ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
"НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ "МЭИ"

ОТЧЕТ

По курсовой работе

Дисциплина: «Численные методы»

Тема: «Решение интегральных уравнений методом квадратур»

Группа: А-05-19
Студент: Ушаков Н.А.
Преподаватель: Амосова О.А.

Москва 2021

Содержание

Список иллюстраций

1 Задание курсовой работы

Раздел 1. Задача 1.2

Найти решение интегрального уравнения:

$$u(x) - \lambda \int_a^b K(x,s)u(s)ds = f(x), \quad x \in [a,b]$$

методом квадратур для каждого значения λ из указанного отрезка $[\alpha,\beta]$. При каждом значении λ построить график решения и вычислить площадь полученной криволинейной трапеции. Определить, при каком значении площадь трапеции максимальна.

Порядок решения задачи

1. Составить систему линейных уравнений на основе квадратурной формулы индивидуального варианта.

2. Составить процедуру решения СЛАУ указанным в индивидуальном варианте методом.

3. Составить программу интерполирования функции правой части уравнения указанным в индивидуальном варианте методом.

4. Решить исходную задачу.

Данные

Задача:

Ядро	Отрезки $[a,b]$ $[\alpha,\beta]$	Метод интерполяции функции $f(x)$	Квадратурная формула	Решение СЛАУ
$e^{-\|x-t\|}$	$[1,2]$ $[0.2,0.7]$	Многочлен Лагранжа	Метод трапеций	Метод сопряженных градиентов

Функция $f(x)$:

x	1	1.1	1.2	1.3	1.4	1.5	1.6	1.7	1.8	1.9	2
$f(x)$	7.679	7.329	7.012	6.725	6.466	6.231	6.012	5.827	5.653	5.496	5.353

2 Теория

2.1 Уравнение Фредгольма второго рода

Уравнение Фредгольма второго рода имеет следующий вид:

$$u(x) - \lambda \int_a^b K(x,s)u(s)ds = f(x), \quad x \in [a,b] \tag{1}$$

Здесь $u(x)$ - неизвестная функция, $K(x,s)$ - ядро интегрального уравнения, $f(x)$ - свободный член уравнения, λ - числовой параметр.

2.2 Метод квадратур

Одним из наиболее простых методов решения интегральных уравнений с гладкими ядрами является *метод квадратур*, основанный на аппроксимации значений интегрального оператора, входящего в уравнение с использованием одной из квадратурных формул. [1]

Построим на отрезке $[a,b]$ сетку с узлами $x_1,x_2,\ldots,x_n$. Запишем уравнение (1) в узлах сетки:

$$u(x_i) - \lambda \int_a^b K(x_i,s)u(s)ds = f(x_i), \quad i = 1,2,\ldots,N \tag{2}$$

Пусть за основу взята квадратурная формула:

$$\int_a^b g(s)ds \approx \sum_{j=1}^{N} c_j g(x_j) \tag{3}$$

Используя (3) при $g(s) = K(x,s)u(s)$ для приближенного вычисления значения интегрального оператора, имеем:

$$\int_a^b K(x,s)u(s)ds \approx \sum_{j=1}^{N} c_j K(x,x_j)u(x_j) \tag{4}$$

Подставив записанную выше аппроксимацию (4) в равенства (3), получим следующую систему линейных алгебраических уравнений:

$$u_i - \lambda \sum_{j=1}^{N} c_j K_{ij} u_j = f_i, \quad i = 1,2,\ldots,N \tag{5}$$

Здесь: $u_i = u(x_i)$, $\quad f_i = f(x_i)$, $\quad K_{ij} = K(x_i,x_j)$, $\quad c_j > 0$ - веса квадратуры.

Используем указанную в задании квадратурную *формулу трапеций*. Тогда СЛАУ примет следующий вид [2]:

$$u_i - \lambda \sum_{j=1}^{N} \omega_j K_{ij} u_j = f_i, \quad i = 1,2,\ldots,N \tag{6}$$

Где ω_j - веса формулы трапеций:

$$\omega_j = \begin{cases} \frac{1}{2}, & j = 1,N \\ 1, & j = 2,3,\ldots,N-1 \end{cases}$$

Введем матрицу B_N с элементами $b_{ij} = \omega_j K_{ij}, 0 \leq i,j \leq N$, вектор свободных членов $f_N = (f_0,f_1,\ldots,f_N)^T$ и вектор неизвестных $u_N = (u_0,u_1,\ldots,u_N)^T$. Запишем систему (6) в матричном виде и перенесем часть, содержающую B_N направо:

$$u_N = \lambda B_N u_N + f_N$$

Эту же систему можно записать в виде, удобном для решения СЛАУ:

$$\lambda A_N u_N = f_N \tag{7}$$

Где: $A_N = E_N - B_N$, а E_N - единичная матрица.

Примечание

При написании алгоритма поиска решения интегрального уравнения, опираясь на вышеизложенное, будем сначала определять матрицу B_N, используя квадратурную формулу трапеций, после чего, вычитая из нее единичную, находить матрицу A_N (7). Задавая вектор неизвестных u_N и вектор свободных членов f_N, решая СЛАУ, получим искомый ответ.

Приведем некоторый допольнительный теоретический материал и перейдем к *интерполированию* функции и *решению СЛАУ*.

2.3 Теория Фредгольма

Введем следующую норму [1]:

$$\|K\| = \sqrt{\int_a^b \int_a^b |K(x,s)|^2 dxds}$$

Теорема 16.5 (Единственность решения)

Пусть ядро K удовлетворяет условию:

$$\|K\| = \sqrt{\int_a^b \int_a^b |K(x,s)|^2 dxds} < 1 \tag{8}$$

Тогда при любой правой части f уравнение (1) имеет единственное решение и справедлива оценка:

$$\|u\| = C_2\|f\|, \quad C_2 = \frac{1}{1 - \|K\|} \tag{9}$$

2.4 Невязка и погрешность

Введем интегральный оператор $\mathcal{K}$:

$$\mathcal{K}u(x) = \int_a^b K(x,s)u(s)ds$$

Наряду с развернутой формой записи интегрального уравнения (1) будем использовать его краткую операторную форму записи:

$$u = \lambda\mathcal{K}u + f \tag{10}$$

Пусть u - точное решение интегрального уравнения (10), а $\widetilde{u}$ - некоторое приближенное решение того же уравнения. Определим невязку, отвечающую этому приближенному решению:

$$r[\widetilde{u}] = \widetilde{u} - \lambda\mathcal{K}\widetilde{u} - f \tag{11}$$

В развернутой форме записи невязка задается формулой:

$$r[\widetilde{u}](x) = \widetilde{u}(x) - \lambda\int_a^b K(x,s)\widetilde{u}(s)ds - f(x), \quad x \in [a,b]$$

Заметим, что точному решению u отвечает нулевая невязка:

$$r[u] = u - \lambda\mathcal{K}u - f = 0 \tag{12}$$

Вычитая из равенства (12) равенство (11), убеждаемся в том, что погрешность $u - \widetilde{u}$ является решением уравнения того же вида, что и (10), но с невязкой $r[\widetilde{u}]$ в роли правой части f:

$$u - \widetilde{u} = \lambda\mathcal{K}(u - \widetilde{u}) + r[\widetilde{u}]$$

Если выполняется неравенство (8) и справедлива оценка (9), то погрешность $u - \widetilde{u}$ можно оценить через невязку следующим образом:

$$\|u - \widetilde{u}\| \leq C_2 \, \|r[\widetilde{u}]\| \tag{13}$$

Таким образом, если невязка, отвечающая приближенному решению $\widetilde{u}$, достаточно мала, то малой будет и погрешность $u - \widetilde{u}$.

Теперь стоит обсудить интерполирование функции правой части $f(x)$ интегрального уравнения.

2.5 Многочлен Лагранжа

Данный многочлен имеет следующий вид:

$$L_N(x) = \sum_{i=0}^{N} f_i \prod_{k=0 k\neq i}^{N} \frac{(x - x_k)}{(x_i - x_k)} \tag{14}$$

Он представляет собой сумму $N+1$ слагаемого, каждое из которых есть многочлен степени N. При $k \neq i$ k-ое слагаемое обращается в ноль, а при $k = i$ числитель и знаменатель дроби совпадают.

Перейдем к решению СЛАУ (7).

2.6 Метод сопряженных градиентов

Пусть дана система линейных уравнений [3]:

$$Ax = b \tag{15}$$

Причем матрица системы (15) – это симметричная положительно определенная матрица, то есть: $A = A^T > 0$. Тогда процесс решения СЛАУ можно представить как минимизацию следующего функционала:

$$F(x) = \langle Ax,x\rangle - 2\langle b,x\rangle \to \min \tag{16}$$

Где за $\langle\cdot,\cdot\rangle$ обозначено скалярное произведение.

Идея метода сопряженных градиентов состоит в следующем [4]

Пусть квадратичный функционал (16) имеет минимум в точке x_* и $\{p_k\}_{k=0}^{n}$ – базис в пространстве $\mathbb{R}^n$. Тогда для любой точки $x_0 \in \mathbb{R}^n$ вектор $x_* - x_0$ раскладывается по базису $x_* - x_0 = \alpha_1 p_1 + \dots + \alpha_n p_n$. Таким образом, x_* представимо в виде:

$$x_* = x_0 + \alpha_1 p_1 + \dots + \alpha_n p_n$$

Каждое новое приближение вычисляется по формуле:

$$x_k = x_0 + \alpha_1 p_1 + \dots + \alpha_k p_k \tag{17}$$

Определение

Два вектора p и q называются *сопряженными* относительно симметричной матрицы B, если $\langle Bp,q\rangle = 0$.

Опишем способ построения базиса

В качестве начального приближения x_0 выбираем произвольный вектор. На каждой итерации α_k выбираются по правилу:

$$\alpha_k = \underset{\alpha\in\mathbb{R}}{\mathrm{argmin}} F\left(x_{k-1} + \alpha_k p_k\right) \tag{18}$$

Где argmin – аргумент, при котором данное выражение достигает минимума.

Базисные вектора вычисляются по формулам:

$$p_1 = -\nabla F(x_0), \qquad p_{k+1} = -\nabla F(x_k) + \beta_k p_k \tag{19}$$

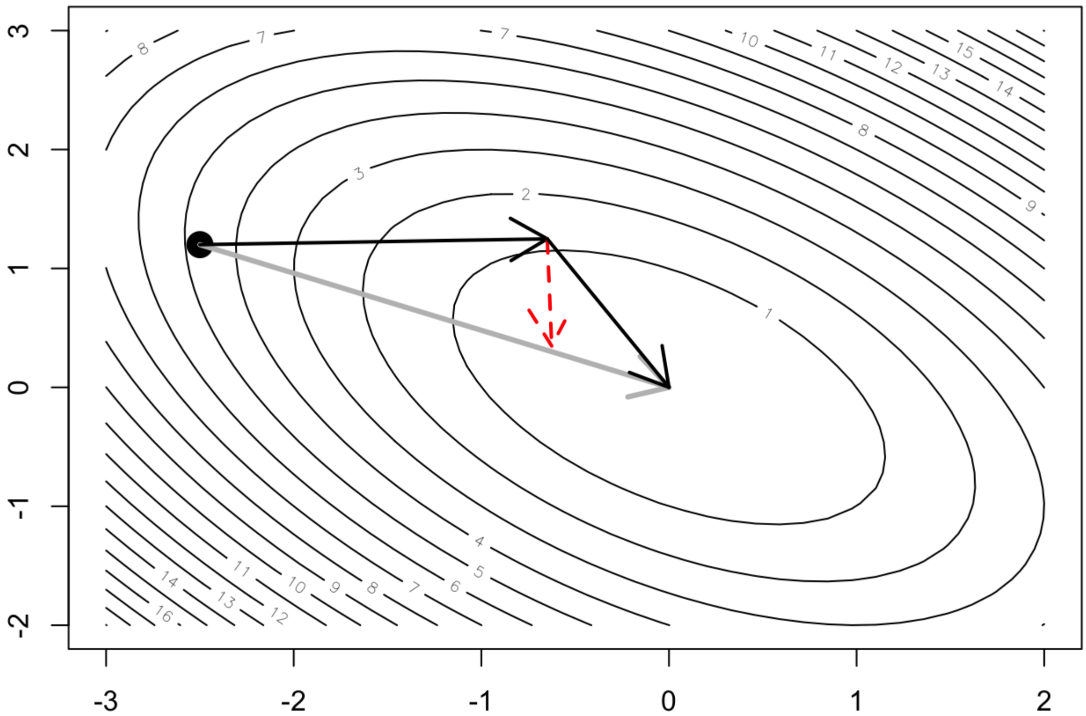
Коэффициенты β_k выбираются так, чтобы векторы p_k и $p_k + 1$ были сопряженными относительно A:

$$\beta_k = \frac{\langle F(x_k), Ap_k\rangle}{\langle Ap_k, p_k\rangle} \tag{20}$$

Если ввести следующий вектор невязки: $r_k = b - Ax_k = -\nabla F(x_k)$, то после нескольких упрощений получим окончательные формулы для: (17), (18), (19), (20), используемые при применении метода сопряженных градиентов на практике:

1. Начальный вектор невязки: $r_0 = b - Ax_0$
2. Начальное направление спуска: $p_0 = r_0$
3. Параметр: $\alpha_{k+1} = \dfrac{\langle r_k, r_k \rangle}{\langle Ap_k, p_k \rangle}$
4. Шаг итерации: $x_{k+1} = x_k + \alpha_{k+1} p_k$
5. Вектор невязки: $r_{k+1} = r_k - \alpha_{k+1} A p_k$
6. Параметр: $\beta_{k+1} = \dfrac{\langle r_{k+1}, r_{k+1} \rangle}{\langle r_k, r_k \rangle}$
7. Вектор направления: $p_{k+1} = r_{k+1} + \beta_{k+1} p_k$
8. Критерий окончания итераций: $\dfrac{\|r_k\|}{\|r_0\|} \leq \varepsilon$

Рис. 1: Графическая интерпретация сопряженных градиентов



3 Составление алгоритма

Перейдем к непосредственной реализации алгоритма и решению поставленной задачи. Подключаем необходимые библиотеки для работой с графиками, массивами, и определенными интегралами:

```
import matplotlib.pyplot as plt
import numpy as np
import scipy.integrate as inteq
from itertools import cycle
```

3.1 Основные функции

Напишем функции интерполяции многочленом Лагранжа, решения СЛАУ методом сопряженных градиентов, вычисления Евклидовой нормы вектора, получения коэффициентов квадратурной формулы трапеций:

```
# Значение многочлена Лагранжа в точке
def lagrange(x, y, x_0):
    polynom = 0
    for i in range(len(y)):
        prod = 1
        for k in range(len(x)):
            if k != i:
                prod *= ((x_0 - x[k]) / (x[i] - x[k]))
        polynom += y[i] * prod
    return polynom

# Решение СЛАУ методом сопряженных градиентов
def conjugate_gradient(A, b, eps):
    n = A.shape[0]
    x = np.zeros(n)
    # Вычисляем вектор невязки
    r0 = b - A @ x
    r = r0
    p = r
    n_r0 = euclid_norm(r0)
    while euclid_norm(r) / n_r0 > eps:
        # Вычисляем параметр α
        a = np.dot(r, r) / np.dot(A @ p, p)
        x = x + a * p
        r_next = r - a * (A @ p)
        # Вычисляем параметр β
        B = np.dot(r_next, r_next) / np.dot(r, r)
        p = r_next + B * p
```

```
        r = r_next
    return x

# Евклидова норма вектора
def euclid_norm(x):
    s = 0
    n = x.shape[0]
    for i in range(n):
        s += (x[i])**2
    return np.sqrt(s)

# Коэффициенты квадратурной формулы трапеций
def trapezoid(j):
    return 1/2*h if j == 0 or j == n-1 else h
```

3.2 Метод и валидация полученного решения

Опираясь на вышеуказанные рассуждения и формулы, составим алгоритм поиска решения интегрального уравнения:

```
# Вычисление нормы ядра интегрального уравнения
def K_norm(K, a, b):
    return np.sqrt(inteq.dblquad(lambda x, s: np.abs(K(x, s))**2, a, b,
                                 lambda x: a, lambda x: b)[0])

# Решение интегрального уравнения Фредгольма второго рода
def fredholm(K, a, b, h, lambd = 1, f = None, normres = True):
    # Задаем число точек и сетку
    n = int((b - a) / h) + 1
    x = np.linspace(a, b, n)

    # Определяем матрицу B
    B = np.array([[lambd * trapezoid(j) * K(x[i], x[j]) for j in range(n)]
                  for i in range(n)])
    E = np.eye(len(B))
    # Вычисляем матрицу A
    A = E - B

    # Задаем данные для интерполяционной функции
    x_data = np.array([1., 1.1, 1.2, 1.3, 1.4, 1.5, 1.6, 1.7, 1.8, 1.9, 2.])
    y_data = np.array([7.679, 7.329, 7.012, 6.725, 6.466, 6.231,
                       6.012, 5.827, 5.653, 5.496, 5.353])
    # Составляем вектор правой части в зависимости от функции
    # Если параметр f = None - табличная функция, иначе - заданная аналитически
    F = np.array([lagrange(x_data, y_data, x[i])
```

```python
                for i in range(n)]) if f is None
                else np.array([f(x[j]) for j in range(n)])

    # Решаем СЛАУ методом сопряженных градиентов с заданной точностью
    u = conjugate_gradient(A, F, eps)

    # Вычисляем норму для ядра интегрального уравнения
    k_norm = K_norm(K, a, b)
    # Вычисляем коэффициент C2
    C2 = 1 / (1 - k_norm)

    # Задаем вектор невязки, интеграл вычисляем приближенно
    r = np.array([u[i] - F[i] - lambd * sum([trapezoid(j) * K(x[i], x[j]) * u[j]
                                    for j in range(n)]) for i in range(n)])
    # Вычисляем Евклидову норму для вектора невязки
    residual = C2 * euclid_norm(r) if normres is True else r

    # Будем теперь возвращать сетку x, решение u, погрешность residual
    return x, u, residual
```

Примечание 1

Для компактной записи, при формировании векторов и матриц, используются генераторы массивов, вместо привычных конструкций с циклами for. В данном методе, в качестве свободного члена можно указать функцию, заданную как таблично, так и аналитически, за что отвечает параметр f, значение по умолчанию которого – None, указывает на таблично заданную функцию. Числовой параметр lambd, так же имеет значение по умолчанию равное единице.

Примечание 2

В соответствии с формулами $(9), (11), (13)$ в данном алгоритме реализовано нахождение вектора невязки и вычисление погрешности решения, с помощью его нормы. В качестве возвращаемого значения функция выдает вектор x, вектор решения $u(x)$, вектор невязки $r[u(x)]$ или его норму $\|r[u(x)]\|$ – в зависимости от параметра normres.

4 Тестирование алгоритма

Чтобы убедиться в корректности работы написанного алгоритма, составим несколько тестовых примеров.

4.1 Тестовый пример 1

4.1.1 Аналитическое решение

Общий вид интегрального уравнения:

$$u(x) - \lambda \int_a^b K(x,s)u(s)ds = f(x), \quad x \in [a,b]$$

Положим: $K(x,s) = 1 - xs, \quad f(x) = x^3, \quad [a,b] = [0,1], \quad \lambda = 1.$

Имеем:

$$u(x) = \int_0^1 (1 - xs)u(s)ds + x^3 \tag{21}$$

Обозначим следующие константы:

$$\begin{cases} C_1 = \int_0^1 u(s)ds \\ C_2 = \int_0^1 su(s)ds \end{cases} \tag{22}$$

Представим решение уравнения (21) в виде:

$$u(x) = C_1 - C_2x + x^3 \tag{23}$$

Заменим x на s в (23) и подставим в (22):

$$\begin{cases} C_1 = \int_0^1 \left(C_1 - C_2s + s^3\right) ds \\ \\ C_2 = \int_0^1 \left(C_1s - C_2s^2 + s^4\right) ds \end{cases}$$

$$\begin{cases} C_1 = C_1 - \dfrac{C_2}{2} + \dfrac{1}{4} \\ \\ C_2 = \dfrac{C_1}{2} - \dfrac{C_2}{3} + \dfrac{1}{5} \end{cases} \Rightarrow \begin{cases} C_1 = \dfrac{14}{15} \\ \\ C_2 = \dfrac{1}{2} \end{cases} \tag{24}$$

Подставим полученные из (24) C_2 и C_2 в (23):

$$u(x) = x^3 - \frac{1}{2}x + \frac{14}{15}$$

Получено аналитическое решение. Далее, найдем численное решение тестового примера, используя построенный алгоритм. Графики решений, полученные обеими методами, должны совпадать, а площади криволинейных трапеций – быть примерно равными, с учетом некоторой погрешности.

4.1.2 Численное решение

```
# Данные тестового примера 1
a, b = 0, 1                      # Границы отрезка [a,b]
h = 0.01                         # Шаг разбиения
n = int((b - a) / h) + 1         # Количество точек
lambd, eps = 1, 1e-6             # Параметр λ и точность ε

# Модели тестового примера 1
u = lambda x: x**3 - 1/2*x + 14/15     # Точное аналитическое решение
k = lambda x,s: 1 - x*s                # Ядро интегрального уравнения
f = lambda x: x**3                     # Функция правой части
```

Также следует вспомнить про Теорему 16.5. Проверим условие (8):

```
# Проверяем условие единственности решения
k_norm = K_norm(k, a, b)
print(f'‖K‖ = {k_norm:.4f}', '<' if k_norm < 1 else '>', '1')
----------------
||K|| = 0.7817 < 1
```

Получаем значение:

$$\|K\| = \sqrt{\int_0^1 \int_0^1 |1 - xs|^2 \, dxds} = 0.7817 < 1$$

Условие выполнено $\Rightarrow$ решение единственно.
Переходим к поиску решения, вычислению площадей криволинейных трапеций и построению графиков.

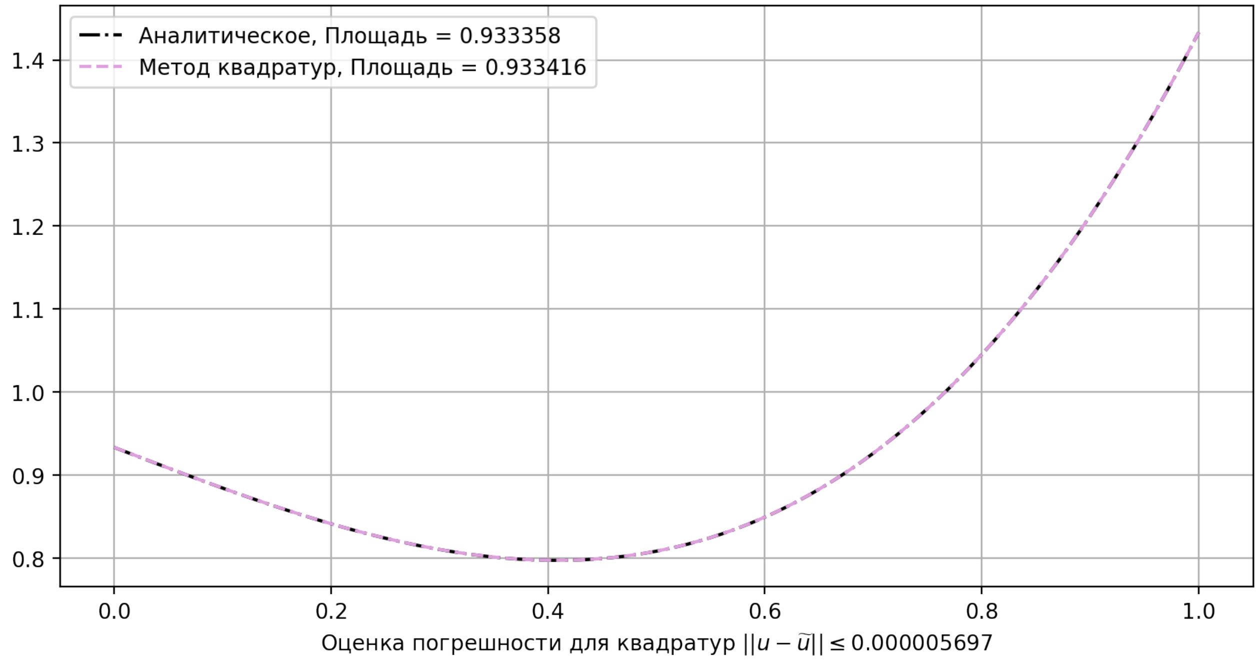
Примечание

Для вычисления площади криволинейной трапеции используется метод trapz из библиотеки numpy. Результат выводится в легенду соответствующего графика. Полученная оценка погрешности так же отображается под графиком.

```python
# Получаем данные для построения графика
x_data, u_data, r_data = fredholm(k, a, b, h, lambd, f)

# Выводим графики решений (аналитическое и численное)
fig, axs = plt.subplots(1, 1, figsize=(10,5), dpi=200)
axs.plot(x_data, u(x_data), color='k', linestyle='-.',
        label=f'Аналитическое, Площадь = {np.trapz(u(x_data), x_data):.6f}')
axs.plot(x_data, u_data, color='plum', linestyle='--',
         label=f'Метод квадратур, Площадь = {np.trapz(u_data, x_data):.6f}')
axs.set_xlabel(r'Оценка погрешности для квадратур $||u-\widetilde{u}|| \leq$'
                                                            + f'{r_data:.9f}')
plt.legend()
plt.grid()
```

Рис. 2: Решение тестового примера 1



4.2 Тестовый пример 2

4.2.1 Аналитическое решение

Составим еще один тестовый пример для валидации решения. Положим:

$$\lambda = 3, \quad K(x,s) = x \sin\left(\pi - \frac{s}{2}\right), \quad a = 0, \quad b = \frac{\pi}{2}, \quad f(x) = \cos(2x)$$

Тогда:

$$u(x) = 3\int_0^{\frac{\pi}{2}} x \sin\left(\pi - \frac{s}{2}\right) u(s) ds + \cos(2x)$$

Распишем уравнение, используя формулу синуса разности:

$$u(x) = 3x\left[\int_0^{\frac{\pi}{2}} \sin(\pi)\cos\left(\frac{s}{2}\right)u(s)ds - \int_0^{\frac{\pi}{2}} \cos(\pi)\sin\left(\frac{s}{2}\right)u(s)ds\right] + \cos(2x)$$

$$u(x) = 3x\left[\int_0^{\frac{\pi}{2}} (0)ds + \int_0^{\frac{\pi}{2}} \sin\left(\frac{s}{2}\right)u(s)ds\right] + \cos(2x)$$

$$u(x) = 3x\underbrace{\int_0^{\frac{\pi}{2}} \sin\left(\frac{s}{2}\right)u(s)ds}_{C_1} + \cos(2x) \tag{25}$$

Запишем решение в виде:

$$u(x) = 3xC_1 + \cos(2x) \tag{26}$$

Заменим x на s в решении (26) и подставим в C_1 из (25):

$$C_1 = \int_0^{\frac{\pi}{2}} \left[3C_1 s\sin\left(\frac{s}{2}\right) + \sin\left(\frac{s}{2}\right)\cos(2s)\right]ds$$

Вычисляя следующий интеграл, получаем:

$$C_1 = \left[-6C_1 s\cos\left(\frac{s}{2}\right) + 12C_1\sin\left(\frac{s}{2}\right) - \frac{1}{5}\cos\left(\frac{5}{2}s\right) + \frac{1}{3}\cos\left(\frac{3}{2}s\right)\right]\Bigg|_0^{\frac{\pi}{2}}$$

$$C_1 = -\frac{3\pi\sqrt{2}}{2}C_1 + 6\sqrt{2}C_1 - \frac{\sqrt{2}+2}{15} \Rightarrow C_1 = \frac{\left[\dfrac{\sqrt{2}+2}{15}\right]}{\left[\dfrac{3\sqrt{2}}{2}(4-\pi) - 1\right]}$$

Подставляя полученное C_1 в решение (26):

$$u(x) = \cos\big(2x\big) + x\cdot\frac{\left[\dfrac{\sqrt{2}+2}{5}\right]}{\left[\dfrac{3\sqrt{2}}{2}(4-\pi) - 1\right]}$$

Получено аналитическое решение. Аналогично, ищем численное решение и анализируем результаты:

4.2.2 Численное решение

```
# Данные тестового примера 2
a, b = 0, np.pi/2                 # Границы отрезка [a,b]
h = 0.01                          # Шаг разбиения
n = int((b - a) / h) + 1          # Количество точек
lambd, eps = 3, 1e-6              # Параметр λ и точность ε

# Модели тестового примера 2                  # Точное аналитическое решение
u = lambda x: np.cos(2*x)+x*((np.sqrt(2)+2)/5)/(3*np.sqrt(2)/2*(4-np.pi)-1)
k = lambda x,s: x*np.sin(np.pi-s/2)          # Ядро интегрального уравнения
f = lambda x: np.cos(2*x)                    # Функция правой части
```

Проверяем условие (8):

```
# Проверяем условие единственности решения
k_norm = K_norm(k, a, b)
print(f'‖K‖ = {k_norm:.4f}', '<' if k_norm < 1 else '>', '1')
----------------
||K|| = 0.6072 < 1
```

Получаем:

$$\|K\| = \sqrt{\int_0^{\frac{\pi}{2}} \int_0^{\frac{\pi}{2}} \left| x \sin\left(\pi - \frac{s}{2}\right) \right|^2 dx ds} = 0.6072 < 1$$

Условие выполнено $\Rightarrow$ решение единственно.

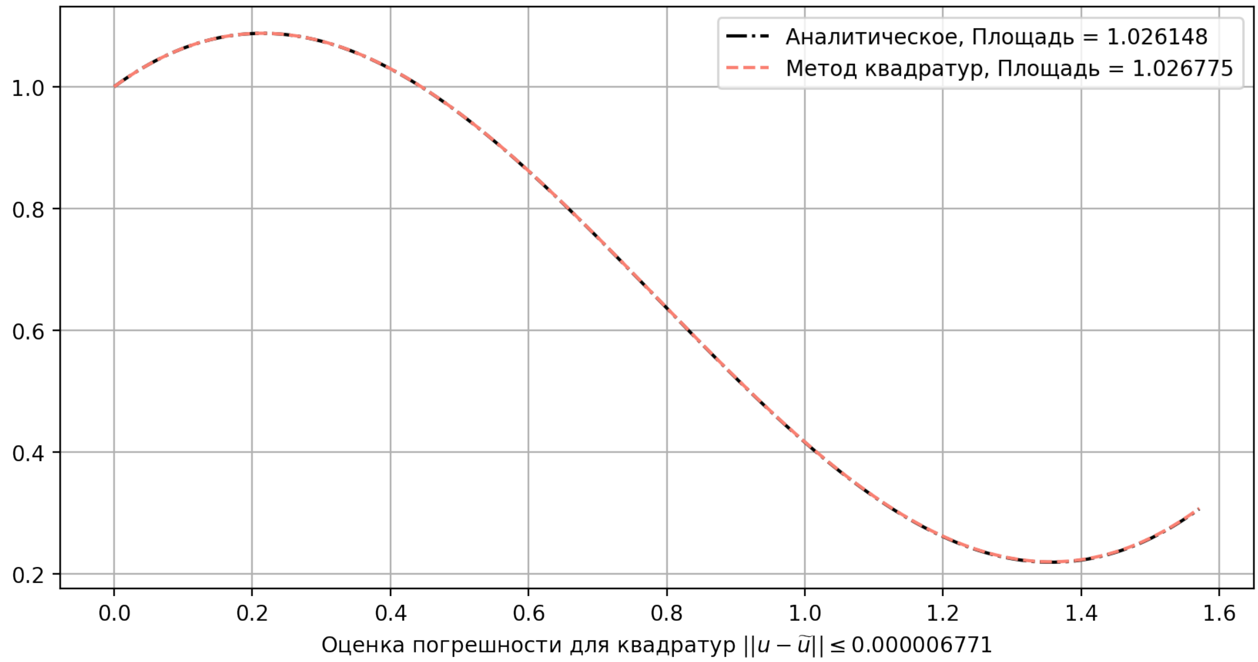
Строим график решения:

```
# Данные для построения графика
x_data, u_data, r_data = fredholm(k, a, b, h, lambd, f)

# Выводим графики решений
fig, axs = plt.subplots(1, 1, figsize=(10,5), dpi=200)
axs.plot(x_data, u(x_data), color='k', linestyle='-.',
         label=f'Аналитическое, Площадь = {np.trapz(u(x_data), x_data):.6f}')
axs.plot(x_data, u_data, color='salmon', linestyle='--',
         label=f'Метод квадратур, Площадь = {np.trapz(u_data, x_data):.6f}')
```

```
axs.set_xlabel(r'Оценка погрешности для квадратур $||u-\widetilde{u}|| \leq$'
                                                        + f'{r_data:.9f}')
plt.legend()
plt.grid()
```

Рис. 3: Решение тестового примера 2



Результат

Графики совпадают и накладываются друг на друга, а площади трапеций примерно одинаковые. Погрешность, полученная с помощью нормы невязки, мала. Все это свидетельствует о верной работе алгоритма. Большей точности можно добиться, уменьшая шаг разбиения, тем самым увеличивая число точек.

4.3 Исходный пример

Имеем уравнение:

$$u(x) - \lambda \int_1^2 e^{-|x - s|} u(s) ds = f(x), \quad x \in [1,2]$$

Определим начальные данные и найдем решение задачи:

```
# Ядро интегрального уравнения
K = lambda x,s: np.exp(-1 * np.abs(x - s))
```

```python
# Начальные данные
a, b = 1, 2                  # Границы отрезка [a,b]
h = 0.01                     # Шаг разбиения
n = int((b - a) / h) + 1     # Количество точек
lambd, eps = 1, 1e-6         # Параметр λ и точность ε
alpha, beta = 0.2, 0.7       # Границы отрезка [α,β]
```

Снова обращаемся к Теореме 16.5 и проверяем условие (8):

```python
# Проверяем условие единственности решения
k_norm = K_norm(K, a, b)
print(f'‖K‖ = {k_norm:.4f}', '<' if k_norm < 1 else '>', '1')
```

```
----------------
||K|| = 0.7534 < 1
```

Получаем, что:

$$\|K\| = \sqrt{\int_1^2 \int_1^2 \left|e^{|x-s|}\right|^2 dxds} = 0.7534 < 1$$

Условие выполнено $\Rightarrow$ решение единственно.

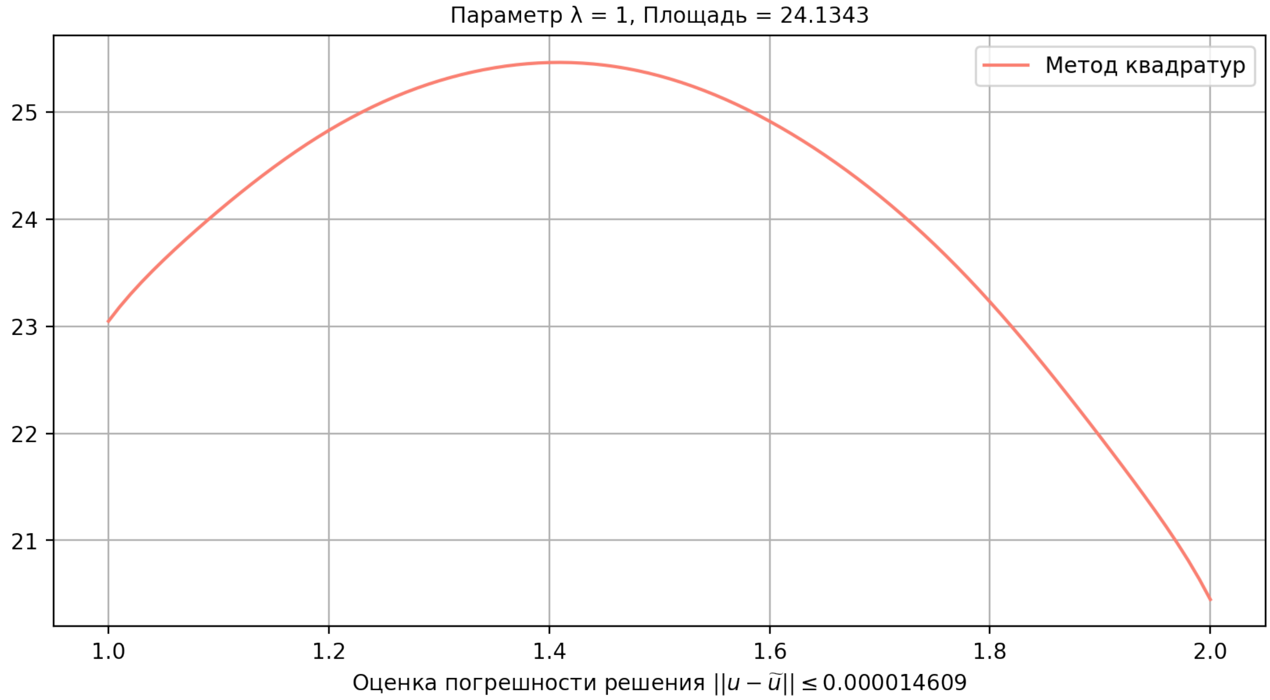
4.3.1 Решение при $\lambda = 1$

Используя написанный алгоритм, находим решение исходной задачи при $\lambda = 1$ и строим его график.

```python
# Выводим график решения поставленной задачи
fig, axs = plt.subplots(1, 1, figsize=(10,5), dpi=200)
# Находим решение интегрального уравнения
x_data, u_data, r_data = fredholm(K, a, b, h)
# Вычисляем площадь криволинейной трапеции
area = np.trapz(u_data, x_data)
axs.plot(x_data, u_data, color='salmon', label='Метод квадратур')
axs.set_title(f'Параметр λ = {lambd}, Площадь = {area:.4f}', fontsize=10)
axs.set_xlabel(r'Оценка погрешности решения $||u-\widetilde{u}|| \leq$'
                                                       + f'{r_data:.9f}')

plt.legend()
plt.grid()
```

Рис. 4: Решение поставленной задачи при $\lambda = 1$



4.3.2 Решение для каждого λ из указанного отрезка

Выполним аналогичные действия для значений λ из указанного диапазона $[\alpha,\beta] = [0.2,0.7]$.

```
# Зададим для отрезка [α,β] шаг 0.1 (получаем 6 значений λ)
lambdas = np.linspace(alpha, beta, 6)

# Кортежи стилей и цветов для графиков
styles = ('-.', '--', '-')
colors = ('g', 'c', 'orange', 'salmon', 'teal', 'plum')

cycler = cycle(styles)
lambdas = np.reshape(lambdas, (-1, 2))
colors = np.reshape(colors, (-1, 2))
# Список для площадей криволинейных трапеций
areas = []

# Выводим графики решения при разных значениях λ
fig, axs = plt.subplots(3, 2, figsize=(15,20), dpi=100)
for i in range(lambdas.shape[0]):
    for j in range(lambdas.shape[1]):
        # Находим решение интегрального уравнения
        x_data, u_data, r_data = fredholm(K, a, b, h, lambdas[i][j])
        # Добавляем в список текущую вычисленную площадь
```

```
        areas.append(np.trapz(u_data, x_data))
        axs[i][j].plot(x_data, u_data, color=colors[i][j],
        linestyle=next(cycler), label=f'Параметр λ = {lambdas[i][j]}')
        axs[i][j].set_xlabel(r'$||u-\widetilde{u}|| \leq$' + f'{r_data:.9f}')
        axs[i][j].set_title(f'Площадь = {areas[-1]:.4f}')
        axs[i][j].legend()
        axs[i][j].grid()

lambdas = np.reshape(lambdas, -1)
colors = np.reshape(colors, -1)

# Ищем значение параметра, при котором достигается наибольшая площадь
print('Максимальное значение:',
 f'Параметр λ = {lambdas[areas.index(max(areas))]},
                Площадь = {max(areas):.4f}.', sep='\n')
------------------------------------
Максимальное значение:
Параметр λ = 0.7, Площадь = 13.0713.
```

Рис. 5: Решения при различных λ (на одном чертеже)

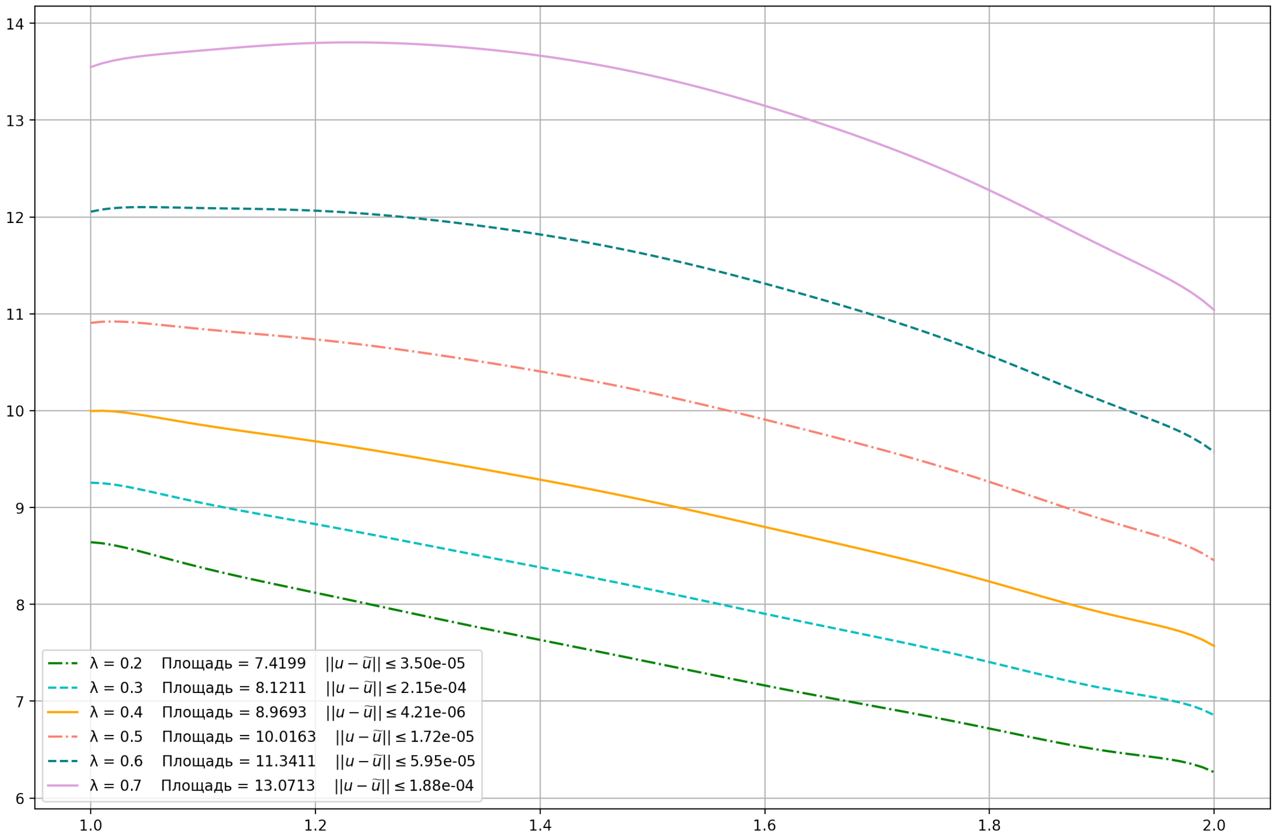
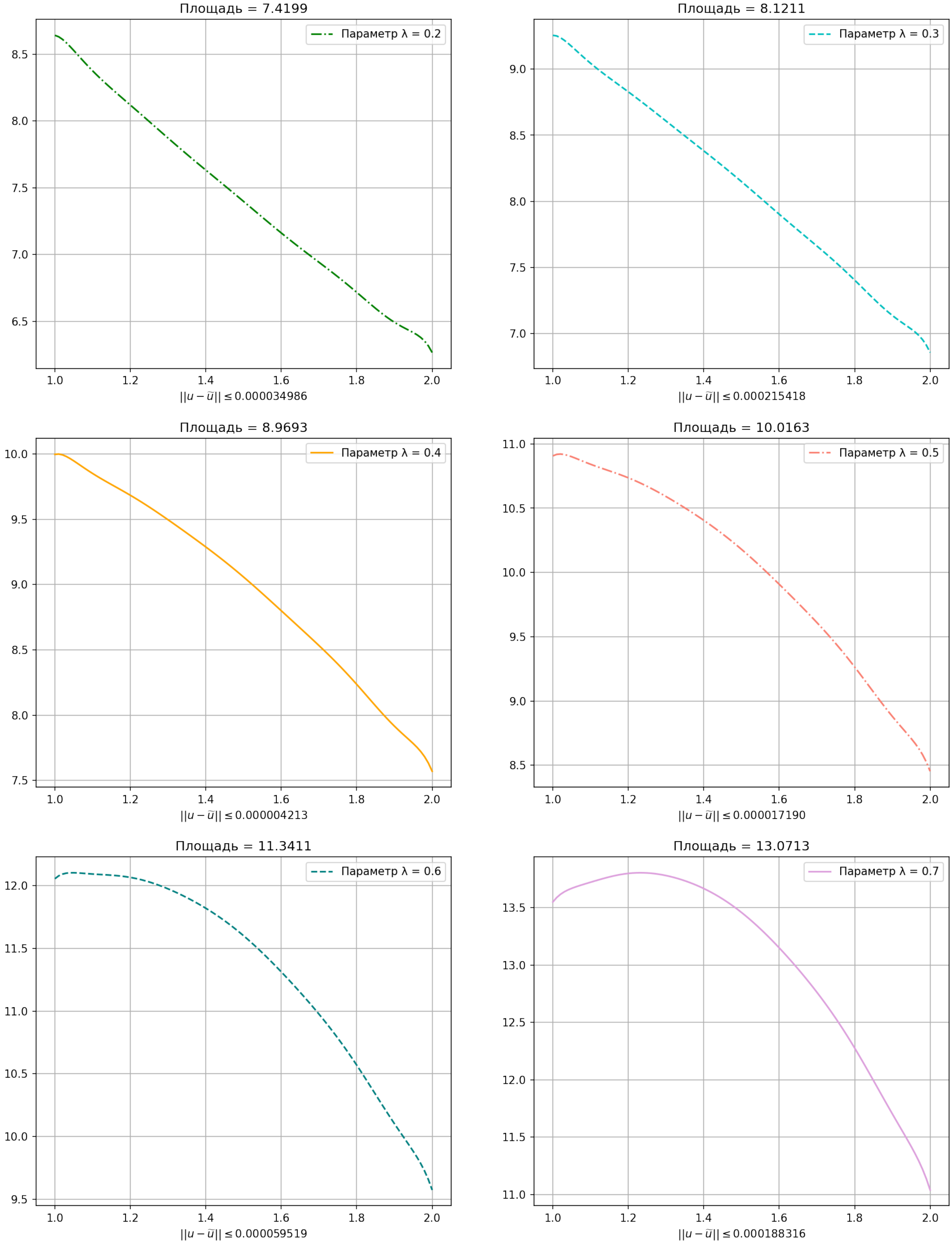


Рис. 6: Решения при различных λ (индивидуально)



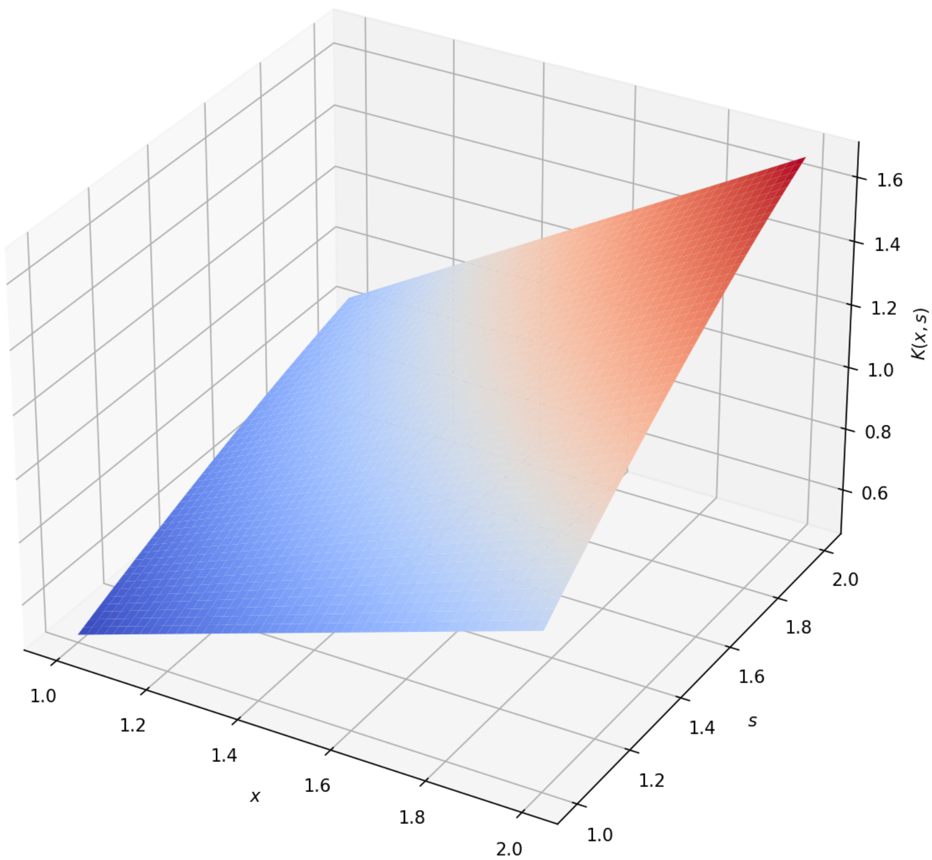
5 Вывод

Используемые методы и корректно построенный алгоритм, позволили решить задачу с хорошей точностью, о чем свидетельствуют вычислительные эксперименты.

6 Приложение

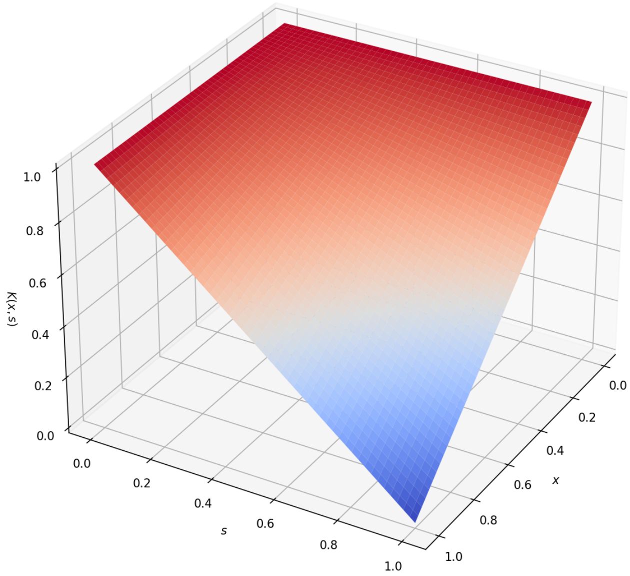
Дополнительные иллюстрации:

Рис. 7: Ядро $K(x,s)$ исходного примера



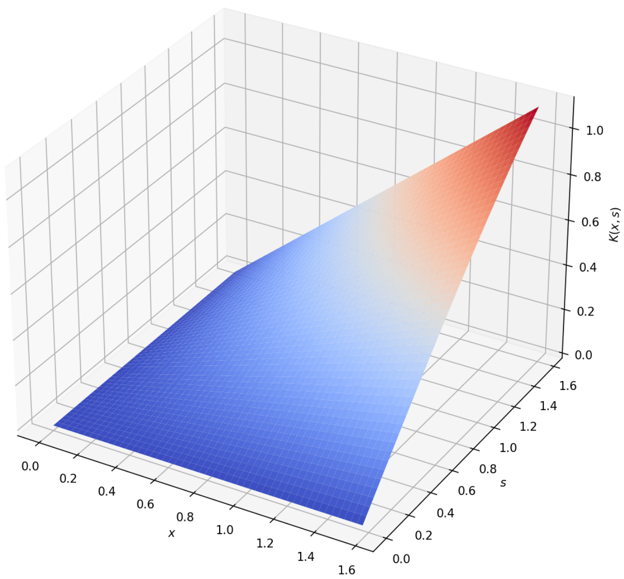
$$K(x,s) = e^{-|x-s|} \qquad [a,b] = [1,2]$$

Рис. 8: Ядро $K(x,s)$ тестового примера 1



$$K(x,s) = 1 - xs \qquad [a,b] = [0,1]$$

Рис. 9: Ядро $K(x,s)$ тестового примера 2



$$K(x,s) = x \sin\left(\pi - \frac{s}{2}\right) \qquad [a,b] = \left[0,\frac{\pi}{2}\right]$$

Список литературы

[1] Амосов А.А. *Вычислительные методы* [Текст]: Учебное пособие / А.А.Амосов, Ю.А.Дубинский, Н.В.Копченова – Изд. 2-е, стер. – М.: Лань, 2014. – Гл.6: § 6.5; Гл.11: § 11.3; Гл.13: § 13.1; Гл.16: § 16.3, 16.4.

[2] Карчевский Е.М. *Численные методы решения интегральных уравнений* [Текст]: Учебное пособие / Е.М.Карчевский – Казань, 2019. – Гл.2 § 2.1.

[3] *Метод сопряженных градиентов* [Электронный ресурс]: Википедия. Свободная энциклопедия. – Режим доступа: *Ссылка*

[4] *Метод сопряженных градиентов* [Электронный ресурс]: Machine Learning. Профессиональный информационно-аналитический ресурс, посвященный машинному обучению и интеллектуальному анализу данных. – Режим доступа: *Ссылка*

LaTeX