МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ
(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)

Институт №8 «Компьютерные науки и прикладная математика»
Кафедра 806 «Вычислительная математика и программирование»

Лабораторная работа №*6*
по курсу «Численные методы»

Численное решение уравнений гиперболического типа.

Выполнил: *К. А. Полонский*
Группа: *М8О-408Б-20*
Преподаватель: Д. Е. Пивоваров

Москва, 2023

Условие

1. Используя явную схему крест и неявную схему, решить начально-краевую задачу для дифференциального уравнения гиперболического типа. Аппроксимацию второго начального условия произвести с первым и со вторым порядком. Осуществить реализацию трех вариантов аппроксимации граничных условий, содержащих производные: двухточечная аппроксимация с первым порядком, трехточечная аппроксимация со вторым порядком, двухточечная аппроксимация со вторым порядком. В различные моменты времени вычислить погрешность численного решения путем сравнения результатов с приведенным в задании аналитическим решением $U(x,t)$. Исследовать зависимость погрешности от сеточных параметров τ, h.
2. Вариант 10:

   $$\frac{\partial^2 u}{\partial t^2} + 3\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} + \frac{\partial u}{\partial x} - u - \cos x \exp(-t),$$

   $u_x(0,t) = \exp(-t),$

   $u_x(\pi,t) = -\exp(-t),$

   $u(x,0) = \sin x,$

   $u_t(x,0) = -\sin x.$

   Аналитическое решение: $U(x,t) = \exp(-t)\sin x$..

Метод решения

Программа позволяет пользователю с помощью консольного ввода выбрать режим ввода параметров и метод решения гиперболического уравнения.

Сеточная функция представлена матрицей U размерности KxN, где K — число временных слоёв, N — число пространственных шагов.

Явная конечно-разностная схема была записана в форме:

$$u_j^{k+1} = \{(\sigma + \mu) * u_{j+1}^k + \left(-2 * \sigma + 2 + c * \tau^2\right) * u_j^k + (\sigma - \mu) * u_{j-1}^k + \left(-1 + d * \frac{\tau}{2}\right) * u_j^{k-1} + \tau^2 * f(j * h, k * \tau)\}/\left\{1 + d * \frac{\tau}{2}\right\}.$$

Неявная конечно-разностная схема была записана в форме:

$$a_j u_{j-1}^{k+1} + b_j u_j^{k+1} + c_j u_{j+1}^{k+1} = d_j,\ j = 1 \ldots N - 2,\ k = 0,\ 1,\ 2,\ \ldots$$

где (для двухточечной аппроксимации с первым порядком)

$$a_j = \frac{b}{2h} - \frac{a}{h^2},$$

$$b_j = \frac{1}{\tau^2} + \frac{d}{2\tau} + 2\frac{a}{h^2},$$

$$c_j = -\frac{b}{2h} - \frac{a}{h^2},\ j = 1,\ldots, N - 2.$$

$$a_0 = 0,$$

$b_0 = \beta_0 - \frac{\alpha_0}{h},$

$c_0 = \frac{\alpha_0}{h},$

$a_{N-1} = -\frac{\alpha_1}{h},$

$b_{N-1} = \beta_1 + \frac{\alpha_1}{h},$

$c_{N-1} = 0.$

Аппроксимация второго начального условия первым порядком была записана следующей формулой:

$u_j^1 = \psi_0(j * h) + \psi_1(j * h) * \tau.$

Аппроксимация второго начального условия вторым порядком была записана следующей формулой:

$$u_j^1 = \psi_0(j * h) + \psi_1(j * h) * \left(\tau - d\frac{\tau^2}{2}\right) + \left(a * deriv2(\psi_0, j * h) + b * deriv(\psi_0(j * h)) + c * \psi_0(j * h) + f(j * h, \tau)\right) * \frac{\tau}{2}$$

где

$f(x, t) = -cos(x) * e^{-t}$ — свободный член уравнения.

В конце работы программа записывает параметры условия, сеточную функцию и вектор ошибок в файл для скрипта отрисовки графиков.

Описание программы

Программа состоит из одного файла lab6.cpp, включающего функции:

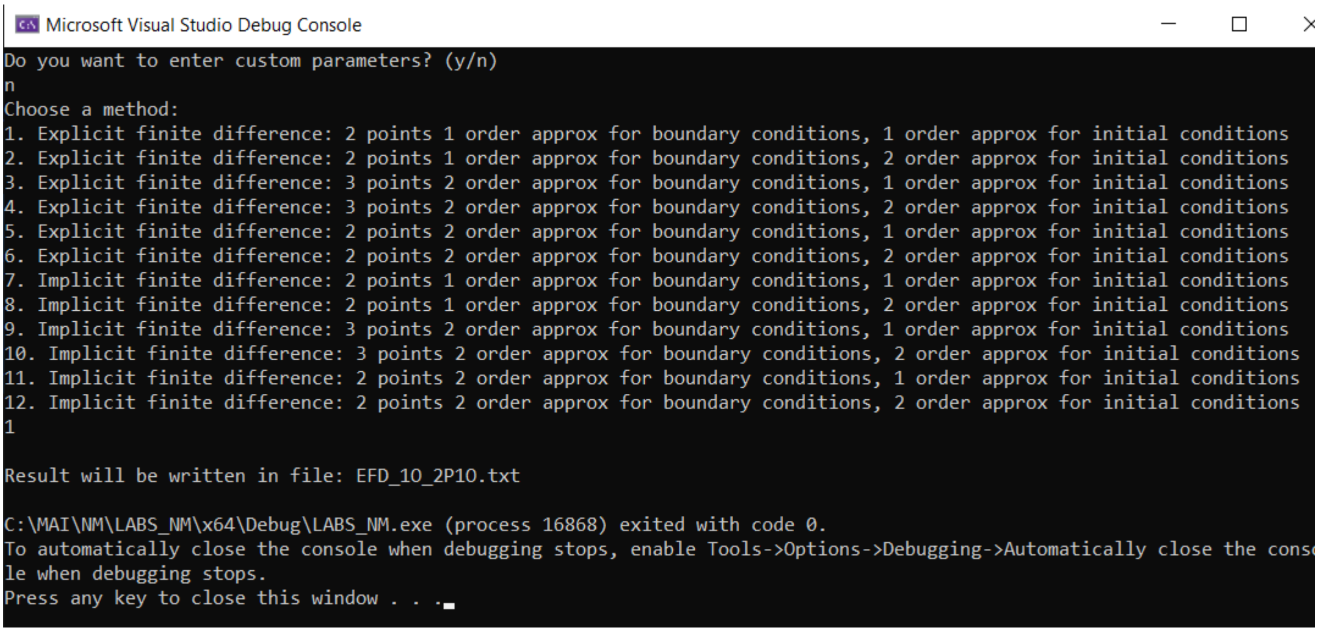
- double phi0(double t) — функция граничного условия
- double phi1(double t) — функция граничного условия
- double psi0(double x) — функция начального условия
- double psi1(double x) — функция начального условия
- double func(double x, double t) — функция свободного члена уравнения
- double analSol(double x, double t) — функция аналитического решения
- std::vector<std::vector<double>> explicitMethod(int initApprox, int boundApprox) — функция для запуска явной конечно-разностной схемы, принимающая вид аппроксимации граничных условий
- void explicitBound2Points1Order(std::vector<std::vector<double>>& U) — функция, осуществляющая двухточечную аппроксимацию с первым порядком
- void explicitBound3Points2Order(std::vector<std::vector<double>>& U) — функция, осуществляющая трёхточечную аппроксимацию со вторым порядком

- void explicitBound2Points2Order(std::vector<std::vector<double>>& U) — функция, осуществляющая двухточечную аппроксимацию со вторым порядком
- void explicitInit1Order(std::vector<std::vector<double>>& U) — функция, осуществляющая аппроксимацию второго начального условия с первым порядком
- void explicitInit2Order(std::vector<std::vector<double>>& U) — функция, осуществляющая аппроксимацию второго начального условия со вторым порядком
- double tridiagonalAlgo(std::vector<double>& a, std::vector<double>& b, std::vector<double>& c, std::vector<double>& d, std::vector<double>& x, int step, double prevP, double prevQ) — функция метода прогонки для решения СЛАУ с трёхдиагональной матрицей
- std::vector<std::vector<double>> implicitMethod(int initApprox, int boundApprox) — функция для запуска неявной конечно-разностной схемы, принимающая вид аппроксимации граничных условий
- void implicitBound2Points1Order(std::vector<std::vector<double>>& U, std::vector<double>& lower, std::vector<double>& main, std::vector<double>& upper, std::vector<double>& coeffs, bool getA, int k) — функция, осуществляющая двухточечную аппроксимацию с первым порядком и принимающая три массива, соответствующие диагоналям матрицы СЛАУ, а также вектор свободных коэффициентов
- void implicitBound3Points2Order(std::vector<std::vector<double>>& U, std::vector<double>& lower, std::vector<double>& main, std::vector<double>& upper, std::vector<double>& coeffs, bool getA, int k) — функция, осуществляющая трёхточечную аппроксимацию со вторым порядком и принимающая три массива, соответствующие диагоналям матрицы СЛАУ, а также вектор свободных коэффициентов
- void implicitBound2Points2Order(std::vector<std::vector<double>>& U, std::vector<double>& lower, std::vector<double>& main, std::vector<double>& upper, std::vector<double>& coeffs, bool getA, int k) — функция, осуществляющая двухточечную аппроксимацию со вторым порядком и принимающая три массива, соответствующие диагоналям матрицы СЛАУ, а также вектор свободных коэффициентов
- void implicitInit1Order(std::vector<std::vector<double>>& U) — функция, осуществляющая аппроксимацию второго начального условия с первым порядком
- void implicitInit2Order(std::vector<std::vector<double>>& U) — функция, осуществляющая аппроксимацию второго начального условия со вторым порядком
- std::vector<double> getError(std::vector<std::vector<double>>& U) — функция, вычисляющая погрешность
- double deriv(double (*f)(double), double x) — функция для расчёта первой производной

- double deriv2(double (*innerFunc)(double x1), double x) — функция расчёта второй производной

Результаты

Для построения графиков функций (аналитического решения и численного) была написана программа на языке Python, использующая библиотеки numpy и matplotlib. Графики были построены для временного слоя timeSlice = 20, оранжевый цвет использовался для аналитического решения, чёрный — для численного.



```
Microsoft Visual Studio Debug Console
Do you want to enter custom parameters? (y/n)
n
Choose a method:
1. Explicit finite difference: 2 points 1 order approx for boundary conditions, 1 order approx for initial conditions
2. Explicit finite difference: 2 points 1 order approx for boundary conditions, 2 order approx for initial conditions
3. Explicit finite difference: 3 points 2 order approx for boundary conditions, 1 order approx for initial conditions
4. Explicit finite difference: 3 points 2 order approx for boundary conditions, 2 order approx for initial conditions
5. Explicit finite difference: 2 points 2 order approx for boundary conditions, 1 order approx for initial conditions
6. Explicit finite difference: 2 points 2 order approx for boundary conditions, 2 order approx for initial conditions
7. Implicit finite difference: 2 points 1 order approx for boundary conditions, 1 order approx for initial conditions
8. Implicit finite difference: 2 points 1 order approx for boundary conditions, 2 order approx for initial conditions
9. Implicit finite difference: 3 points 2 order approx for boundary conditions, 1 order approx for initial conditions
10. Implicit finite difference: 3 points 2 order approx for boundary conditions, 2 order approx for initial conditions
11. Implicit finite difference: 2 points 2 order approx for boundary conditions, 1 order approx for initial conditions
12. Implicit finite difference: 2 points 2 order approx for boundary conditions, 2 order approx for initial conditions
1

Result will be written in file: EFD_10_2P10.txt

C:\MAI\NM\LABS_NM\x64\Debug\LABS_NM.exe (process 16868) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the conso
le when debugging stops.
Press any key to close this window . . .
```

Рис. 1. Консольное взаимодействие программы с пользователем.

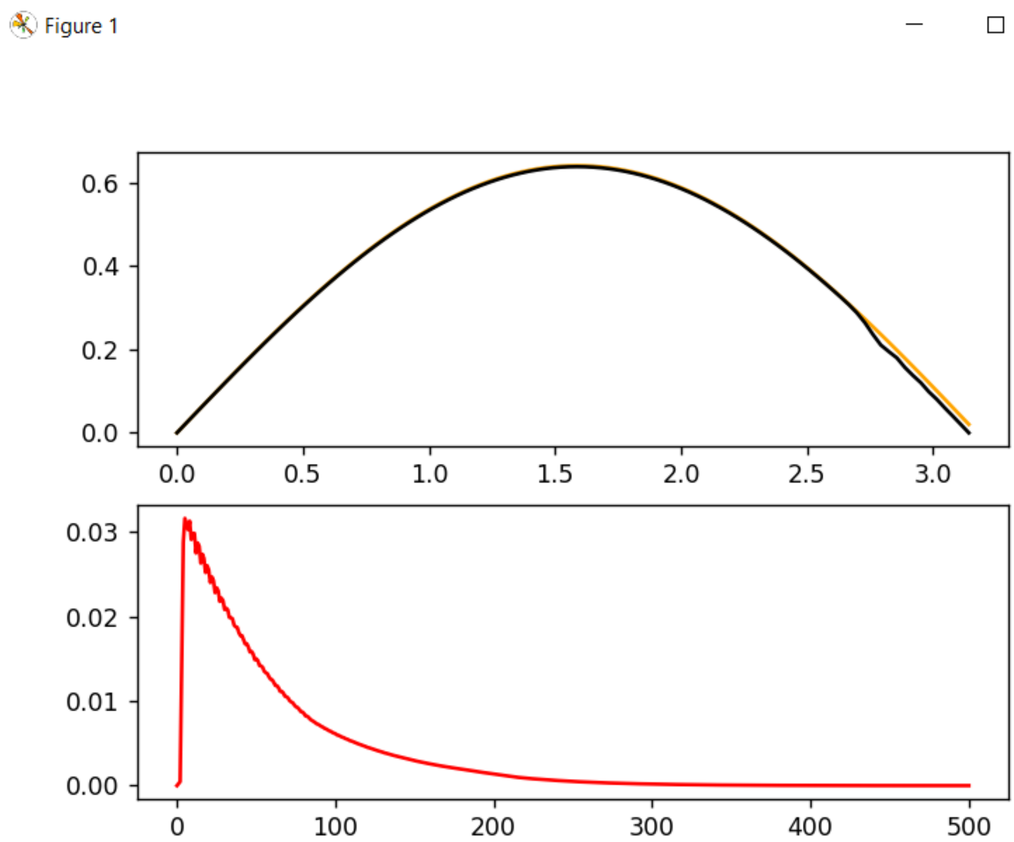


Рис. 2. График численного решения явной конечно-разностной схемой с двухточечной аппроксимацией второго порядка для граничных условий и аппроксимацией начального условия со вторым порядком.

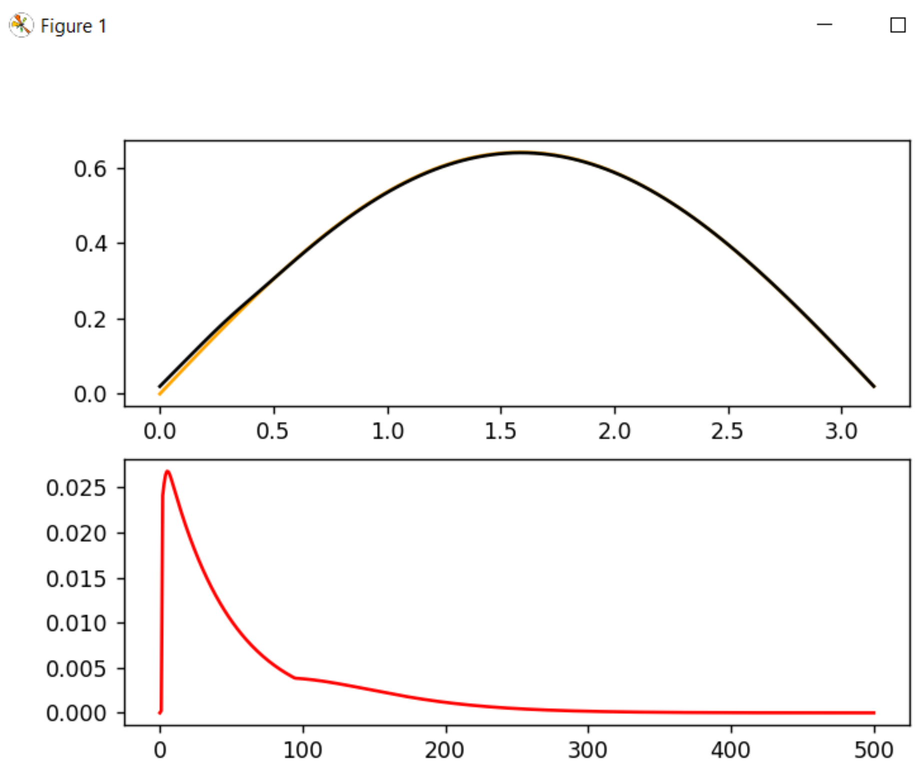


Рис. 3. График численного решения неявной конечно-разностной схемой с трёхточечной аппроксимацией второго порядка для граничных условий и аппроксимацией начального условия с первым порядком.

Выводы

В ходе выполнения данной лабораторной работы я освоил численные методы решения уравнений гиперболического типа, а именно явную и неявную конечно-разностные схемы. Практическое применение решения данной задачи лежит в области моделирования физических процессов (в частности, процесс малых поперечных колебаний струны) и может применяться как при исследовательской деятельности, так и при разработке специализированного ПО.