Московский авиационный институт
(национальный исследовательский университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

Курсовой проект
по курсу «Численные методы»

Студент: А. А. Садаков
Группа: М8О-406Б-19

Москва, 2023

Лабораторная работа №6

Задача: 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.

Вариант: 5

Начально-краевая задача:

$$\begin{cases} \dfrac{\partial^2 u}{\partial t^2} = 2\dfrac{\partial^2 u}{\partial x^2} + 4\dfrac{\partial u}{\partial x}, \\ u(0,t) = 0, \\ u(\pi,t) = 0, \\ u(x,0) = 0, \\ u_t(x,0) = \exp(-x)\sin(x) \end{cases}$$

Точное решение:

$$U(x,t) = 0.5 * \exp(-x)\sin(x)\sin(2t)$$

Вывод программы для шагов $h_x = 0.0314$ и $h_t = 0.001$:

=====6.1=====
Введите начально-краевую задачу:
0: utt = 2 * uxx + 4 * ux
1: u(0) = 0
2: u(3.14) = 0
3: u(0) = 0
4: ut(0) = exp(0 - x) * sin(x)

Введите количество шагов для "x"и для "t":
Введите функцию для сравнения:
Размер шага для x: 0.03140000
Размер шага для t: 0.00100000
Размер таблицы функции U: 100x1000
Point 2 Order 1
Средняя погрешность явной схемы: 0.00101716
Средняя погрешность неявной схемы: 0.00105933
Point 2 Order 2
Средняя погрешность явной схемы: 0.00101716
Средняя погрешность неявной схемы: 0.00105933
Point 3 Order 2
Средняя погрешность явной схемы: 0.00101716
Средняя погрешность неявной схемы: 0.00105933

Вывод программы для шагов $h_x = 0.0314$ и $h_t = 0.01$:

=====6.1=====
Введите начально-краевую задачу:
0: utt = 2 * uxx + 4 * ux
1: u(0) = 0
2: u(3.14) = 0
3: u(0) = 0
4: ut(0) = exp(0 - x) * sin(x)
Введите количество шагов для "x"и для "t":
Введите функцию для сравнения:
Размер шага для x: 0.03140000
Размер шага для t: 0.01000000
Размер таблицы функции U: 100x100
Point 2 Order 1
Средняя погрешность явной схемы: 0.00100324
Средняя погрешность неявной схемы: 0.00156031
Point 2 Order 2
Средняя погрешность явной схемы: 0.00100324
Средняя погрешность неявной схемы: 0.00156031

Point 3 Order 2

Средняя погрешность явной схемы: 0.00100324

Средняя погрешность неявной схемы: 0.00156031

Исходный код

6-1.cpp:

```cpp
#include "6-1.hpp"

double Point2Order1 (const std::vector<double> &coeff, double h, double u1, double f,
    uint64_t i) {
    double alpha = coeff[0], beta = coeff[1];
    double ans = 0;
    if (i == 0) {
        ans += f - alpha * u1 / h;
        ans /= beta - alpha / h;
    } else {
        ans += f + alpha * u1 / h;
        ans /= beta + alpha / h;
    }
    return ans;
}

double Point2Order2 (const std::vector<double> &ux, const std::vector<double> &coeff,
    double h, double t, double u0, double u1, double f, uint64_t i) {
    double alpha = ux[0], beta = ux[1];
    double a = coeff[0], b = coeff[1], c = coeff[2];
    double ans = 0;
    if (i == 0) {
        ans += h / t * u0 - f * (2 * a - b * h) / alpha + 2 * u1 * a / h;
        ans /= 2 * a / h + h / t - c * h - (beta / alpha) * (2 * a - b * h);
    } else {
        ans += h / t * u0 + f * (2 * a + b * h) / alpha + 2 * u1 * a / h;
        ans /= 2 * a / h + h / t - c * h + (beta / alpha) * (2 * a + b * h);
    }
    return ans;
}

double Point3Order2 (const std::vector<double> &coeff, double h, double u1, double u2,
    double f, uint64_t i) {
    double alpha = coeff[0], beta = coeff[1];
    double ans = 0;
    if (i == 0) {
        ans += f - alpha * (4 * u1 - u2) / (2 * h);
        ans /= beta - 3 * alpha / (2 * h);
    } else {
        ans += f + alpha * (4 * u1 - u2) / (2 * h);
        ans /= beta + 3 * alpha / (2 * h);
    }
    return ans;
}

```

```
43 void Time1Approx (std::vector<std::vector<double>> &u, const std::function<double(double)>
       &f, double xh, double th, double a) {
44     for (uint64_t i = 0; i < u[0].size(); ++i) {
45         u[1][i] = u[0][i] + u[1][i] * th + a * derivative(f, xh * i, 2) * th * th / 2;
46     }
47 }
48
49 void TIteration (std::vector<std::vector<double>> &u, const std::vector<std::vector<double
       >> &ux, double X0, double xh, double th, const std::vector<double> &coeff, const std::
       function<double(double, double)> &f, ApproxLevel left, ApproxLevel right, uint64_t i)
       {
50     double a = coeff[0], b = coeff[1], c = coeff[2], d = coeff[3];
51     double alpha[2] = {ux[0][0], ux[1][0]}, beta[2] = {ux[0][1], ux[1][1]};
52     double coff;
53     uint64_t n = u[0].size() - 2;
54     uint64_t start = 0, end = n;
55     if (left != ApproxLevel::NONE) {
56         ++n;
57         start = 2;
58     } else {
59         start = 1;
60     }
61     if (right != ApproxLevel::NONE) {
62         ++n;
63         end = n - 2;
64     } else {
65         end = n - 1;
66     }
67     Matrix<double> M(n, n);
68     std::vector<double> ans(n, 0);
69     if (left != ApproxLevel::NONE) {
70         M(1, 0) = 2*a*th*th + b*th*th*xh;
71         M(1, 1) = 2*th*th*xh*xh*c - 4*a*th*th - 4*b*th*th*xh - 2*xh*xh - 3*d*th*xh*xh;
72         M(1, 2) = 2*a*th*th + 3*b*th*th*xh;
73         ans[1] = u[i - 1][1] * (-4*xh*xh - 4*d*th*xh*xh) + u[i - 2][1] * (2*xh*xh + d*th*xh
               *xh) - 2*th*th*xh*xh*f(X0 + xh, i*th);
74         switch (left) {
75             case ApproxLevel::POINT2_ORDER1:
76                 M(0, 0) = beta[0]*xh - alpha[0];
77                 M(0, 1) = alpha[0];
78                 ans[0] = u[i][0] * xh;
79                 break;
80             case ApproxLevel::POINT2_ORDER2:
81                 break;
82             case ApproxLevel::POINT3_ORDER2:
83                 coff = -alpha[0] / M(1, 2);
84                 M(0, 0) = 2*beta[0]*xh - 3*alpha[0] - M(1, 0) * coff;
85                 M(0, 1) = 4*alpha[0] - M(1, 1) * coff;
```

```
86                ans[0] = u[i][0] * 2*xh - ans[1] * coff;
87                break;
88            default:
89                break;
90        }
91    } else {
92        M(0, 0) = 2*th*th*xh*xh*c - 4*a*th*th - 4*b*th*th*xh - 2*xh*xh - 3*d*th*xh*xh;
93        M(0, 1) = 2*a*th*th + 3*b*th*th*xh;
94        ans[0] = u[i - 1][1] * (-4*xh*xh - 4*d*th*xh*xh) + u[i - 2][1] * (2*xh*xh + d*th*xh
              *xh) - 2*th*th*xh*xh*f(X0 + xh * 0, i*th) - u[i][0] * (2*a*th*th + b*th*th*xh);
95    }
96
97    for (uint64_t j = start; j < end; ++j) {
98        M(j, j - 1) = 2*a*th*th + b*th*th*xh;
99        M(j, j) = 2*th*th*xh*xh*c - 4*a*th*th - 4*b*th*th*xh - 2*xh*xh - 3*d*th*xh*xh;
100       M(j, j + 1) = 2*a*th*th + 3*b*th*th*xh;
101       ans[j] = u[i - 1][j + 1] * (-4*xh*xh - 4*d*th*xh*xh) + u[i - 2][j + 1] * (2*xh*xh +
               d*th*xh*xh) - 2*th*th*xh*xh*f(X0 + j*xh, i*th);
102   }
103
104   if (right != ApproxLevel::NONE) {
105       M(n - 2, n - 3) = 2*a*th*th + b*th*th*xh;
106       M(n - 2, n - 2) = 2*th*th*xh*xh*c - 4*a*th*th - 4*b*th*th*xh - 2*xh*xh - 3*d*th*xh*
              xh;
107       M(n - 2, n - 1) = 2*a*th*th + 3*b*th*th*xh;
108       ans[n - 2] = u[i - 1][n - 2] * (-4*xh*xh - 4*d*th*xh*xh) + u[i - 2][n - 2] * (2*xh*
              xh + d*th*xh*xh) - 2*th*th*xh*xh*f(X0 + (n - 2) * xh, i*th);
109       switch (right) {
110           case ApproxLevel::POINT2_ORDER1:
111               M(n - 1, n - 2) = -alpha[1];
112               M(n - 1, n - 1) = beta[1]*xh + alpha[1];
113               ans[n - 1] = u[i][n - 1] * xh;
114               break;
115           case ApproxLevel::POINT2_ORDER2:
116               break;
117           case ApproxLevel::POINT3_ORDER2:
118               coff = alpha[1] / M(1, 2);
119               M(n - 1, n - 2) = -4*alpha[1] - M(n - 2, n - 2) * coff;
120               M(n - 1, n - 1) = 2*beta[1]*xh + 3*alpha[1] - M(n - 2, n - 1) * coff;
121               ans[n - 1] = u[i][n - 1] * 2*xh - ans[n - 2] * coff;
122               break;
123           default:
124               break;
125       }
126   } else {
127       M(n - 1, n - 2) = 2*a*th*th + b*th*th*xh;
128       M(n - 1, n - 1) = 2*th*th*xh*xh*c - 4*a*th*th - 4*b*th*th*xh - 2*xh*xh - 3*d*th*xh*
              xh;
```

```
129         ans[n - 1] = u[i - 1][n] * (-4*xh*xh - 4*d*th*xh*xh) + u[i - 2][n] * (2*xh*xh + d*
                th*xh*xh) - 2*th*th*xh*xh*f(X0 + n*xh, i*th) - u[i][n - 1] * (2*a*th*th + b*th*
                th*xh);
130     }
131     ans = RUNsolveSLAE(M, ans);
132     for (uint64_t j = 0; j < ans.size(); ++j) {
133         u[i][j + 1] = ans[j];
134     }
135 }
136 
137 void T (std::vector<std::vector<double>> &u, const std::vector<std::vector<double>> &ux,
        double X0, double xh, double th, const std::vector<double> &coeff, const std::function
        <double(double, double)> &f, ApproxLevel left, ApproxLevel right) {
138     for (uint64_t i = 2; i < u.size(); ++i) {
139         TIteration(u, ux, X0, xh, th, coeff, f, left, right, i);
140     }
141 }
142 
143 void CrossIteration (std::vector<std::vector<double>> &u, double X0, double xh, double th,
         const std::vector<double> &coeff, const std::function<double(double, double)> &f,
        uint64_t i) {
144     double a = coeff[0], b = coeff[1], c = coeff[2], d = coeff[3];
145     for (uint64_t j = 1; j < u[i].size() - 1; ++j) {
146         double tmp = 0;
147         tmp += u[i - 1][j + 1] * (2*a*th*th + 3*b*xh*th*th);
148         tmp += u[i - 1][j] * (2*c*th*th*xh*xh - 4*a*th*th - 4*b*xh*th*th + 4*xh*xh + 4*d*th
                *xh*xh);
149         tmp += u[i - 1][j - 1] * (2*a*th*th + b*xh*th*th);
150         tmp += u[i - 2][j] * (-2*xh*xh - d*th*xh*xh);
151         tmp += 2*th*th*xh*xh * f(X0 + j*xh, i*th);
152         tmp /= (2*xh*xh + 3*d*th*xh*xh);
153         u[i][j] = tmp;
154     }
155 }
156 
157 void Cross (std::vector<std::vector<double>> &u, const std::vector<std::vector<double>> &
        ux, double X0, double xh, double th, const std::vector<double> &coeff, const std::
        function<double(double, double)> &f, ApproxLevel left, ApproxLevel right) {
158 
159     for (uint64_t i = 2; i < u.size(); ++i) {
160         --i;
161         uint64_t start = 0, end = u[i].size() - 1;
162         switch (left) {
163             case ApproxLevel::POINT2_ORDER1:
164                 u[i][start] = Point2Order1(ux[0], xh, u[i][start + 1], u[i][start], 0);
165                 break;
166             case ApproxLevel::POINT2_ORDER2:
```

```
167                 u[i][start] = Point2Order2(ux[0], coeff, xh, th, u[i - 1][start], u[i][start
                        + 1], u[i][start], 0);
168                 break;
169             case ApproxLevel::POINT3_ORDER2:
170                 u[i][start] = Point3Order2(ux[0], xh, u[i][start + 1], u[i][start + 2], u[i
                        ][start], 0);
171                 break;
172             default:
173                 break;
174         }
175         switch (right) {
176             case ApproxLevel::POINT2_ORDER1:
177                 u[i][end] = Point2Order1(ux[1], xh, u[i][end - 1], u[i][end], 0);
178                 break;
179             case ApproxLevel::POINT2_ORDER2:
180                 u[i][end] = Point2Order2(ux[1], coeff, xh, th, u[i - 1][end], u[i][end - 1],
                        u[i][end], 0);
181                 break;
182             case ApproxLevel::POINT3_ORDER2:
183                 u[i][end] = Point3Order2(ux[1], xh, u[i][end - 1], u[i][end - 2], u[i][end],
                        0);
184                 break;
185             default:
186                 break;
187         }
188         ++i;
189         CrossIteration(u, X0, xh, th, coeff, f, i);
190     }
191 }
192
193 std::vector<std::vector<double>> SolveIBVP (const Task &task, uint64_t xCount, uint64_t
        tCount, double Tmax, Method method, ApproxLevel approx) {
194     double X0 = task.X[0], Xl = task.X[1];
195     double th = Tmax / tCount, xh = (Xl - X0) / xCount;
196     std::vector<std::vector<double>> u(tCount, std::vector<double>(xCount, 0));
197     ApproxLevel left = ApproxLevel::NONE, right = ApproxLevel::NONE;
198     if (task.ux[0][0] != 0) {
199         left = approx;
200     }
201     if (task.ux[1][0] != 0) {
202         right = approx;
203     }
204     auto f = [&] (double x, double t) -> double {
205         return task.trees[0]({0, 0, 0, x, t});
206     };
207     auto fx0 = [&] (double t) -> double {
208         return task.trees[1](t);
209     };
```

```
    auto fxl = [&] (double t) -> double {
        return task.trees[2](t);
    };
    auto ft0 = [&] (double x) -> double {
        return task.trees[3](x);
    };
    auto ftt0 = [&] (double x) -> double {
        return task.trees[4](x);
    };
    for (uint64_t i = 0; i < u[0].size(); ++i) {
        u[0][i] = ft0(i * xh);
        u[1][i] = ftt0(i * xh);
    }
    Time1Approx(u, ft0, xh, th, task.coeff[0]);
    for (uint64_t i = 2; i < u.size(); ++i) {
        u[i].front() = fx0(i * th);
        u[i].back() = fxl(i * th);
    }
    switch (method) {
        case Method::EXPLICIT:
            Cross(u, task.ux, X0, xh, th, task.coeff, f, left, right);
            break;
        case Method::NOT_EXPLICIT:
            T(u, task.ux, X0, xh, th, task.coeff, f, left, right);
            break;
        default:
            break;
    }
    return u;
}
```

Выводы

При уменьшении шага h_t точность явной схемы повысилась в то время как точность неявной наоборот — понизиласть.