

НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Механико-математический факультет
Кафедра: Математика и компьютерные науки

Тлепбергенова Дарья Дулатовна

ОТЧЕТ ПО ВЫЧИСЛИТЕЛЬНОМУ ПРАКТИКУМУ

Решение уравнения переноса
с помощью неявной схемы бегущего счета.
Вариант 12.

3 курс, группа 16121

Преподаватель:
Махоткин Олег Александрович

Новосибирск, 2018 г.

1 Постановка задачи.

Дано уравнение переноса в виде:

$$\begin{cases} \frac{\partial u}{\partial t} + C(x, t) \frac{\partial u}{\partial x} = 0, \\ 0 \leq x \leq 1, 0 \leq t \leq 1 \\ u(x, 0) = \varphi_0(x) \end{cases} \quad (1)$$

С помощью неявной схемы бегущего счета:

$$\frac{u_k^{j+1} - u_k^j}{\tau} + C \frac{u_k^{j+1} - u_{k-1}^{j+1}}{h} = 0, \text{ при } C \geq 0 \quad (2)$$

$$\frac{u_k^{j+1} - u_k^j}{\tau} - C \frac{u_k^{j+1} - u_{k-1}^{j+1}}{h} = 0, \text{ при } C < 0 \quad (3)$$

Для:

$$\begin{cases} u(x, t) = -(t - 0.3)^3 + \cos(\pi x) + 2\pi x, \\ \varphi_0(x) = \cos(\pi x) + 0.027 + 2\pi x, \\ u(0, t) = t \\ C(x, t) = \frac{3(t-0.3)^2}{2\pi - \pi \sin(\pi x)}, \end{cases}$$

Выполнить следующие пункты:

1. Исследовать данную схему на точность и устойчивость.
2. Найти решение уравнения переноса в узлах сетки.
3. Вывести на экран графики приближенного и точного решения для значений t от 0 до 1 с шагом 0.1 и для $x = 0.5$.

2 Описание вычислительного метода.

Перейдем к дискретной постановке задачи: разобьем наши промежутки по x и по t на N_x и N_t равных частей соответственно. Тогда задача (1) с учетом (2):

$$\begin{cases} \frac{u_k^{j+1} - u_k^j}{\tau} + C_k^{j+1} \frac{u_k^{j+1} - u_{k-1}^{j+1}}{h} = 0, \text{ при } C_k^{j+1} \geq 0 \\ \frac{u_k^{j+1} - u_k^j}{\tau} - C_k^{j+1} \frac{u_k^{j+1} - u_{k-1}^{j+1}}{h} = 0, \text{ при } C_k^{j+1} < 0 \\ \left\{ x_k = kh : h = \frac{1}{N_x}, k = 0..N_x \right\} \\ \left\{ t_j = j\tau : \tau = \frac{1}{N_t}, j = 0..N_t \right\} \\ u_k^0 = \varphi_0(x_k) \\ u_0^j = u_0(t_j) \end{cases} \quad (4)$$

данный метод соответствует графической схеме:

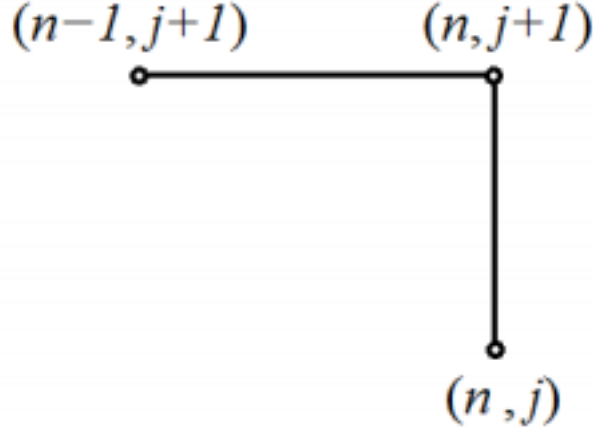


Рис. 1: Шаблон для схемы бегущего счета

3 Исследование данной схемы на точность и устойчивость.

3.1 Погрешность аппроксимации:

В данном случае разностный оператор $L_{h\tau}$ будет иметь вид:

$$L_{h\tau}u = \frac{u_k^{j+1} - u_k^j}{\tau} + C_k^{j+1} \frac{u_k^{j+1} - u_{k-1}^{j+1}}{h}$$

и выражение $L_{h\tau}u$ аппроксимирует Lu в точке (x_k, t_{j+1}) с погрешностью $O(\tau + h)$:

$$L_{h\tau}u = \left(\frac{\partial u}{\partial t} + C(x, t) \frac{\partial u}{\partial x} \right) \Big|_{x_k, t_{j+1}} + O(\tau + h)$$

3.2 Погрешность решения:

Погрешность ξ_k^j решения разностной схемы (2), обусловленная погрешностью начальных данных, будет удовлетворять уравнению

$$\begin{aligned} L_{\tau, h} \xi^{\tau, h} &= \psi^{\tau, h} \\ \xi_k^j &= u_k^j - u(x_k, t_j) \\ \xi_k^{j+1} &= \xi_k^j - \frac{c\tau}{h} (\xi_k^{j+1} - \xi_{k-1}^{j+1}) \end{aligned}$$

Разложим сеточную функцию ξ_k^j в ряд по e^{iqx_k}

$$\xi_k^j = \sum_q \xi_{k,q}^j = \sum_q C_q^j e^{iqx_k}$$

Так как схема (2) является линейной двухслойной схемой с постоянными коэффициентами, то на слое $(j + 1)$ погрешность будет иметь вид

$$\xi_k^{j+1} = \sum_q \xi_{k,q}^{j+1} = \sum_q \lambda_q C_q^j e^{iqx_k} \text{ где } \lambda_q - \text{ множители роста}$$

Введем обозначения $r = \frac{c\tau}{h}$ и $\alpha_q = qh$. Тогда, рассматривая уравнения для каждой гармоники $\xi_{k,q}^j$ в отдельности, получаем:

$$\xi_k^{j+1} = \xi_k^j - r(\xi_k^{j+1} - \xi_{k-1}^{j+1})$$

Или, что тоже самое,

$$\lambda_q C_q^j e^{i\alpha_q k} = C_q^j e^{i\alpha_q k} - r(\lambda_q C_q^j e^{i\alpha_q k} - \lambda_q C_q^j e^{i\alpha_q (k-1)})$$

Сокращая на $C_q^j e^{i\alpha_q k}$, и выполнив несколько несложных преобразований, получаем:

$$\frac{\lambda_q - 1}{\tau} + c\lambda_q \frac{1 - e^{-i\alpha_q}}{h} = 0,$$

Откуда получаем:

$$\lambda_q = \frac{1}{1 + r - re^{-i\alpha_q}} = \frac{1}{\beta_q}, \text{ где } \beta_q = 1 + r - re^{-i\alpha_q},$$

Спектр $\lambda_q(\alpha_q)$ оператора перехода со слоя на слой в рассматриваемой задаче представляет собой окружность на комплексной плоскости с центром в точке $1 - r$ и радиусом $|r|$. Так как $\lambda_q(\alpha_q)$ не зависит от τ явным образом, то спектральное условие устойчивости схемы принимает вид:

$$|\lambda_q| \leq 1, \forall q. \quad (5)$$

Следовательно, для того чтобы схема (2) была устойчивой по начальным данным, необходимо и достаточно, чтобы спектр оператора перехода со слоя на слой полностью содержался в круге единичного радиуса с центром в нуле на комплексной плоскости.

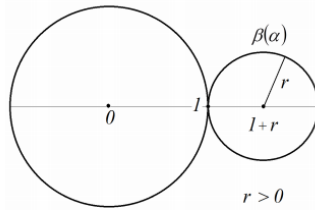


Рис. 2: Спектр оператора перехода со слоя на слой в схеме (2) при $c \geq 0$, $0 \leq r \leq 1$

Очевидно, что условие (5) выполняется, если $|\beta_q| \geq 1$. При изменении параметра α_q от минус до плюс бесконечности значения функции $\beta_q(\alpha_q)$ пробегают окружность радиуса $|r|$ с центром в точке $1 + r$ на комплексной плоскости.

При $c > 0$ параметр r также положителен. Следовательно, окружность, заполняемая значениями β_q , расположена вне единичного круга с центром в начале координат и касается единичной окружности в точке 1 ((2)). Это означает, что $|\beta_q| \geq 1$, при любом соотношении τ и h , то есть при положительном c схема (2) безусловно устойчива по начальным данным.

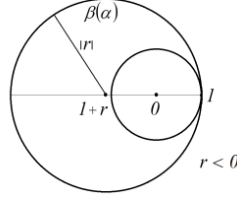


Рис. 3: Спектр оператора перехода со слоя на слой в схеме (2) при $c \leq 0$

Если скорость переноса c отрицательна, то и параметр r отрицателен, и условие $|\beta_q| \geq 1$ устойчивости схемы (2) будет выполнено при $|r| \geq 1$ ((3)). Следовательно, при $c < 0$ схема (2) является условно устойчивой по начальным данным, и условие ее устойчивости имеет вид $|c|\tau \geq h$.

Но в силу того, что в нашем случае при $c < 0$ мы выбираем зеркальный аналог сахемы, который наоборот безусловно устойчив при $c < 0$ и условно устойчив иначе получаем, что для такой комбинированной схемы имеем безусловную устойчивость на всем промежутке.

Метод гармоник применим только для схем с постоянными коэффициентами. Для исследования на устойчивость разностных схем для уравнений с переменными коэффициентами широко используют прием «замораживания» коэффициентов уравнения. При этом на устойчивость исследуется схема с постоянными коэффициентами, равными своим значениям в какой-то выбранной точке. Схему с переменными коэффициентами считают устойчивой, если условие устойчивости выполняется для соответствующей схемы с постоянными коэффициентами независимо от того, в какой точке были «заморожены» коэффициенты.

4 Проверим, что $u(x, t)$ - решение системы.

$$\frac{\partial u}{\partial t} + C(x, t) \frac{\partial u}{\partial x} = -3(t - 0.3)^2 + \frac{3(t - 0.03)^2}{2\pi - \pi \sin(\pi x)} (2\pi - \pi \sin(\pi x)) = 0 - \text{верно}$$

$$u(x, 0) = -(-0.3)^3 + \cos(\pi x) + 2\pi x = \varphi_0(x) - \text{верно}$$

$$u(0, t) = -(t - 0.3)^3 + 1 \neq u_0(t)$$

\Rightarrow Неверное граничное условие. Будем пользоваться $u(0, t) = -(t - 0.3)^3 + 1$

5 Описание алгоритма.

- Main class
 - Создаем поля для разбиения сетки, коэффициента для второй производной (main class для простоты замены начальных данных)
 - Создаем функции краевых условий и решения (main class для простоты замены начальных данных)
 - В main функции обращаемся к методу решения уравнения переноса и запускаем визуализацию на питоне сначала для приближенного решения, потом для точного решения
- ConvectionEquation class
 - создаем поля для шага по t, x
 - функция для решения уравнения переноса:
 - * создаем двумерный массив для записи решения
 - * заполняем первую строку и столбец начальными значениями
 - * далее в зависимости от знака функции C находим оставшиеся значения дискретной приближенной функции
 - * после нахождения всех коэффициентов матрицы решений, записываем ее в текстовый файл для графического вывода, подсчитываем максимальную ошибку (через максимум модуля) и печатаем ее.

6 Код программы (на Java).

6.1 Класс ConvectionEquation

```
package ru.nsu.mmfm.g16121.ddt.math;

import java.io.FileNotFoundException;
import java.io.PrintWriter;

import static ru.nsu.mmfm.g16121.ddt.main.Main.*;

public class ConvectionEquation {
    private static final double stepX = (rightBound - leftBound) /
    NUMBERS_COUNT_OF_GRID_BY_X;
    private static final double stepT = (rightBound - leftBound) /
    NUMBERS_COUNT_OF_GRID_BY_T;

    private static void writeForPython
    (String fileName, double[][] u, boolean func) {
```

```

try (PrintWriter writer = new PrintWriter(fileName)) {
writer.print("[[" + (int) leftBound + ", " + (int) rightBound
+ ", " + (NUMBERS_COUNT_OF_GRID_BY_X + 1) + "],");
writer.print "[" + (int) leftBound + ", " + (int) rightBound
+ ", " + (NUMBERS_COUNT_OF_GRID_BY_T + 1) + "],");

double x;
double t = leftBound;

writer.print("");
for (int i = 0; i <= NUMBERS_COUNT_OF_GRID_BY_T; i++) {
x = leftBound;
writer.print("");
for (int j = 0; j < NUMBERS_COUNT_OF_GRID_BY_X; j++) {
if (func) {
writer.print(u(x, t) + ",");
} else {
writer.print(u[i][j] + ",");
}
x += stepX;
}
if (func) {
writer.print(u(rightBound, t));
} else {
writer.print(u[i][NUMBERS_COUNT_OF_GRID_BY_X]);
}
if (i == NUMBERS_COUNT_OF_GRID_BY_T) {
writer.print("");
} else {
writer.print(",");
}
t += stepT;
}
writer.print("]]");
} catch (FileNotFoundException e) {
e.printStackTrace();
}
}

private static double maxError(double[][] u, double[][] error) {
double t = leftBound;
double max = 0;
for (int i = 0; i <= NUMBERS_COUNT_OF_GRID_BY_T; i++) {
double x = leftBound;

```

```

for (int j = 0; j <= NUMBERS_COUNT_OF_GRID_BY_X; j++) {
    error[i][j] = Math.abs(u(x, t) - u[i][j]);
    if (error[i][j] > max) {
        max = error[i][j];
    }
    x += stepX;
}
t += stepT;
}
return max;
}

public static void solveConvectionEquation() {
    double[][] u = new double[NUMBERS_COUNT_OF_GRID_BY_T + 1]
        [NUMBERS_COUNT_OF_GRID_BY_X + 1];

    //The first row of the matrix is filled by the initial data
    for (int i = 0; i <= NUMBERS_COUNT_OF_GRID_BY_X; i++) {
        u[0][i] = fi0(i * stepX);
    }

    //The first column are filled with source data
    for (int j = 0; j <= NUMBERS_COUNT_OF_GRID_BY_T; j++) {
        u[j][0] = u0(j * stepT);
    }

    for (int j = 0; j < NUMBERS_COUNT_OF_GRID_BY_T; ++j) {
        for (int i = 1; i <= NUMBERS_COUNT_OF_GRID_BY_X; ++i) {
            double currant = C(i * stepX, j * stepT) * stepT / stepX;
            if (currant >= eps) {
                u[j + 1][i] = (u[j][i] + currant * u[j + 1][i - 1]) / (1 + currant);
            } else {
                u[j + 1][i] = (u[j][i] - currant * u[j + 1][i - 1]) / (1 - currant);
            }
        }
    }

    double[][] error = new double
        [NUMBERS_COUNT_OF_GRID_BY_T + 1][NUMBERS_COUNT_OF_GRID_BY_X + 1];
    //write in the txt for display the result
    writeForPython("mainFunc.txt", u, true);
    writeForPython("result.txt", u, false);
    System.out.println("Max error = " + maxError(u, error));
    //      writeForPython("error.txt", error, false);
}

```



```
}  
}
```

6.2 Класс Main

```
package ru.nsu.mmf.g16121.ddt.main;  
  
import java.io.IOException;  
  
import static ru.nsu.mmf.g16121.ddt.math.  
ConvectionEquation.solveConvectionEquation;  
  
public class Main {  
    public static final double leftBound = 0;  
    public static final double rightBound = 1;  
  
    public static final int NUMBERS_COUNT_OF_GRID_BY_X = 500;  
    public static final int NUMBERS_COUNT_OF_GRID_BY_T = 500;  
  
    public static final double eps = 0.1E-6;  
  
    public static double u(double x, double t) {  
        return -Math.pow(t - 0.3, 3) + Math.cos(Math.PI * x) + 2.0 * Math.PI * x;}  
  
    public static double C(double x, double t) {  
        return 3.0 * Math.pow(t - 0.3, 2) /  
        (2.0 * Math.PI - Math.PI * Math.sin(Math.PI * x));}  
  
    public static double fi0(double x) {  
        return Math.cos(Math.PI * x) + 0.027 + 2.0 * Math.PI * x;  
    }  
  
    public static double u0(double t) {  
        return -Math.pow(t - 0.3, 3) + 1;  
    }  
  
    public static void main(String[] args) throws IOException {  
        solveConvectionEquation();  
        Runtime.getRuntime().exec("python3 vizualization.py");  
        Runtime.getRuntime().exec("python3 vizualization2.py");  
        //      Runtime.getRuntime().exec("python3 vizualization3.py");  
    }  
}
```

7 Графический вывод (Тесты).

7.1 Для $\tau = 0.2$ и $h = 0.2$

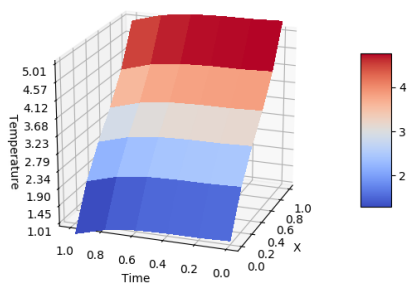


Рис. 4: Точное решение

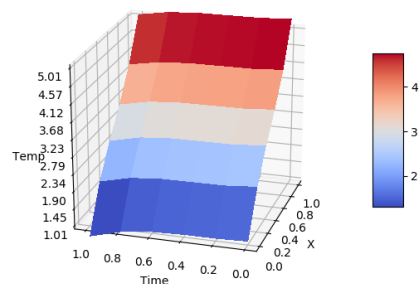


Рис. 5: Приближенное решение методом бегущего счета

Max error = 0.14000236418988088

Рис. 6: Максимальная погрешность решения

7.2 Для $\tau = 0.1$ и $h = 0.1$

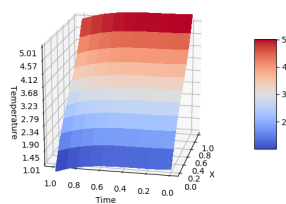


Рис. 7: Точное решение

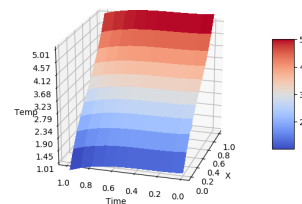


Рис. 8: Приближенное решение методом бегущего счета

Max error = 0.08073029752277794

Рис. 9: Максимальная погрешность решения

Таким образом, при увеличении τ и h в 2 раза погрешность решения уменьшилась примерно в 1,7 раз.

7.3 Для $\tau = 0.05$ и $h = 0.05$

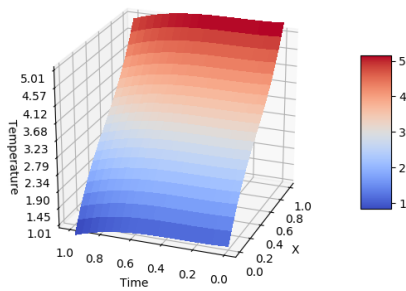


Рис. 10: Точное решение

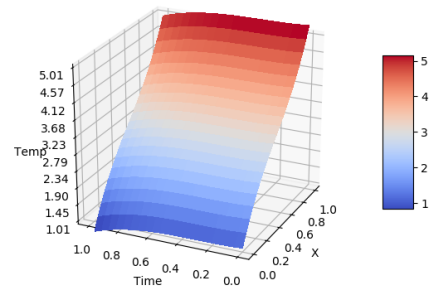


Рис. 11: Приближенное решение методом бегущего счета

Max error = 0.04344222122601327

Рис. 12: Максимальная погрешность решения

Таким образом, при увеличении τ и h еще в 2 раза погрешность решения уменьшилась примерно в 1,86 раз

7.4 График приближенного и точного решения для значений t от 0 до 1 с шагом $\tau = 0.1$ и для $x = 0.5$

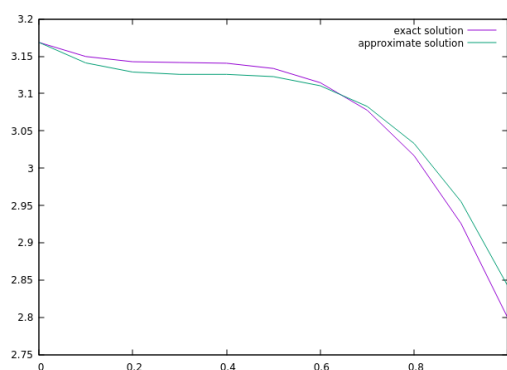


Рис. 13: Графики точного и приближенного решения при фиксированном $x = 0.5$

7.5 Для $\tau = 0.002$ и $h = 0.002$

Для достаточно большого числа разбиения графики практически неотличимы, но погрешность решения не нулевая.

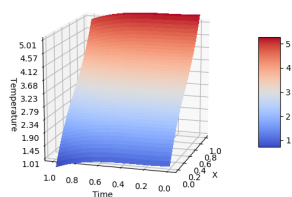


Рис. 14: Точное решение

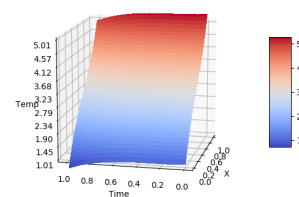


Рис. 15: Приближенное решение методом бегущего счета

Max error = 0.0018634348691239921

Рис. 16: Максимальная погрешность решения

8 Выводы.

Таким образом мы установили, что схема бегущего счета с модификацией для $C \leq 0$ является устойчивой, и не зависит от τ, h или a как, например, половина этой схемы бегущего счета, но, с другой стороны при малых τ, h погрешность решения достаточно велика и, следовательно, решение не достаточно точное.

Этот метод прост для понимания и реализации, что является большим плюсом. Также убедились на практике, что данный метод при достаточно больших τ, h наше дискретное решение практически не отличимо от непрерывного, что значительно упрощает решения многих видов уравнений.

Мы увидели, что теоретическая погрешность, которую мы посчитали до прогонки решения, практически совпадает с действительной погрешностью, а значит мы можем выбрать нужную нам точность решения заранее, что не мало важно для методов вычислений.