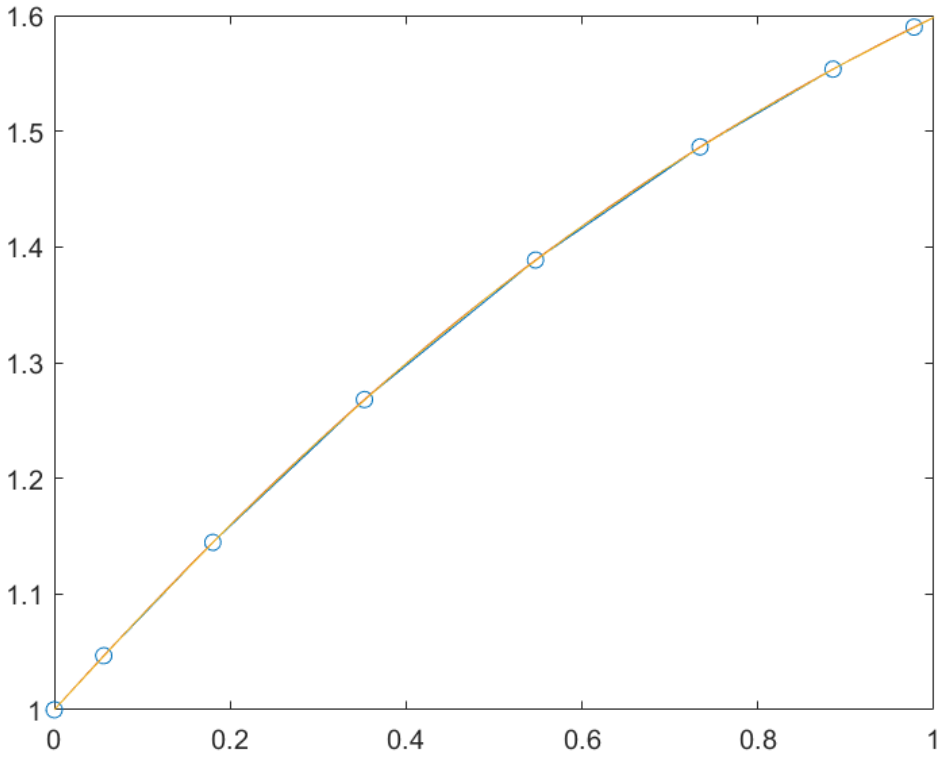


Рисунок 10 – Результаты метода интерполяционного полинома Лагранжа и ode23.

В листинге 7 приведен текст программы для решения дифференциального уравнения с помощью метода полиномов Лагранжа на неравномерной сетке и с использованием численного метода ode23 пакета Matlab.

6 Заключение

В результате проведенной работы получены следующие результаты:

1. Построены полиномы Лагранжа для тестовых функций с использованием равномерных сеток и с различным числом точек. Продемонстрирован феномен Рунге при увеличении числа узлов N. Предложено противодействовать эффекту Рунге, используя неравномерную сетку.

2.Построены полиномы Лагранжа с использованием неравномерных сеток Гаусса-Лежандра, Гаусса-Лежандра-Лобато и Гаусса-Лежандра-Радао. Из полученных результатов следует, что применение неравномерных сеток позволяет избежать появления эффекта Рунге. Проведено сравнение точности интерполяции на различных сетках и показано, что точность интерполяции для рассмотренных трех вариантов неравномерных сеток практически одинаковая.

3. Рассмотрен метод построения приближенного решения дифференциальных уравнений с использованием полиномов Лагранжа с неравномерной сеткой. Этот метод состоит в том, что решение дифференциального уравнения сводится к решению системы нелинейных алгебраических уравнений. Для иллюстрации данного подхода, приведен пример решения дифференциального уравнения и выполнено сравнение с решением, полученным стандартным методом ode23 пакета MATLAB.

4. Разработаны программы в среде MATLAB для построяния интерполяционных полиномов Лагранжа на равномерной и неравномерной сетках, а также для решения дифференциальных уравнений с использованием полиномов Лагранжа.

Список литературы

[1] Лев Семенович Понтрягин, Владимир Григорьевич Болтянский, Р В Гамкрелидзе, and Евгений Фролович Мищенко. *Математическая теория оптимальных процессов*. 1969.

[2] David Benson. *A Gauss pseudospectral transcription for optimal control*. PhD thesis, Massachusetts Institute of Technology, 2005.

[3] Fariba Fahroo and I Michael Ross. Advances in pseudospectral methods for optimal control. In *AIAA guidance, navigation and control conference and exhibit*, page 7309, 2008.

[4] I Michael Ross and Fariba Fahroo. Pseudospectral knotting methods for solving nonsmooth optimal control problems. *Journal of Guidance, Control, and Dynamics*, 27(3):397–405, 2004.

[5] Jean-Paul Berrut and Lloyd N Trefethen. Barycentric lagrange interpolation. *SIAM review*, 46(3):501–517, 2004.

[6] Павел Суетин. *Классические ортогональные многочлены*. Litres, 2018.

[7] Carl Runge. Über empirische funktionen und die interpolation zwischen äquidistanten ordinaten. *Zeitschrift für Mathematik und Physik*, 46(224-243):20, 1901.

[8] Divya Garg, Michael Patterson, William Hager, Anil Rao, David Benson, and Geoffrey Huntington. An overview of three pseudospectral methods for the numerical solution of optimal control problems. 2017.

[9] Fariba Fahroo and I Michael Ross. Costate estimation by a legendre pseudospectral method. *Journal of Guidance, Control, and Dynamics*, 24(2):270–277, 2001.

[10] I Michael Ross and Fariba Fahroo. A direct method for solving nonsmooth optimal control problems. *IFAC Proceedings Volumes*, 35(1):479–484, 2002.

[11] Wei Kang, I Michael Ross, and Qi Gong. Pseudospectral optimal control and its convergence theorems. In *Analysis and design of nonlinear control systems*, pages 109–124. Springer, 2008.

[12] Divya Garg, Michael Patterson, William W Hager, Anil V Rao, David A Benson, and Geoffrey T Huntington. A unified framework for the numerical solution of optimal control problems using pseudospectral methods. *Automatica*, 46(11):1843–1851, 2010.

[13] I Michael Ross and F Fahroo. Convergence of pseudospectral discretizations of optimal control problems. In *Proceedings of the 40th IEEE Conference on Decision and Control (Cat. No. 01CH37228)*, volume 4, pages 3175–3177. IEEE, 2001.

7 Листинги программ

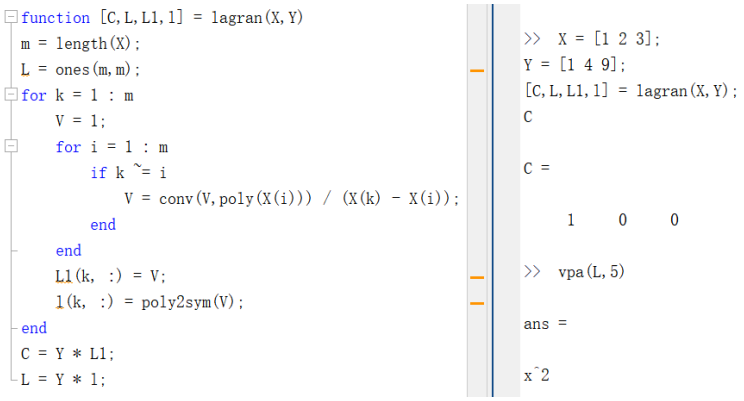
1. Программа для вычисления полинома Лагранжа.

Find Lagrange interpolation polynomials and basis functions

Input quantity: n+1 nodes (x_i, y_i) (i = 1,2, ..., n+1) abscissa vector X, ordinate vector Y

Output quantity: Lagrangian interpolation polynomial of degree n L and its coefficient vector C, basis function l and its coefficient matrix L

```
function [C,L,L1,l] = lagran(X,Y)
m = length(X);
L = ones(m,m);
for k = 1 : m
    V = 1;
    for i = 1 : m
        if k ~= i
            V = conv(V,poly(X(i))) / (X(k) - X(i));
        end
    end
    L1(k, :) = V;
    l(k, :) = poly2sym(V);
end
C = Y * L1;
L = Y * l;
```



```
function [C,L,L1,l] = lagran(X,Y)
m = length(X);
L = ones(m,m);
for k = 1 : m
    V = 1;
    for i = 1 : m
        if k ~= i
            V = conv(V,poly(X(i))) / (X(k) - X(i));
        end
    end
    L1(k, :) = V;
    l(k, :) = poly2sym(V);
end
C = Y * L1;
L = Y * l;
```

```
>> X = [1 2 3];
Y = [1 4 9];
[C,L,L1,l] = lagran(X,Y);
C

C =

     1     0     0

>> vpa(L,5)

ans =

x^2
```

Рисунок 11 – Программа для вычисления полинома Лагранжа в Matlab.

2. Программа для построения интерполяционного полинома Лагранжа на равномерной сетке для заданной функции и построение графиков.

```
syms x0 x1 y x n x_para
x0 = -1:0.02:1; %As a drawing point 101
y = 1./(1+25.*(x0.^2));

for n = [5,10,15,20]
    k = 0:n; %k ranges from 0 to n, a total of n+1 items

   x(k+1) = -1+2*k/n;
   %From x(1) to x(n+1), a total of n+1 items, that is,
   %from the 0-th point to the n-th point
    %[x,w]=legs(n+1);
    %x = x;

    for i = 1:(n+1)
    %From x(1) to x(n+1), a total of n+1 items, that is,
    %from the 0th point to the nth point
        part_x(i) = (x_para-x(i));
    %The composition of each factor, a total of k+1 items
    end
    for i = 1:(n+1) %Total of n+1 items
        W1_x(i) = 1;
        for j = 1:(n+1)
            if i~=j
                W1_x(i) = W1_x(i)*(x(i)-x(j));
    %The composition of each factor, a total of k+1 items
            end
```

```
        end
    end
    W_x(n+1) = prod(part_x);
    %Multiplication function to find w_x
    for j = 1:(n+1)
        ln_x(j) = (1/(1+25*(x(j)^2))) * W_x(n+1)
        /((x_para-x(j))*(W1_x(j)));
    end

    Ln_x = sum(ln_x);
    x_vector = -0.99:0.008:0.99;
    Ln_x = subs(Ln_x,x_para,x_vector);
    m = n/5;
    subplot(2,2,m);plot(x0,y);
    hold on
    subplot(2,2,m);plot(x_vector,Ln_x);
    legend ('original function','n=x')
    ylabel('f(x) and Ln(x) ');
end
```

3. Программа для реализации метода Лежандра-Гаусса.

x=legs(n) returns n Legendre-Gauss points arranged in ascending order

[x,w]= legs(n) returns n Legendre-Gauss points and weights

Newton iteration method is used for computing nodes

```
function [varargout]=legs(n)

% Compute the initial guess of the interior LGL points
 thetak=(4*[1:n]-1)*pi/(4*n+2);
 ze=-(1-(n-1)/(8*n^3)-(39-28./sin(thetak).^2)/(384*n^4))
 .*cos(thetak);
 ep=eps*10;% error tolerance for stopping iteration
 ze1=ze+ep+1;

while max(abs(ze1-ze))>=ep,% Newton's iteration procedure
      ze1=ze;
      [dy,y]=lepoly(n,ze);
      ze=ze-y./dy;
end; % around 6 iterations are required for n=100
   varargout{1}=ze';
if nargout==1, return; end;

% Use the weight expression (3.178) to compute the weights
 varargout{2}=(2./((1-ze.^2).*dy.^2))';
```

4.Программа для реализации метода Лежандра-Гаусса-Лобато.

x=legslb(n) returns n Legendre-Gauss-Lobatto points with x(1)=-1, x(n)=1

[x,w]= legslb(n) returns n Legendre-Gauss-Lobatto points and weights

Newton iteration method is used for computing nodes

```
function [varargout]=legslb(n)

% Compute the initial guess of the interior LGL points
 nn=n-1; thetak=(4*[1:nn]-1)*pi/(4*nn+2);
 sigmak=-(1-(nn-1)/(8*nn^3)-(39-28./sin(thetak).^2)/(384*
 nn^4)).*cos(thetak);
 ze=(sigmak(1:nn-1)+sigmak(2:nn))/2;
 ep=eps*10;% error tolerance for stopping iteration
 ze1=ze+ep+1;

while max(abs(ze1-ze))>=ep,% Newton's iteration procedure
      ze1=ze;
      [dy,y]=lepoly(nn,ze);
      ze=ze-(1-ze.*ze).*dy./(2*ze.*dy-nn*(nn+1)*y);
end;% around 6 iterations are required for n=100
   varargout{1}=[-1,ze,1]';
if nargout==1, return; end;

% Use the weight expression (3.188) to compute the weights
 varargout{2}=[2/(nn*(nn+1)),2./(nn*(nn+1)*y.^2),2/(nn*
 (nn+1))]';
```

5. Программа для реализации метода Лежандра-Гаусса-Радао.

x=legsrd(n) returns n Legendre-Gauss-Radau points with x(1)=-1

[x,w]= legsrd(n) returns n Legendre-Gauss-Radau points and weights

Eigenmethod is used for computing nodes

```
function [varargout]=legsrd(n)

j=[0:n-2];% indices
A=diag(1./((2*j+1).*(2*j+3)));% Main diagonal
j=[1:n-2];
A=A+diag(sqrt(j.*(j+1))./(2*j+1),1) ...
  +diag(sqrt(j.*(j+1))./(2*j+1),-1);% Create Jacobi matrix
x= sort(eig(sparse(A)));% Compute eigenvalues
x=[-1;x];
      varargout{1}=x;
  if nargout==1, return; end;
y=lepoly(n-1,x);
varargout{2}= (1-x)./(n^2*y.^2);
% return the weights by (3.179)
```

6. Программа для вычисления матрицы D.

dL(i,j)=derivative of the j-th Lagrange polynomial L_j at p_i

Every basis lagrange polynomial can be represented in baricentroc form as $L_i(x) = L/(x - x_i)/L'(x_i)$,we compute derivatives of L_i at x_k

```
function dL=dLag(N,p)
if length(p)~=N
    error('The number of input points must be equal to N')
end
if ~isrow(p)
    p=p';
end

iL=zeros(N);

% L(x)
L=poly(p);
% L'(x)
dL=polyder(L);
% vector of L'(x_i)
dLe=polyval(dL,p);

r=zeros(N,N-1);
Lp=zeros(N,N);
iLp=zeros(N,N+1);

for j=1:N
    % r(j,:) - roots x_i, i!=j
    r(j,:)=[p(1:j-1),p(j+1:N)];
```

```
        % Lp(j)=L(x)/(x-x_j), i.e. we exclude the jth root
        Lp(j,:)=poly(r(j,:));
        % iLp(j)=derivative of Lp(j)
        dLp(j,:)=polyder(Lp(j,:));
        for i=1:N
            dL(i,j)=polyval(dLp(j,:),p(i))/dLe(j);
        end
    end
end
```

7. Использование полиномов Лагранжа для решения дифференциальных уравнений.

```
function dy()
n=7;
x0=1;
[x,w]=legsrd(n+1);
%x = (-1:0.1:1)';
t = (x+1)/2;

% differentiation matrix
D1=dLag(n+1,t);
Dn=D1(2:end,:);

function diff = my_eq(xg)
Xg=[x0;xg];
tg=t(2:end);
F=exp(-tg).*sin(xg);
% now we need to compute the discrepancy dF = D*X-F(X,t)
dF=Dn*Xg-F;
diff=norm(dF);
end

x_guess = ones(n,1);
x_res = fmincon(@my_eq,x_guess,[],[],[],[],zeros(n,1))
X_res=[x0; x_res];

figure()
plot(t,X_res,'o-')
```

```
hold on

tt=0:0.01:1;
xt=lagrange(tt,t',X_res');
plot(tt,xt)

function dx=de_eq(t,x)
dx=exp(-t)*sin(x);
end
% now use ode23
[tnum,xnum] = ode23(@de_eq, [0,1], x0);
plot(tnum,xnum)
end
```