Московский авиационный институт

(Национальный исследовательский университет)

Факультет прикладной математики и физики

Кафедра вычислительной математики и программирования

Курсовая работа по курсу

«Численные методы»

На тему: «Численное решение интегральных уравнений Вольтерра 2-го рода»

Вариант 14

Выполнил: Муртазин Р.Ю.

Группа: М8О-409Б-20

Проверил: Пивоваров Д.Е.

Дата:

Оценка:

Москва, 2023

ОГЛАВЛЕНИЕ

Теоретические сведения

Линейное уравнение Вольтерра II рода имеет следующий вид:

$$y(x) - \int\limits_a^x K(x,s)y(s)ds = f(x), \quad x \in [a,b].$$

Здесь y(x) — неизвестная функция, K(x, s) — ядро интегрального уравнения, f(x) — свободный член (правая часть) интегрального уравнения. Однородное уравнение (при f ≡ 0) имеет только тривиальное решение, а условия существования решения неоднородного уравнения связаны с различными ограничениями на ядро K(x, s) и правую часть f(x) В частности , решение существует и единственно в классе непрерывных на отрезке [a, b] функций, если ядро непрерывно внутри и на сторонах треугольника, ограниченного прямыми s = a, x = b, x = s, а функция f(x) непрерывна на [a, b].

Далее рассмотрим несколько методов численного решения уравнений Вольтерра II рода.

1. Метод квадратур

При численном решении интегральных уравнений входящие в них интегралы обычно заменяют конечными суммами. Согласно методу квадратур интегральные операторы заменяют суммами, полученными с помощью различных квадратурных формул

$$\int\limits_a^b g(x)dx = \sum_{i=1}^n A_i g(x_i) + R.$$

Здесь $a \le x_1 < x_2 < \cdot \cdot \cdot < x_n \le b$ — узлы, A_i , $i = 1, 2, . . . , n$ — веса, а R — ошибка аппроксимации квадратурной формулы. Чтобы применить метод квадратур к решению уравнения, необходимо использовать следующие равенства:

$$y(x_i) - \int_a^{x_i} K(x_i, s)y(s)ds = f(x_i), \quad i = 1,\, 2,\, ...,\, n.$$

Они получаются из исходного уравнения при фиксированных значениях xi независимой переменной x. Узлы сетки x_i могут быть выбраны специальным образом или заданы заранее, если, например, правая часть f задана таблицей. Примем значения x_i в качестве узлов квадратурной формулы и заменим с ее помощью интеграл в конечной суммой. Получим систему:

$$y_i - \sum_{j=1}^{i} A_j K_{ij}\, y_j = f_i, \quad i = 1,\, 2,\, ...,\, n.$$

Достаточно простым и во многих случаях эффективным является применение формулы трапеций. Для равномерной сетки с шагом h имеем:

$$A_1 = A_n = \frac{h}{2}, \quad A_2 = A_3 = \ldots = A_{n-1} = h.$$

Тогда формула примет следующий вид:

$$y_i = \left(1 - \frac{h}{2}K_{ii}\right)^{-1} \left(f_i + \frac{h}{2}K_{i1}y_j + h\sum_{j=2}^{i-1} K_{ij}y_j\right), \quad i = 1,\, 2,\, ...,\, n.$$

2. Метод простой итерации

Запишем линейное уравнение Вольтерра II рода в удобном для применения метода простой итерации виде:

$$y(x) = f(x) + \int_a^{x} K(x, s)y(s)ds, \quad x \in [a, b].$$

Построим последовательность функций $y_k(x)$, $k = 0, 1, 2, \ldots$, с помощью рекуррентного соотношения:

$$y_k(x) = f(x) + \int\limits_a^x K(x,s) y_{k-1}(s) ds, \quad k = 1, 2, \ldots$$

Если правая часть f(x) непрерывна на отрезке [a, b], а ядро K(x, s) непрерывно в замкнутом треугольнике $a \le s \le x \le b$, эта последовательность сходится при любом начальном приближении $y_0(x)$. Скорость сходимости зависит от свойств ядра и правой части уравнения. Ясно, что число итерационных шагов для получения аппроксимации необходимой точности зависит от степени близости начального приближения к искомому решению. В качестве начального приближения часто выбирают f(x), если нет дополнительной информации о решении.

При численной реализации итерационных методов интеграл вычисляется посредством квадратурных формул. Воспользуемся квадратурной формулой трапеций с равномерной сеткой и шагом h.

Узлы сетки обозначим x_i, $i = 0, 1, \ldots, n$. Пусть $K_{ij} = K(x_i, x_j)$, $y_{ki} = y_k(x_i)$. Получим расчетное выражение:

$$y_{k+1}(x_i) = f(x_i) + \int\limits_0^{x_i} K(x_i, s) y_k(s) ds \approx$$

$$\approx f(x_i) + \frac{h}{2}\left[K_{i0}y_{k0} + 2\left(K_{i1}y_{k1} + K_{i2}y_{k2} + \cdots + K_{i,i-1}y_{k,i-1}\right) + K_{ii}y_{ki}\right],$$

где $i = 0, 1, \ldots, n$. Для окончания итерационного процесса, будем использовать условие:

$$\frac{||y_k - y_{k-1}||}{||y_k||} \leqslant \varepsilon,$$

где $||y|| = \max |y(x)|$ на отрезке [a,b], ε — заданная относительная ошибка. Данное условие означает, что в процессе решения необходимо сравнивать результаты, полученные для двух смежных итерационных шагов; близость полученных при этом приближений свидетельствует о достигнутой точности. Таким образом, количество итерационных шагов зависит также от требований к точности результата.

Реализация методов на языке Python

```
import numpy as np, matplotlib.pyplot as plt


def K(x, s):
    return np.e**(s - x)


def f(x):
    return np.e**(-x)


def accurate_y(x):
    return 1


def quadrature_method(a, b, h):
    x = np.arange(a, b + h, h)
    y = np.zeros(len(x))
    y[0] = f(x[0])
    for i in range(1, len(x)):
        sum = 0
        if i > 1:
            for j in range(1, i):
                sum = K(x[i], x[j]) * y[j] + sum
        y[i] = ((1 - h / 2 * K(x[i], x[i]))**(-
1)) * ((f(x[i])) + (h / 2) * K(x[i], x[0]) * y[0] + h * sum)
    return y


def norm_vector(b):
    norm = 0
    for i in range(len(b)):
        norm += b[i]*b[i]
    return np.sqrt(norm)


def count_yk(x, y):
    yk = np.zeros(len(y))
    for i in range(len(yk)):
        for j in range(0, i + 1):
            yk[i] = yk[i] + 2 * K(x[i], x[j]) * y[j]
        yk[i] = yk[i] - K(x[i], x[0]) * y[0] - K(x[i], x[i]) * y[i]
        yk[i] = f(x[i]) + yk[i] * h / 2
```

```
    return yk


def simple_iteration(a, b, h, e):
    x = np.arange(a, b + h, h)
    y = f(x)
    yk = count_yk(x, y)
    while (norm_vector(yk - y) / norm_vector(yk)) > e:
        y = yk
        yk = count_yk(x, y)
    return y


def main(a, b, h, e):
    x = np.arange(a, b + h, h)
    y = np.zeros(len(x))
    for i in range(len(y)):
        y[i] = accurate_y(x[i])
    q_y = quadrature_method(a, b, h)
    si_y = simple_iteration(a, b, h, e)
    print("Решения интегрального уравнения Вольтерра второго рода \n")
    print("             X           ", x)
    print("   Точное значение Y:    ", y)
    print("    Метод квадратур:     ", *np.around(q_y, 4))
    print("Метод простой итерации: ", *np.around(si_y, 4), "\n")
    print("Разница от точного значения")
    print("    Метод квадратур:     ", *np.around(q_y - y, 4))
    print("Метод простой итерации: ", *np.around(si_y - y, 4))
    plt.title("График точного решения функции и приближенного с помощью методов")
    plt.xlabel("x")
    plt.ylabel("y")
    plt.grid()
    plt.axis([-0.1, 1.1, 0.999, 1.001])
    plt.plot(x, y, label = "Точное решение")
    for i in range(len(x)):
        plt.scatter(x[i], q_y[i])
    plt.plot(x,q_y, label = "Значения Y методом квадратур")
    for i in range(len(x)):
        plt.scatter(x[i], si_y[i])
    plt.plot(x,si_y, label = "Значения Y методом простых итераций")
    plt.legend()
    plt.show()


a = 0
b = 2
h = 0.1
e = 0.0001
main(a, b, h, e)
```

Код приведён для численного решения следующего уравнения Вольтерра II рода

$$y(x) - \int\limits_0^x e^{-(x-s)} y(s) ds = e^{-x}, \quad x \in [0, 1].$$

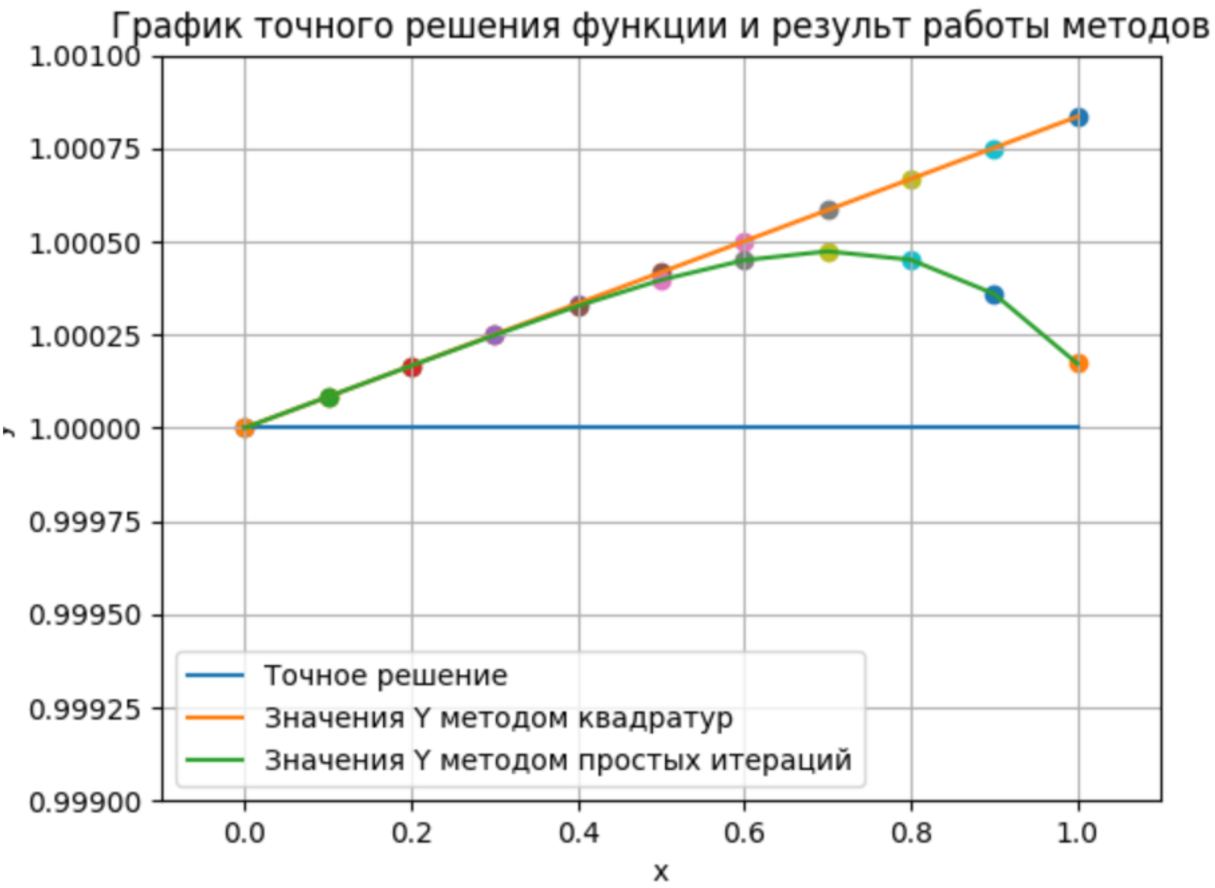
Точное решение этого уравнения y(x) = 1

Результаты тестов:

```
Решения интегрального уравнения Вольтерра второго рода

            X           [0.  0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1. ]
   Точное значение Y:   [1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]
    Метод квадратур:    1.0 1.0001 1.0002 1.0003 1.0003 1.0004 1.0005 1.0006 1.0007 1.0008 1.0008
Метод простой итерации:  1.0 1.0001 1.0002 1.0002 1.0003 1.0004 1.0004 1.0005 1.0005 1.0004 1.0002

Разница от точного значения
    Метод квадратур:    0.0 0.0001 0.0002 0.0003 0.0003 0.0004 0.0005 0.0006 0.0007 0.0008 0.0008
Метод простой итерации:  0.0 0.0001 0.0002 0.0002 0.0003 0.0004 0.0004 0.0005 0.0005 0.0004 0.0002
```



Приведём для примера ещё одно уравнение Вольтерра II рода

$$y(x) = 1 + \int_0^x y(s)ds, \qquad x \in [0,2]$$

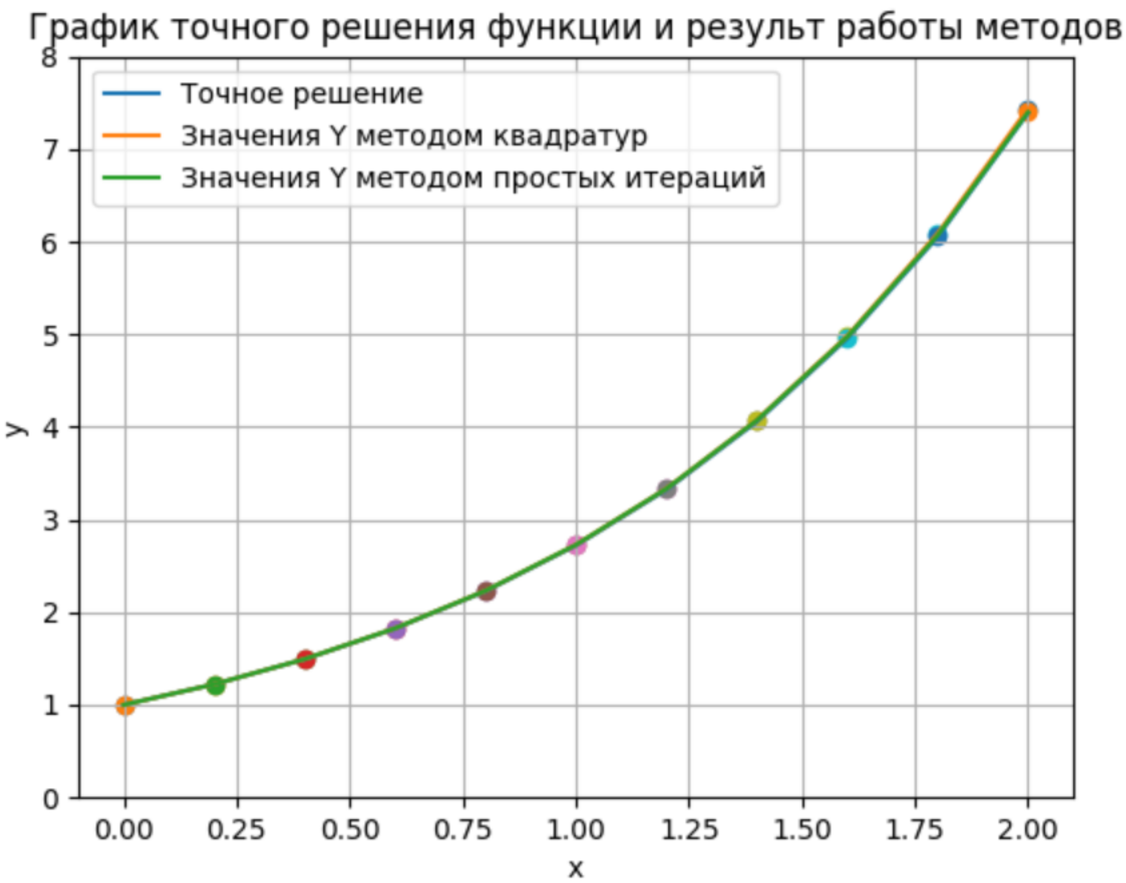
Точное решение этого уравнения $y(x) = e^x$

Результаты тестов:

```
Решения интегрального уравнения Вольтерра второго рода

            X            [0.  0.2 0.4 0.6 0.8 1.  1.2 1.4 1.6 1.8 2. ]
   Точное значение Y:    [1.         1.22140276 1.4918247  1.8221188  2.22554093 2.71828183
 3.32011692 4.05519997 4.95303242 6.04964746 7.3890561 ]
    Метод квадратур:     1.0 1.2222 1.4938 1.8258 2.2315 2.7274 3.3335 4.0743 4.9797 6.0863 7.4388
Метод простой итерации:  1.0 1.2222 1.4938 1.8258 2.2314 2.727 3.3322 4.0707 4.9708 6.0666 7.3979

Разница от точного значения
    Метод квадратур:     0.0 0.0008 0.002 0.0037 0.006 0.0091 0.0134 0.0191 0.0266 0.0366 0.0497
Метод простой итерации:  0.0 0.0008 0.002 0.0036 0.0059 0.0087 0.0121 0.0155 0.0178 0.0169 0.0089
```



Вывод

При решении уравнений Вольтерра II рода чаще всего используются два этих метода. Однако метод квадратур с каждым шагом выдаёт всё большую погрешность в вычислениях, в отличие от метода простых итераций. Это связано с тем, что замена интеграла квадратурной формулой, влечёт за собой слагаемое ошибки аппроксимации квадратурной формулы. Для работы алгоритма, это слагаемое считают достаточно маленьким, чтобы принебречь им, однака при каждой новой итерации – это слагаемое возрастает всё больше и больше. Поэтому погрешность в методе квадратур возрастает непрерывно. В методе простых итераций тоже используются квадратурные формулы, однако задаваемое точность в этом итерационном методе позволяет уменьшить влияние ошибки апроксимации на конечный результат, из-за чего погрешность скачет в определённых пределах, в чём можно убедиться в вышеизложенных тестах.