|  |  |
| --- | --- |
|  | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ Информатики и систем управления

КАФЕДРА Теоретической информатики и компьютерных технологий

**Лабораторная работа № 4**

«Сравнительный анализ методов численного решения краевой задачи для линейного дифференциального уравнения второго порядка»

по курсу «Численные методы»

Выполнила:

студент группы ИУ9-61Б

Яровикова Анастасия

Проверила:

Домрачева А. Б.

Москва, 2023

1. **Цель**

Целью данной работы является сравнение по точности решения двух методов численного решения краевой задачи для линейного дифференциального уравнения второго порядка:

1. Метод прогонки

2. Метод стрельбы

1. **Постановка задачи**

**Дано:** краевая задача для линейного дифференциального уравнения (ДУ) второго порядка

**Задание:**

* Найти аналитическое решение задачи Коши:

;

* По найденному решению задачи Коши вычислить ;
* С помощью метода прогонки и метода стрельбы найти численное решение краевой задачи для того же уравнения с краевыми условиями ;
* Для каждого метода вычислить , найти погрешность численного решения и сравнить (здесь – аналитическое решение, – численное решение).

**Индивидуальный вариант:**

Краевая задача имеет вид:

1. **Основные теоретические сведения**

**Метод прогонки**

Пусть требуется решить краевую задачу на отрезке (т.е. краевые условия ДУ заданы в точках ). Тогда отрезок разбивается на равных отрезков длины . Получаем разбиение отрезка точками .

Приближенный численным решением краевой задачи для ДУ второго порядка называется сеточная функция , заданная в точках .

Обозначим значения коэффициентов уравнения в точках через . При помощи разностной аппроксимации производных получаем приближенную систему уравнений относительно :

После преобразования система имеет вид:

с краевыми условиями

Данная система имеет порядок и представляет собой трехдиагональную систему линейных алгебраических уравнений, ее необходимо решить методом прогонки.

Напомним, что метод прогонки позволяет решать системы вида , где - трехдиагональная матрица:

где – массив элементов под главной диагональю, – массив элементов главной диагонали, – массив элементов над главной диагональю.

Для рассматриваемой задачи элементы массивов будут иметь вид:

Поскольку , то

**Метод стрельбы**

Пусть требуется решить краевую задачу на отрезке (т.е. краевые условия ДУ заданы в точках ). Тогда отрезок разбивается на равных отрезков длины . Получаем разбиение отрезка точками .

Приближенный численным решением краевой задачи для ДУ второго порядка называется сеточная функция , заданная в точках .

При помощи разностной аппроксимации производных получаем формулы для производных :

Обозначим значения , а значения коэффициентов уравнения в точках через .

Ищем решения , удовлетворяющие условиям:

где . За возьмем значение .

Для определения получаем уравнения:

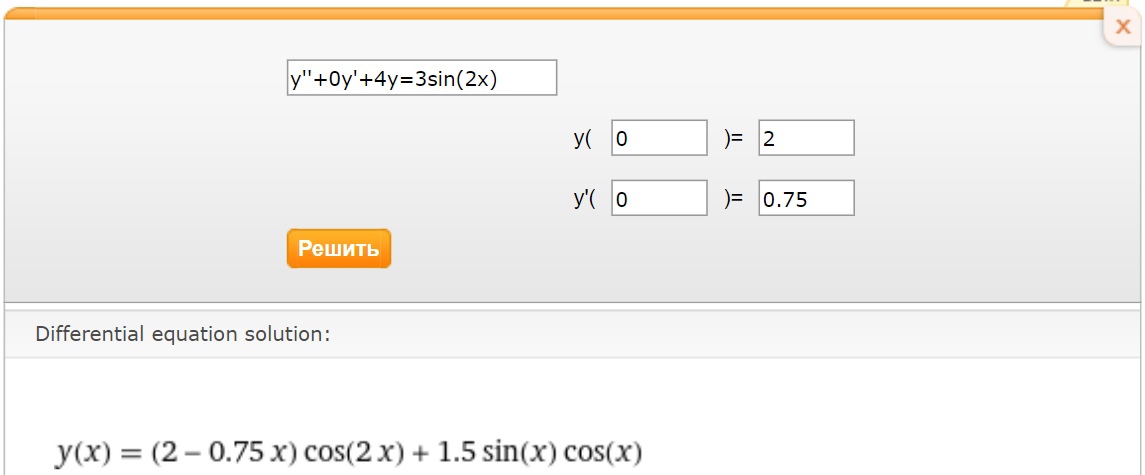
Используя условия , имеем:

Далее последовательно определяем и находим .

Искомое решение ищется по формулам:

1. **Реализация**

Аналитическое решение для задачи Коши найдено с помощью WolframAlpha:



По вычисленному решению находим

Листинг 1. Метод прогонки для решения краевой задачи для ДУ второго порядка

package main

import (

"fmt"

"math"

)

func direct(b, a, c, d []float64, size int) (alpha, beta []float64) {

alpha = append(alpha, -c[0] / b[0])

beta = append(beta, d[0] / b[0])

var y float64

for i := 1; i < size - 1; i++ {

y = a[i - 1] \* alpha[i - 1] + b[i]

alpha = append(alpha, -c[i] / y)

beta = append(beta, (d[i] - a[i - 1] \* beta[i - 1]) / y)

}

y = a[size - 2] \* alpha[size - 2] + b[size - 1]

beta = append(beta, (d[size - 1] - a[size - 2] \* beta[size - 2]) / y)

return alpha, beta

}

func reverse(alpha, beta []float64, size int) (x []float64) {

x = make([]float64, size)

x[size - 1] = beta[size - 1]

for i := size - 2; i >= 0; i-- {

x[i] = alpha[i] \* x[i + 1] + beta[i]

}

return x

}

func f(x float64) float64 {

return 3 \* math.Sin(2 \* x)

}

func analytical(x float64) float64 {

return (2 - 0.75 \* x) \* math.Cos(2 \* x) + 1.5 \* math.Sin(x) \* math.Cos(x)

}

var (

n = 10

p = 0.0

q = 4.0

a = analytical(0)

b = analytical(1)

)

func main() {

fmt.Println("МЕТОД ПРОГОКНИ")

fmt.Printf("y'' + %0.1fy' + %0.1fy = (2 - 0.75\*x)\*cos(2x) + 1.5\*sin(x)\*cos(x)\n", p, q)

fmt.Printf("y(0) = %f\ny(1) = %f\n", a, b)

h := 1.0 / float64(n)

xs := make([]float64, 0, n)

for i := 0; i < n + 1; i++ {

xs = append(xs, float64(i) \* h)

}

as := make([]float64, 0, n - 2)

bs := make([]float64, 0, n - 1)

cs := make([]float64, 0, n - 2)

ds := make([]float64, 0, n -1)

for i := 1; i < n - 1; i++ {

as = append(as, 1 - h / 2 \*p)

cs = append(cs, 1 + h / 2 \* p)

}

for i := 1; i < n; i++ {

bs = append(bs, h \* h \* q - 2)

}

ds = append(ds, h \* h \* f(0) - a \* (1 - h / 2 \* p))

for i := 2; i < n; i++ {

ds = append(ds, h \* h \* f(float64(i) \* h))

}

ds[len(ds) - 1] = h \* h \* f(float64(len(ds) - 1) \* h) - b \* (1 + h / 2 \* p)

alpha, beta := direct(bs, as, cs, ds, len(ds))

ys := []float64{a}

ys = append(ys, reverse(alpha, beta, len(ds))...)

ys = append(ys, b)

maxInaccuracy := 0.0

for i := 0; i < len(ys); i++ {

fmt.Printf("x=%.1f, y=%.6f, y\*=%.6f |y-y\*|=%.6f\n",

float64(i) \* h, analytical(xs[i]), ys[i], math.Abs(ys[i] - analytical(xs[i])))

if math.Abs(ys[i] - analytical(xs[i])) > maxInaccuracy {

maxInaccuracy = math.Abs(ys[i] - analytical(xs[i]))

}

}

fmt.Printf("||y-y\*||=%.6f\n", maxInaccuracy)

}

Листинг 2. Метод стрельбы для решения краевой задачи для ДУ второго порядка

package main

package main

import (

"fmt"

"math"

)

func f(x float64) float64 {

return 3 \* math.Sin(2 \* x)

}

func analytical(x float64) float64 {

return (2 - 0.75 \* x) \* math.Cos(2 \* x) + 1.5 \* math.Sin(x) \* math.Cos(x)

}

var (

n = 10

p = 0.0

q = 4.0

a = analytical(0)

b = analytical(1)

ys = make([][]float64, 2)

)

func getC1() float64 {

return (b - ys[0][n]) / ys[1][n]

}

func getYi(i int) float64 {

return ys[0][i] + getC1() \* ys[1][i]

}

func main() {

fmt.Println("МЕТОД СТРЕЛЬБЫ")

fmt.Printf("y'' + %0.1fy' + %0.1fy = (2 - 0.75\*x)\*cos(2x) + 1.5\*sin(x)\*cos(x)\n", p, q)

fmt.Printf("y(0) = %f\ny(1) = %f\n", a, b)

fmt.Printf("Количество разбиений: %d\n", n)

h := 1.0 / float64(n)

fmt.Println(h)

delta := h

xs := make([]float64, 0, n + 1)

for i := 0; i <= n; i++ {

xs = append(xs, float64(i) \* h)

}

for i := 0; i < 2; i++ {

ys[i] = make([]float64, 2, n)

}

ys[0][0], ys[0][1] = a, a + delta

ys[1][0], ys[1][1] = 0, delta

for i := 1; i < n; i++ {

ys[0] = append(ys[0],

(h \* h \* f(xs[i]) + (2.0 - q \* h \* h) \* ys[0][i] - (1.0 - h / 2 \* p) \* ys[0][i - 1]) / (1 + h / 2 \* p))

ys[1] = append(ys[1],

((2.0 - q \* h \* h) \* ys[1][i] - (1.0 - h / 2 \* p) \* ys[1][i - 1]) / (1 + h / 2 \* p))

}

y := make([]float64, 0, n + 1)

for i := 0; i <= n; i++ {

y = append(y, getYi(i))

}

maxInaccuracy := 0.0

for i := range y {

fmt.Printf("x=%.1f, y=%.6f, y\*=%.6f |y-y\*|=%.6f\n",

float64(i) \* h, analytical(xs[i]), y[i], math.Abs(y[i] - analytical(xs[i])))

if math.Abs(y[i] - analytical(xs[i])) > maxInaccuracy {

maxInaccuracy = math.Abs(y[i] - analytical(xs[i]))

}

}

fmt.Printf("||y-y\*||=%.6f\n", maxInaccuracy)

}

1. **Результаты**

Таблица 1 - Результаты метода прогонки

|  |  |  |  |
| --- | --- | --- | --- |
| Значение | Значение  (аналитическое решение) | Значение  (численное решение) |  |
| 0 | 2.000000 | 2.000000 | 0 |
| 0.1 | 2.035630 | 2.035760 | 0.000130 |
| 0.2 | 1.996027 | 1.996174 | 0.000147 |
| 0.3 | 1.888453 | 1.888599 | 0.000147 |
| 0.4 | 1.722418 | 1.722550 | 0.000132 |
| 0.5 | 1.509094 | 1.509200 | 0.000106 |
| 0.6 | 1.260684 | 1.260758 | 0.000074 |
| 0.7 | 0.989789 | 0.989829 | 0.000040 |
| 0.8 | 0.708801 | 0.708810 | 0.000009 |
| 0.9 | 0.429343 | 0.429326 | 0.000016 |
| 1.0 | 0.161790 | 0.161790 | 0 |

Таблица 2 - Результаты метода стрельбы

|  |  |  |  |
| --- | --- | --- | --- |
| Значение | Значение  (аналитическое решение) | Значение  (численное решение) |  |
| 0 | 2.000000 | 2.000000 | 0 |
| 0.1 | 2.035630 | 2.036225 | 0.000595 |
| 0.2 | 1.996027 | 1.996961 | 0.000934 |
| 0.3 | 1.888453 | 1.889501 | 0.001049 |
| 0.4 | 1.722418 | 1.723401 | 0.000982 |
| 0.5 | 1.509094 | 1.509885 | 0.000790 |
| 0.6 | 1.260684 | 1.261217 | 0.000534 |
| 0.7 | 0.989789 | 0.990063 | 0.000274 |
| 0.8 | 0.708801 | 0.708869 | 0.000068 |
| 0.9 | 0.429343 | 0.429308 | 0.000035 |
| 1.0 | 0.161790 | 0.161790 | 0 |

1. **Вывод**

В ходе выполнения лабораторной работы былы реализованы методы приближенного численного решения краевой задачи для дифференциального уравнения второго порядка: метод прогонки и метод стрельбы.

Вследствие сравнения результатов работы методов сделан вывод о более высокой точности метода прогонки в отличие от метода стрельбы. Такой результат, прежде всего, связан с присутствием случайных значений в методе стрельбы (), в то время как в методе прогонки их нет. Кроме того вычислительная погрешность обусловлена малым количеством разбиений рассматриваемого отрезка. С увеличением числа разбиений, погрешность уменьшается.