МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ
АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО
ОБРАЗОВАНИЯ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ЯДЕРНЫЙ УНИВЕРСИТЕТ «МИФИ»

Кафедра теоретической ядерной физики

На правах рукописи
УДК 519.633.6

Широков Денис Дмитриевич

«Численное решение уравнения теплопроводности с использованием локально-адаптивных сеток»

Выпускная квалификационная работа бакалавра
Направление подготовки 03.03.01 Прикладные математика и физика

Выпускная квалификационная
работа защищена

«___»_____2022 г.

Оценка _____

Секретарь ГЭК _____ Корнеев Ф.А.
к.ф.-м.н., доцент

Москва
7 июня 2022 г.

Пояснительная записка
к бакалаврской дипломной работе:

«Численное решение уравнения теплопроводности с использованием локально-адаптивных сеток»

Студент _____ Широков Д.Д.

Научный руководитель
к.ф.-м.н. _____ Кучугов П.А.

Рецензент
к.ф.-м.н. _____ Ладонкина М.Е.

Заведующий кафедрой
к.ф.-м.н., доцент _____ Муравьев С.Е.

Аннотация

В работе проведён подробный анализ численного метода разностных схем решения уравнения теплопроводности, исследуются преимущества и недостатки использования равномерных статических сеток в случае квазилинейных уравнений с существенно различающимися масштабами (как пространственными, так и временными) физических процессов. Разработано программное обеспечение, реализующее рассмотренные схемы численного решения, проверены описанные особенности каждой из схем. Показана необходимость использования сеток с локальным измельчением, подстраивающихся под особенности решения. Описаны некоторые теоретические основы данного метода. Реализован в виде программного кода соответствующий алгоритм, работоспособность которого проверена на нескольких тестовых задачах. Показаны преимущества использования блочных локально-адаптивных сеток по сравнению с использованием равномерных статических.

Содержание

Введение

Большинство моделей классической физики, таких как гидрогазодинамика, описываются начально-краевыми задачами для дифференциальных уравнений в частных производных второго порядка[1; 2]. Нахождение аналитического решения таких задач представляется возможным только в случае простых, канонических областей (таких, как круг, шар, прямоугольник), простых начальных и граничных условий, а также в случае линейных уравнений, описывающих простые физические процессы.

На практике же часто возникают нелинейные задачи, поставленные в областях сложной формы. Например, задача лазерного термоядерного синтеза (ЛТС), идя которой заключается в быстром нагреве и сжатии термоядерного топлива до температур и плотностей, необходимых для осуществления быстрого и эффективного протекания термоядерных реакций инерциально удерживаемой плазмы. Процессы распространения тепла в такой системе будут описываться нелинейным уравнением теплопроводности. Нелинейное уравнение теплопроводности также возникает в, например, задачах о самофокусировки световых пучков в нелинейных средах, эффекте T–слоя в низкотемпературной плазме, проблемы безударного сжатия; вообще с необходимостью в любой задаче, в которой присутствуют процессы самопроизвольного нарушения симметрии с понижением её степени [3]

Любой численный метод приближённого решения таких задач использует дискретизацию (то есть переход от бесконечномерного функционального пространства к конечномерному пространству). Один из основных методов — *метод конечных разностей*. Его основа заключается в том, что исходная непрерывная задача в области $G \subset \mathbb{R}^n$ сводится к *семейству разностных задач* — системам конечного числа линейных (в общем случае — нелинейных) уравнений на т.н. *разностные функции* — функции, заданные на конечном числе точек (именуемое *сеткой*), и принимающие значения (приближённые значения решения) на конечном числе точек. Такие задачи решаются алгоритмически и тем самым могут быть программно реализованы на современных ЭВМ. Более подробно метод описан в разделе 1.

Принципиальная возможность применения тех или иных алгоритмов основывается на вопросе об их сходимости, точности и устойчивости. Так, в работе [4] даётся обширное описание алгоритмов решения задач на *стати-*

ческих сетках, исследуются вопросы устойчивости и скорости сходимости. Более подробное исследование тех же вопросов в случае *неравномерных статических сеток* дан в работе [5].

Как уже отмечалось, в прикладных задачах приходится сталкиваться с квазилинейными уравнениями теплопроводности:

$$\frac{\partial u}{\partial t} = \sum_{\alpha=1}^{p} \frac{\partial}{\partial x_\alpha} \left[k_\alpha(u) \frac{\partial u}{\partial x_\alpha} \right]$$

Проблема использования *статических* сеток, то есть сеток, не меняющихся на протяжении всего алгоритмического процесса поиска решения, связана со следующим обстоятельством. В статьях [6; 7] показано, что одномерное уравнение теплопроводности в случае зависящего от температуры коэффициента теплопроводности имеет решения, производные которых разрывны в точках обращения в нуль решения $u(x,t)$, при этом поток тепла $k(u)\frac{\partial u}{\partial x}$ — непрерывен, то есть существует фронт температуры, который, как показано в [8], распространяется с конечной скоростью. Эти „проблемные точки решения“оказываются сильно локализованными: если для численного решения использовать достаточно грубые сетки, то основные ошибки в приближённом решении будут локализованны именно в окрестностях этих точек. Конечно, можно использовать более мелкий шаг сетки и улучшить точность решения, ибо, как предсказывает теория [4], приближённое решение должно сходится к точному при стремлении шага сетки к нулю.

Однако даже на мощных вычислительных системах расчёт сложных трёхмерных задач со сложной пространственной геометрией требует огромного числа точек сетки, что значительно увеличивает используемую память и расчётное время [9]. Более того, точность решения в области особенностей *существенно* влияет на точность решения во всей остальной области. Поэтому хотя бы для получения приемлемой картины решения в целом на всей области без точного учёта особенностей неизбежно приходится сильно измельчать сетку. Учитывая, что в подобластях гладкого поведения решения просто нет необходимости измельчать сетку настолько сильно, заключаем, что использование классических алгоритмов приводит к тому, что большая часть компьютерных вычислений производится напрасно.

Поэтому для данного класса гидродинамических проблем с локализован-

ными особенностями разрабатывались специальные методы *локально-адаптивных сеток* (*Adaptive mesh refinement*), учитывающие разномасштабное поведение решения. Например, в работе [10] предлагалось использовать адаптивную сетку, построение которой производится с помощью соответствующего преобразования координат. Конкретный вид преобразования задаётся с помощью некоторой функции Q, вид которой определяется особенностями решения исследуемой задачи. Т.к. вид функции Q выбирался вручную в зависимости от конкретной задачи, этот метод не обладал достаточной автономностью. Многие методы были основаны на геометрической адаптации расчётных сеток, что, в свою очередь, приводит к трудностям реализации на ЭВМ, поскольку неструктурированные сетки порождают нерегулярный доступ к памяти. С учётом современного развития массивно-параллельных архитектур процессоров с большим числом ядер, эффективность работы которых зависит в первую очередь от упорядоченности обращений в память, производительность методов с неструктурированными сетками оказывается неудовлетворительной.

Метод структурированных адаптивных сеток (Block-structured adaptive mesh refinement) был представлен в работах [11; 12] применительно к уравнениям гиперболического типа. Преимущества метода в:

- использовании простых прямоугольных областей определённого размещения, удобных для реализации на компьютере
- возможности использования архитектуры параллельных вычислений
- использовании точно таких же разностных схем, как и для статических декартовых сеток (с некоторыми алгоритмическими модификациями).

Целью данной работы является изучение метода структурированных декартовых локально-адаптивных сеток применительно к задачам для уравнения теплопроводности, программная реализация данного метода, сравнение со статическими аналогами.

В разделе 1 вводятся основные математические формулировки разностных задач. В разделе 2 описываются алгоритмы решения задач на статических сетках (которые в последствии непосредственно используются при решении методом адаптивных сеток), приводятся примеры решения модельных

задач. В разделе 3 приводится описание метода, программной реализации и результатов решения модельных задач.

1 Основные понятия теории разностных схем

Математическая формулировка физических задач, описанных во введении имеет вид:

$$\begin{cases} L[u](x) = f(x), & x = (x_1, \ldots, x_n)^T \in G \subset \mathbb{R}^n \\ \Gamma[u](x) = \mu(x), & x \in \partial G \end{cases}, \tag{1}$$

где

- L — дифференциальный оператор уравнения;
- Γ — оператор начально-краевых условий (в общем случае также дифференциальный);
- f, μ — заданные функции,
- G — заданная область из $\mathbb{R}^n$, ∂G — её граница.

Решения исходной задачи — функции $u(x)$ непрерывного аргумента $x \in G$, являются элементами некоторого функционального пространства H_0 с нормой $\|\cdot\|$. В методе конечных разностей область G заменяется на некоторое дискретное множество точек ω_h, именуемое *сеткой*, а функциональное пространство H_0 заменяется на H_h — гильбертово пространство сеточных функций $y_h : G \supset \omega_h \to \mathbb{R}$, где h — некоторый параметр, характеризующий сетку ω_h в области G. Например, равномерная статическая сетка:

$$\omega_h = \{x = (x_1, \ldots, x_n) \in G \mid x_i = h_i \cdot k,\ k = 0, 1, \ldots, N_i\ i = 1, \ldots, n\}$$

Получив приближённое решение задачи y_h, необходимо оценивать степень „близости“к решению исходной задачи $u(x)$. y_h и u являются элементами разных функциональных пространств, поэтому для оценивания близости в работе используется проекционный метод: пространство H_0 отображается (про-

ектируется) на пространство H_h оператором $\mathcal{P}$:

$$\mathcal{P}_h \colon H_0 \ni u \mapsto u_h \in H_h.$$

Простейший выбор: ограничение u на сетку ω_h:

$$u_h(x) := u(x), \quad x \in \omega_h \subset G$$

Иногда пользуются „более равномерным“способом ограничения с усреднением по окрестности узла:

$$u_h(x) := \frac{1}{2h} \int_{x-h}^{x+h} u(x')dx'$$

Тогда близость приближённого решения y_h и исходного решения u оценивается по норме $\|\cdot\|_h$ пространства H_h:

$$e = \|y_h - u_h\|_h,$$

при этом требуется, чтобы она аппроксимировала норму $\|\cdot\|_0$ в слудеющем смысле[4]:

$$\lim_{h\to 0} \|u_h\|_h = \|u\|_0$$

В работе используется норма $\|y\|_h = \sqrt{\sum_{i=1}^{N} h y_i^2}$.

Исходному дифференциальному оператору L ставится в соответствие *разностный оператор* L_h:

$$L_h[v](x) = \sum_{x' \in T(x)} A_h(x, x')v(x'),$$

где $T(x)$ — некоторое множество узлов сетки, называемое *шаблоном*, $A_h(x, x')$ — некоторые коэффициенты. Например, двумерный оператор Лапласа $L = \Delta$ на двумерной равномерной сетке можно аппроксимировать, используя шаблон „крест“(см. рис. 1):

$$\Delta u(x) = \frac{\partial^2 u}{\partial x_1^2} + \frac{\partial^2 u}{\partial x_2^2} \mapsto \frac{u_{i+1}^j - 2u_i^j + u_{i-1}^j}{h_1^2} + \frac{u_i^{j+1} - 2u_i^j + u_i^{j-1}}{h_2^2},$$

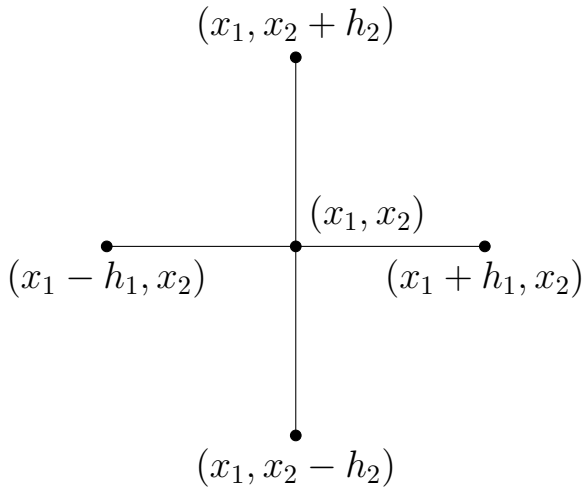

Рис. 1: Шаблон „Крест“

где $u_i^j = u(x_{1i}, x_{2j}), \quad (x_{1i}, x_{2j}) \in \omega_h$. Погрешность аппроксимации оператора L разностным оператором L_h определяется как сеточная функция $\psi_h = L_h[u_h] - (L[u])_h, u \in H_0$. Если $\|\psi_h\| = O(|h|^m)$, то говорят, что оператор L_h аппроксимирует оператор L с порядком m. Если $\psi(x) = O(h^m), m$, то говорят, что оператор L_h аппроксимирует оператор L в точке x с порядком m.

Теперь сформулируем непосредственно то, что называется *разностной схемой*, алгоритмы решения которой и реализуются на компьютере. Исходной задаче (1) ставится в соответствие семейство разностных задач, зависящих от параметра h, называемое разностной схемой:

$$\left\{ \begin{cases} L_h[y_h] = \varphi_h, & x \in \omega_h \\ l_h[y_h] = \chi_h, & x \in \gamma_h \end{cases} \right\}_h, \quad \varphi_h = \mathcal{P}_h[f], \chi_h = \mathcal{P}_h[\mu]$$

Под погрешностью разностной схемы понимается $z_h = y_h - u_h$, где $u_h = \mathcal{P}_h u$ — проекция решения исходной задачи на H_0. Решение разностной задачи сходится к решению исходной задачи, если

$$\|z_h\|_h \to 0 \text{ при } |h| \to 0.$$

Введём также понятие устойчивости схемы. Разностная схема называется устойчивой (корректной, сходящейся), если $\exists h_0 > 0 : \forall h(|h| \leqslant h_0) \Rightarrow$

$$1. \forall \varphi \in H_h \quad \exists! y_h — \text{ решение};$$
$$2. \exists M > 0 : \forall \varphi_h, \tilde{\varphi}_h \|y_h - \tilde{y}_h\| \leqslant M \|\varphi_h - \tilde{\varphi}_h\|$$

На этом только лишь математическая сторона вопроса формулирования проблемы завершена. Дальнейшие шаги по исследованию разностной схемы опираются на конкретный выбор множества сеток $\{\omega_h\}$ и аппроксимирующего оператора L_h, выбор которого, в свою очередь, существенно зависит от некоторых вопросов реализации получаемого алгоритма на компьютере.

2 Разностные схемы на статических сетках

Основное преимущество статических равномерных сеток:

- относительно простая реализация в виде программного кода[1];
- удобное и простое представление данных в программе. Так, например, рассматривая задачу для двумерного нестационарного уравнения теплопроводности, результаты вычислений программы могут храниться в трёхмерном массиве (в отличие от алгоритмов на нестатических и неравномерных сетках, где используются более сложные структуры, см. подробнее в разд. 3);
- существенное упрощение формул и доказательств сходимости, устойчивости получающихся разностных схем.

Всюду далее под равномерной статической сеткой будем понимать:

$$\omega_{h\tau} := \{(x_{ik} = k \cdot h_i, t_j = j \cdot \tau) \mid i = 1, \ldots, n; k = 1, \ldots, N_i\}$$

Как указывалось выше, свойства разностной схемы зависят и от выбора аппроксимирующего оператора L_h. В [4] исследуется множество различных схем. Далее приводятся избранные схемы, каждая из которых обладает отличительной особенностью, для задачи Дирихле для линейного:

$$\begin{cases} u_t = \Delta u + f, & (x,t) \in G \times (0,T) \\ u(x,t) = \mu_{-i}(x,t), \quad x_i = 0 & \\ u(x,t) = \mu_i(x,t), \quad x_i = L_i & , \quad t \in [0,T) \\ u(x,0) = u_0(x), & x \in \bar{G} \end{cases}$$

[1]Так, например, реализация явной схемы может занимать не более 10 строк кода, вне зависимости от размерности уравнения(подробнее см. раздел 2.2)

и квазилинейного уравнений теплопроводности:

$$\begin{cases} u_t = \sum\limits_{\alpha=1}^{p} \dfrac{\partial}{\partial x_\alpha}\left[k_\alpha(u)\dfrac{\partial u}{\partial x_\alpha}\right] + f, & (x,t) \in G \times (0,T) \\ u(x,t) = \mu_{-i}(x,t), & x_i = 0 \\ u(x,t) = \mu_i(x,t), & x_i = L_i \\ u(x,0) = u_0(x), & x \in \bar{G} \end{cases}, \quad t \in [0,T)$$

где $\bar{G} = \prod\limits_{i=1}^{n}[0, L_i]$.

2.1 Некоторые классические разностные схемы для уравнения теплопроводности

2.1.1 Явная схема

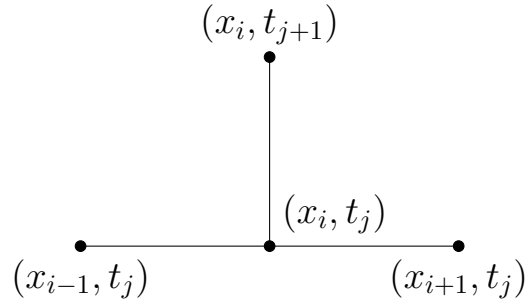

Рис. 2: Пятиточечный шаблон явной схемы

Шаблон явной схемы нагляден в одномерном случае (см. рис. 2). Множество $\{(x,t) \in \omega_h \mid t = t_j\}$ будем называть j–ым временным слоем. Оператор $\frac{\partial u}{\partial t}$ аппроксимируется, используя значения функции на $(j+1)$–ом временном слое и j–ом временном слое, а оператор Лапласа аппроксимируется, используя значения только на j–ом временном слое:

$$L = \frac{\partial u}{\partial t} - \Delta u \mapsto L_{h\tau} = \frac{u_i^{j+1} - u_i^j}{\tau} - \frac{u_{i+1}^j - 2u_i^j + u_i^j}{h^2} + \varphi_i^j, \quad \varphi_i^j = f(x_i, t_j)$$

В многомерном случае:

$$L \mapsto \frac{u_i^{j+1} - u_i^j}{\tau} + \sum_{k=1}^{n} \frac{u_{i+}^j - 2u_i^j + u_{i-}^j}{h_k^2} + \varphi_i^j, \text{ где}$$

$$i_\pm = (i_1, \ldots, i_{k-1}, i_k \pm 1, i_{k+1}, \ldots, i_n)$$

Такой „запаздывающий“выбор шаблона позволяет явно выразить значения функции на $(j+1)$–ом временном слое через значения на j–ом временном слое:

$$u_i^{j+1} = u_i^j + \tau \sum_{k=1}^{n} \frac{u_{i+}^j - 2u_i^j + u_{i-}^j}{h_k^2} + \varphi_i^j \tag{2}$$

(отсюда и название схемы). Значения функции на 1–ом временном слое следуют из начальных условий: $u_i^1 = u_0(x_i)$, а граничные условия дают замкнутую систему уравнений:

$$\begin{cases} u_i^j = u_i^j + \tau \sum\limits_{k=1}^{n} \frac{u_{i+}^j - 2u_i^j + u_{i-}^j}{h_k^2} + \varphi_i^j, & i_k = 2, \ldots, N_k - 1 \\ u_i^1 = u_0(x_i), & i_k = 1, \ldots, N_k \\ u_{i_k}^j = \mu_{-\alpha}(x_i, t_j), & i_{k\neq\alpha} = 1, \ldots, N_k;\ i_\alpha = 0 \\ u_{i_k}^j = \mu_{+\alpha}(x_i, t_j), & i_{k\neq\alpha} = 1, \ldots, N_k;\ i_\alpha = N_\alpha \end{cases}$$

Одно из преимуществ такой схемы — простота программной реализации. Так, задав начальные условия на 1–ом временном слое и граничные значения на всех последующих $j > 1$, явно считаются значения на всех последующих слоях. В разделе 2.2 представлен код и описание программы, реализующей явную разностную схему для одномерного уравнения теплопроводности. Другое преимущество схемы — скорость счёта. Число арифметических операций, необходимых для расчёта значений функции на одном временном слое порядка $O(N_x)$, где N_x — число точек по оси x (в многомерном случае — $O(\prod N_\alpha)$). Ещё одно преимущество схемы — возможность параллельного счёта. Из формулы (2) видно, что расчёт значений u_i^{j+1} не зависит от i: зависимость от i присутствует в правой части, но эти значения в j–ый момент времени уже предполагаются известными, поэтому значения на новом временном слое u_i^{j+1} и u_k^{j+1} для разных i и k могут считаться независимо друг от друга.

Схема обладает лишь первым порядком сходимости: $O(h + \tau)$, что за-

ставляет использовать достаточно мелкую сетку. Более того, схема имеет существенный недостаток — она устойчива и сходится к решению лишь при условии $\tau \leqslant \frac{h^2}{2n}$.

Как показано в [4], в случае квазилинейных уравнений условие устойчивости приобретает вид:

$$\tau \leqslant \frac{h^2}{2 \cdot \max k(u)}$$

Откуда видно, что если $k(u)$ является быстроменяющейся функцией (например $k \sim u^\sigma$), то *использование явных схем нецелесообразно*, поскольку требуется очень мелкий шаг τ по времени. Поэтому для квазилинейных уравнений теплопроводности применяются преимущественно *неявные* схемы.

2.1.2 Однопараметрическое семейство неявных схем

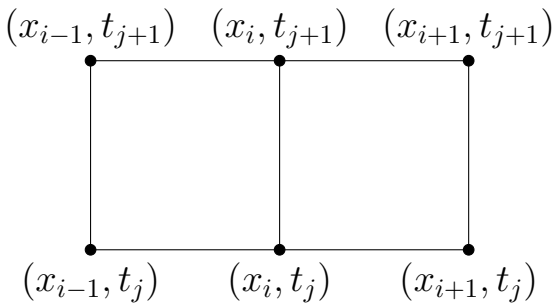

Рис. 3: Шеститочечный шаблон явной схемы

Идея неявных схем заключается в использовании шаблона, затрагивающего как значения функции на j–ом временном слое (который предполагается уже известным), так и на последующих слоях (значения функции на которых ещё не известны). Так, однопараметрическое семейство неявных схем, зависящих от параметра $\sigma \in (0, 1]$ получается при аппроксимации дифференциального оператор на шеститочечном шаблоне (см. рис. 3), причём значения на „верхнем“и „нижнем“временных слоях берутся с весами σ и $1 - \sigma$ соответственно.

Так, аппроксимация дифференциального оператора для одномерного урав-

нения с постоянными коэффициентами выглядит следующим образом:

$$L = \frac{\partial u}{\partial t} - \Delta u \mapsto L_{h\tau} =$$
$$= \frac{u_i^{j+1} - u_i^j}{\tau} - \sigma \frac{u_{i+1}^{j+1} - 2u_i^{j+1} + u_i^{j+1}}{h^2} - (1-\sigma)\frac{u_{i+1}^j - 2u_i^j + u_i^j}{h^2} - \varphi_i^j,$$
$$\varphi_i^{j+1} = f(x_i, t_{j+1})$$

В многомерном случае можно задавать целый вектор $\sigma = (\sigma_1 \dots \sigma_n)$:

$$L \mapsto \frac{u_i^{j+1} - u_i^j}{\tau} -$$
$$- \sum_{k=1}^{n} \left[\sigma_k \frac{u_{i+}^{j+1} - 2u_i^{j+1} + u_{i-}^{j+1}}{h_k^2} + (1-\sigma_k)\frac{u_{i+}^j - 2u_i^j + u_{i-}^j}{h_k^2} \right] -$$
$$- \varphi_i^j, \text{ где}$$
$$i_\pm = (i_1, \dots, i_{k-1}, i_k \pm 1, i_{k+1}, \dots, i_n)$$

Аналогично явной схеме, добавляя граничные и начальные условия, получаем замкнутую систему линейных уравнений. Однако, в отличие от явной схемы, значения на новом временном слое u^{j+1} не выражаются явно через значения на предыдущем временном слое u^j, а получаются путём решения соответствующей системы линейных алгебраических уравнений $Mu^{j+1} = D$. В одномерном случае матрица системы получается *трёхдиагональной*:

$$M = \begin{pmatrix} B_1 & C_1 & 0 & 0 & 0 & 0 \\ A_2 & B_2 & C_2 & 0 & 0 & 0 \\ 0 & A_3 & B_3 & C_3 & 0 & 0 \\ 0 & 0 & \ddots & \ddots & \ddots & 0 \\ 0 & 0 & 0 & A_{n-1} & B_{n-1} & C_{n-1} \\ 0 & 0 & 0 & 0 & A_n & B_n \end{pmatrix},$$

где

$$A_i = -\sigma\tau, \quad B_i = h^2 + 2\sigma\tau, \quad C_i = -\sigma\tau$$
$$D_i = h^2 u_i^j + \tau(1-\sigma)\left(u_{i-1}^j - 2u_i^j + u_{i+1}^j\right) + \tau h^2 \varphi_i^{j+1}$$

В таком случае существует эффективный алгоритм расчёта, именуемый *методом прогонки*: Сначала определяем α и β:

$$\begin{cases} \alpha_1 = -\frac{C_1}{B_1}, \quad \beta_1 = \frac{D_1}{B_1}, \\ \alpha_i = -\frac{C_i}{B_i + A_i \cdot \alpha_{i-1}} \\ \beta_i = \frac{D_i - A_i \cdot \beta_{i-1}}{B_i + A_i \cdot \alpha_{i-1}} \\ \beta_n = \frac{D_n - A_n \cdot \beta_{n-1}}{B_n + A_n \cdot \alpha_{n-1}} \end{cases}, \qquad i = 2, \ldots, n-1$$

Затем по ним определяем неизвестные:

$$\begin{cases} x_n = \beta_n, \\ x_i = \alpha_i \cdot x_{i+1} + \beta_i, \quad i = n-1, \ldots, 1 \end{cases}$$

Сложность такого алгоритма $O(N_x)$, что не уступает явной схеме. Преимущества же неявной схемы в том, что для неё условие устойчивости принимает вид

$$\sigma \geqslant \frac{1}{2} - \frac{h^2}{4\tau}.$$

Из этого условия видно, что при $\sigma \geqslant 0.5$ схема безусловно устойчива, то есть устойчива вне зависимости от выбора шагов h и τ.

При значениях $\sigma \neq 0.5$ схема имеет порядок точности $O(h^2 + \tau)$, а при $\sigma = 0.5$ (так называемая *схема Кранка-Николсона*) $O(h^2 + \tau^2)$, то есть больший, чем явная схема.

Недостатком такой схемы является отсутствие возможности распараллеливания.

В случае многомерной задачи дело обстоит хуже. Получаемые системы линейных уравнений имеют более сложную матрицу (не трёхдиагональную). Для их решения пользуются в общем случае методом последовательного исключения переменных (методом Гаусса), который работает за $O(N^3)$, где $N \times N$ — размерность матрицы. Так, уже в двумерной задаче матрица будет размера $N_x N_y \times N_x N_y$, и расчёт будет проводиться заметно дольше, чем при расчёте явной схемой ($\sim O(N^2)$).

2.1.3 Локально-одномерные схемы

Итак, явные схемы обладают быстрой скоростью счёта ($\sim O(N^n)$), в то время как устойчивость таких систем достигается лишь при определённом выборе параметров сетки. Неявные схемы безусловно устойчивы и имеют больший порядок точности, однако требуют решения системы N^n уравнений, для чего требуется значительно больше вычислительной работы, чем для явной схемы [1].

Для сочетания лучших качеств явных (объём работы $\sim O(N^n)$) и неявных (безусловная устойчивость) схем было предложено несколько *экономичных* схем. Подробнее об этом написано в [4; 13—17].

Локально-одномерный метод является универсальным, пригодным для решения квазилинейного уравнения теплопроводности в произвольной области G любого числа измерений. При использовании в работе блочных локально-адаптивных сеток (раздел 3) используется именно этот метод. Также будут использоваться прямоугольные области, поэтому формулировка метода будет приведена для таковых.

Итак, рассматриваемую многомерную задачу Дирихле в цилиндре $\bar{Q}_T = \bar{G} \times [0,T]$, $\bar{G} = \prod\limits_{i=1}^{n} [0, L_i]$

$$\begin{cases} u_t = \sum\limits_{\alpha=1}^{p} \dfrac{\partial}{\partial x_\alpha} \left[k_\alpha(u) \dfrac{\partial u}{\partial x_\alpha} \right] + f, & (x,t) \in G \times (0,T] \\ u(x,t) = \mu_{-i}(x,t), & x_i = 0 \\ u(x,t) = \mu_i(x,t), & x_i = L_i \\ u(x,0) = u_0(x), & x \in \bar{G} \end{cases}, \quad t \in [0,T)$$

заменяем *цепочкой одномерных* задач „вдоль каждого из направлений“:

$$\begin{cases} \frac{1}{p} \frac{\partial v_{(\alpha)}}{\partial t} = \frac{\partial}{\partial x_\alpha} \left[k_\alpha(v_{(\alpha)}) \frac{\partial v_{(\alpha)}}{\partial x_\alpha} \right] + f_\alpha, & x \in G, t \in \Delta_\alpha = \left(t_{j+\frac{\alpha-1}{p}}, t_{j+\frac{\alpha}{p}} \right] \\ v_{(\alpha)}(x, t_{j+\frac{\alpha-1}{p}}) = v_{(\alpha-1)}(x, t_{j+\frac{\alpha-1}{p}}), & x \in G \\ v_{(\alpha)}(x,t) = \mu_{-\alpha}(x,t), & x_\alpha = 0, t \in [t_{j+\frac{\alpha-1}{p}}, t_{j+\frac{\alpha}{p}}] \\ v_{(\alpha)}(x,t) = \mu_\alpha(x,t), & x_\alpha = L_\alpha, t \in [t_{j+\frac{\alpha-1}{p}}, t_{j+\frac{\alpha}{p}}] \end{cases}$$

$$v_{(\alpha)}(x,0) = u_0(x)$$
$$v_{(1)}(x,t_j) = v_{(p)}(x,t_j)$$
$$u(x,t_{j+1}) = v_{(p)}(x,t_{j+1}).$$

Каждая из одномерных задач решается неявной двухслойной схемой:

$$\frac{v_{(\alpha)}^{j+\frac{\alpha}{p}} - v_{(\alpha)}^{j+\frac{\alpha-1}{p}}}{\tau} = \frac{1}{h_\alpha^2}\left[a_{i_\alpha+1}^{(\alpha)}\left(v_{(\alpha)}^{j+\frac{\alpha-1}{p}}\right)\cdot\left(v_{(\alpha)i_\alpha+1}^{j+\frac{\alpha}{p}} - v_{(\alpha)i_\alpha}^{j+\frac{\alpha}{p}}\right) - \right.$$
$$\left. - a_{i_\alpha}^{(\alpha)}\left(v_{(\alpha)}^{j+\frac{\alpha-1}{p}}\right)\cdot\left(v_{(\alpha),i_\alpha}^{j+\frac{\alpha}{p}} - v_{(\alpha),i_\alpha-1}^{j+\frac{\alpha}{p}}\right)\right] + f_\alpha\left(v_{(\alpha)}^{j+\frac{\alpha-1}{p}}\right),$$
$$\text{где } a_i^{(\alpha)}(y) = k_\alpha\left(\frac{y_{i-1}+y_i}{2}\right), \quad i_\alpha \pm 1 = (i_1,\ldots,i_{\alpha-1},i_\alpha \pm 1, i_{\alpha+1},\ldots,i_p)$$

Напишем подробнее, как выглядит данная схема в случае двумерного квазилинейного уравнения теплопроводности. Цилиндр $\bar{Q}_T = [0,L_1]\times[0,L_2]\times[0,T]$ дискретизуется равномерной сеткой:

$$\omega_{h\tau} = \{(x_i,y_k,t_j) \mid x_i = h_x\cdot(i-1), y_k = h_y\cdot(k-1), t_j = \tau\cdot(j-1),$$
$$i = 1,\ldots,N_x, \quad k = 1,\ldots,N_y, \quad j = 1,\ldots,N_t\}$$

Исходная задача с уравнением

$$\frac{\partial u}{\partial t} = \frac{\partial}{\partial x}\left(k_1(u)\frac{\partial u}{\partial x}\right) + \frac{\partial}{\partial y}\left(k_2(u)\frac{\partial u}{\partial y}\right) + f$$

заменяется на две одномерные задачи, каждая из которых решается неявной схемой. Обозначим $w := u^{j+1/2}$ — промежуточное значение, получающееся при решении первой из задач на интервале времени $(t_j, t_{j+1/2})$ (вдоль оси x_1). Решение в начальный момент времени $t_1 = 0$ известно из начальных данных:

$$u_{i,k}^1 = u_0(x_i,y_k) \quad \forall i,k$$

Зная решение на j–ом временном слое $u_{i,k}^j$, решение на $(j+1)$–ом ищется следующим образом: сначала решается N_y одномерных задач вдоль оси x

(N_y — для каждого $y_k, k = 1, \ldots, N_y$ независимо):

$$\begin{cases} \frac{w_{i,k}-u^j_{i,k}}{\tau} = \frac{1}{h_x^2}\left[k_1\left(\frac{u^j_{i,k}+u^j_{i+1,k}}{2}\right)\cdot(w_{i+1,k}-w_{i,k}) - \right. \\ \qquad\qquad \left. -k_1\left(\frac{u^j_{i-1,k}+u^j_{i,k}}{2}\right)\cdot(w_{i,k}-w_{i-1,k})\right] + \frac{1}{2}\varphi^{j+1/2}_{i,k}, \quad i = 2, \ldots, N_x - 1 \\ w_{1,k} = \mu_{-1}(y_k, t_{j+1/2}) \\ w_{N_x,k} = \mu_1(y_k, t_{j+1/2}) \end{cases}$$

Затем решается N_x одномерных задач вдоль оси y (N_x — для каждого $x_i, i = 1, \ldots, N_x$ независимо):

$$\begin{cases} \frac{u^{j+1}_{i,k}-w_{i,k}}{\tau} = \frac{1}{h_y^2}\left[k_2\left(\frac{w_{i,k}+w_{i,k+1}}{2}\right)\cdot\left(u^{j+1}_{i,k+1}-u^{j+1}_{i,k}\right) - \right. \\ \qquad\qquad \left. k_2\left(\frac{w_{i,k-1}+w_{i,k}}{2}\right)\cdot\left(u^{j+1}_{i,k}-u^{j+1}_{i,k-1}\right)\right] + \frac{1}{2}\varphi^{j+1}_{i,k}, \quad k = 2, \ldots, N_y - 1 \\ u^{j+1}_{i,1} = \mu_{-2}(x_i, t_{j+1}) \\ u^{j+1}_{i,N_y} = \mu_2(x_i, t_{j+1}) \end{cases}$$

Каждая из одномерных задач решается методом прогонки за $O(N)$, где $N = N_x, N_y$. И, таким образом, общая сложность такого алгоритма есть $O(N_y \cdot N_x + N_x \cdot N_y) = O(N^2)$, то есть совпадает по скорости решения с явной схемой.

В многомерном случае область G дискретизуется сеткой ω_h, имеющий вдоль каждого направления N точек. Для каждого значения $\alpha = 1, \ldots, p$ получается N^{p-1} задач. Каждая из них решается методом прогонки за $\sim O(N)$ и, таким образом, число арифметических операций, совершаемых в локально-одномерной схеме есть число порядка $\sim O(N^p)$. Помимо быстрой скорости счёта, ЛОС обладает и существенным достоинством неявных схем. Как показано в [4], схема аппроксимирует исходное уравнение в суммарном смысле: погрешность аппроксимации ψ для локально-одномерной схемы есть сумма погрешностей аппроксимации ψ_α на решении $u = u(x, t)$ для одномерных схем номера α:

$$\psi = \sum_{\alpha=1}^{p} \psi_\alpha = O(|h|^2 + \tau),$$

ЛОС безусловно устойчива и равномерно сходится. Это означает, что счёт по данной схеме устойчив даже при очень крупных шагах по времени, что

```julia
function explicit_scheme(f, u0, mu1, mu2, T, Nx, Nt)
    x = range(0, 1, length = Nx) # Точки секти по оси `x`
    t = range(0, T, length = Nt) # Точки сетки по оси `t`
    c = step(t) / step(x)^2 # Коэффициент Куранта

    u = zeros(Nx, Nt) # Матрица, хранящая решение разностной задачи

    u[:, 1] .= u0.(x) # Учёт начальных условий
    u[1, :] .= mu1.(t) # Учёт граничных условий на левом конце
    u[end, :] .= mu2.(t) # Учёт граничных условий на правом конце

    for j in 1:(Nt - 1) # Цикл по всем слоям
        for i in 2:(Nx - 1)
            u[i, j + 1] = u[i, j] + c * (u[i + 1, j] - 2 * u[i, j] + u[i - 1, j]) +
            f(x[i], t[j])
        end
    end

    return u
end
```

Listing 1: Реализация явной схемы для уравнения $u_t = u_{xx} + f$

позволяет быстро решать задачи, где не требуется большая точность.

Помимо прочего стоит отметить, что данный метод применим не только к параллелепипедам, но и к произвольным областям $G \subset \mathbb{R}^p$, сохраняет порядок точности на неравномерных сетках [5], обладает пониженными (по сравнению с другими схемами) требованиями к объёму оперативной памяти, а также допускает распараллеливание: вдоль каждого из направлений необходимо решать N систем линейных уравнений, и все они могут решаться независимо друг от друга и, следовательно, параллельно.

Учитывая все преимущества локально-одномерной схемы для уравнения теплопроводности, именно она берётся в основу при рассмотрении локально-адаптивных сеток в разделе 3.

2.2 Программный код, примеры расчётов

Реализация выше обозначенных разностных схем производилась с помощью языка программирования `Julia`[18].

В качестве иллюстрации простоты явной разностной схемы, в листинге 1 приведена её реализация для одномерного линейного уравнения теплопроводности. Заметим, что схема реализована для уравнения $u_t = u_{xx} + f$ на отрезке $x \in [0, 1]$, поскольку более общий вид уравнения $u_t = a^2 u_{xx} + \tilde{f}$ на отрезке

$[a, b]$ сводится к первому линейной заменой $t \mapsto \tilde{t} = \frac{(b-a)^2}{a^2}t, \quad x \mapsto \tilde{x} = \frac{x-a}{b-a}$. Работоспособность программы была проверена на некоторых тестовых задачах [19]. Так, для задачи

$$\begin{cases} u_t = u_{xx} + f, & (x,t) \in (0,1) \times (0,T] \\ u(x,0) = x\sin(3\pi x), & x \in [0,1] \\ u(0,t) = 0 \\ u(1,t) = 0 \end{cases}, \quad t \in [0,T] \tag{3}$$

рассчитано решение для $T = 0.05$ для различного набора сеток с числом точек по оси времени $N_t = 25001$ и числом точек по оси x в пределах от $N_x = 10$ до $N_x = 500$. Заметим, что выбор такого, казалось бы, необоснованно большого числа точек по оси времени диктуется неустойчивостью явной разностной схемы: из условия устойчивости $\frac{\Delta t}{(\Delta x)^2} < \frac{1}{2}$ следует, что для $N_x = 500$ требуется $N_t > 25000$. В таких простых, тестовых задачах такое соотношение ещё приемлемо, и современные компьютеры позволяют получить решение достаточно быстро. Однако в более сложных, многомерных задачах с резкими неоднородностями, требующими достаточно мелкого пространственного шага, это условие становится трудно выполнимым (такая точность по оси времени просто не требуется). Явная схема была реализована только для линейного уравнения. Как уже отмечалось выше, применения явной схемы нецелесообразно для квазилинейных уравнений.

На рис. 4 представлено численное решение задачи как поверхность-график $u_h(x,t)$, полученное явной схемой для $N_x = 500$ и $N_t = 25000$. На рис. 5 представлен график зависимости аналитического решения задачи, полученного методом разделения переменных Фурье:

$$u(x,t) = \frac{1}{2}e^{-(3\pi)^2 t}\sin(3\pi x) - \frac{48}{\pi^2}\sum_{m=1}^{\infty}\frac{m}{(9-4m^2)^2}e^{-(2\pi m)^2 t}\sin(2\pi m x)$$

Для точного сравнения также на рис. 6 представлен график зависимости локальной ошибки $|u(x,t) - u_h(x,t)|$. Видно, что в случае линейного уравнения без особенностей решения погрешность численного решения в среднем „размазана“по всей области.

Для проверки утверждения о скорости счёта разностной схемы также по-

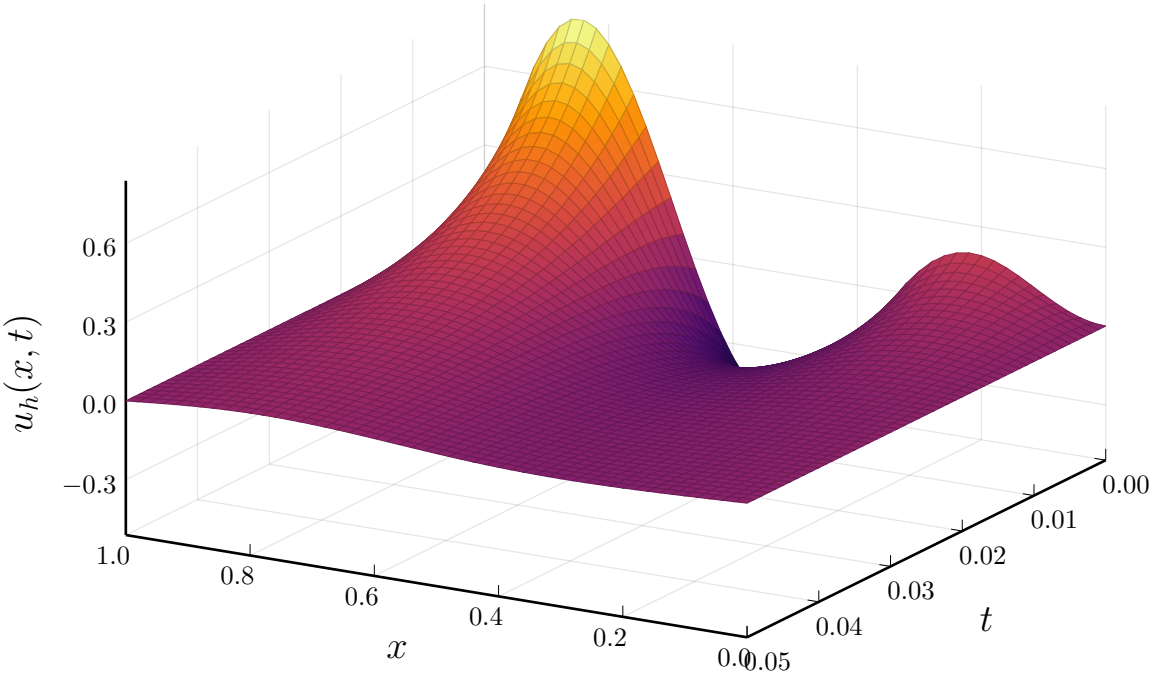

Рис. 4: Численное решение задачи (3) для $N_x = 500$ и $N_t = 25\,000$

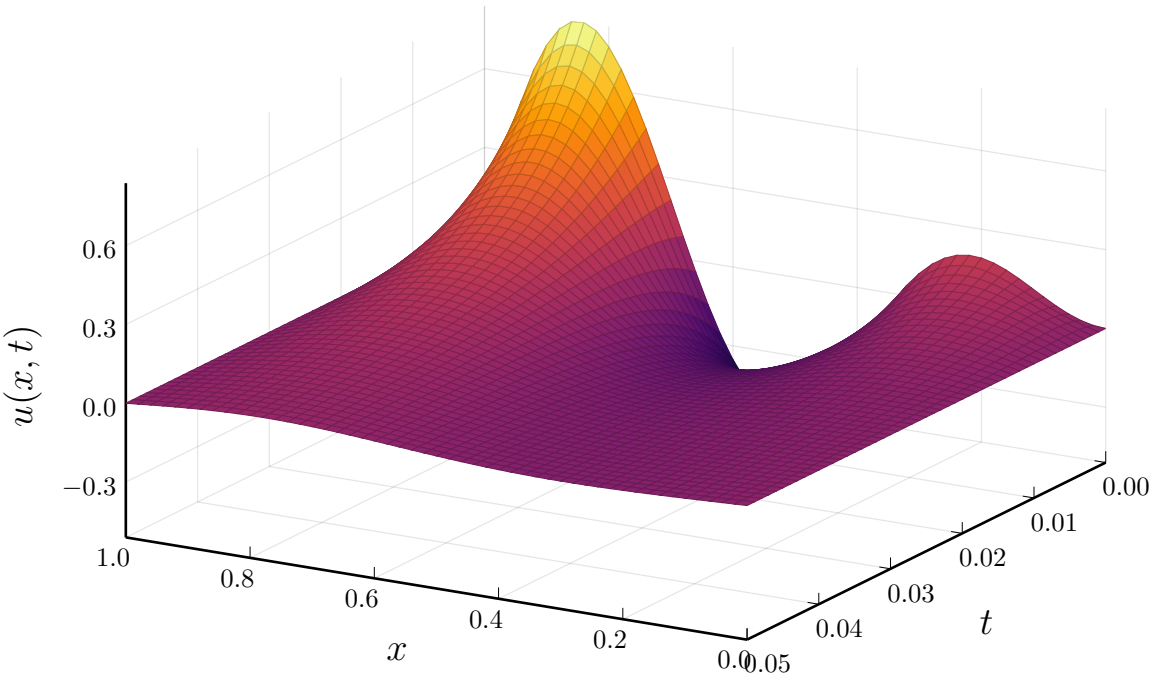

Рис. 5: Аналитическое решение задачи (3)

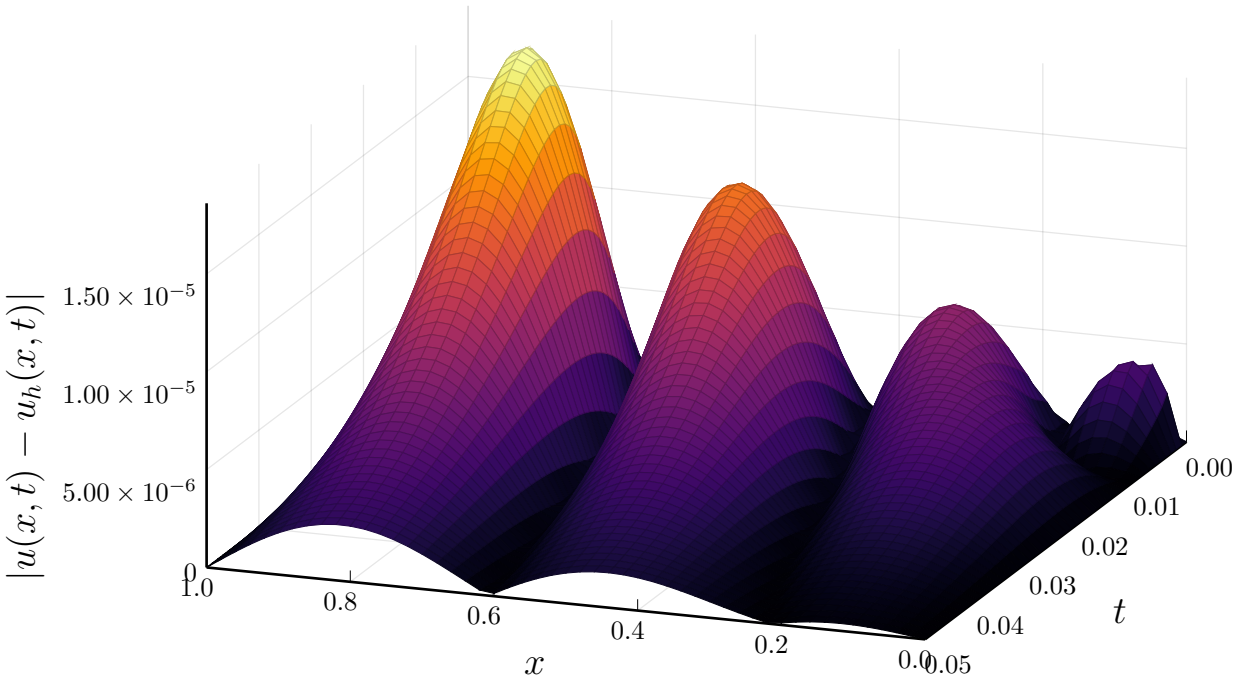

Рис. 6: График локальной ошибки численного решения задачи (3)

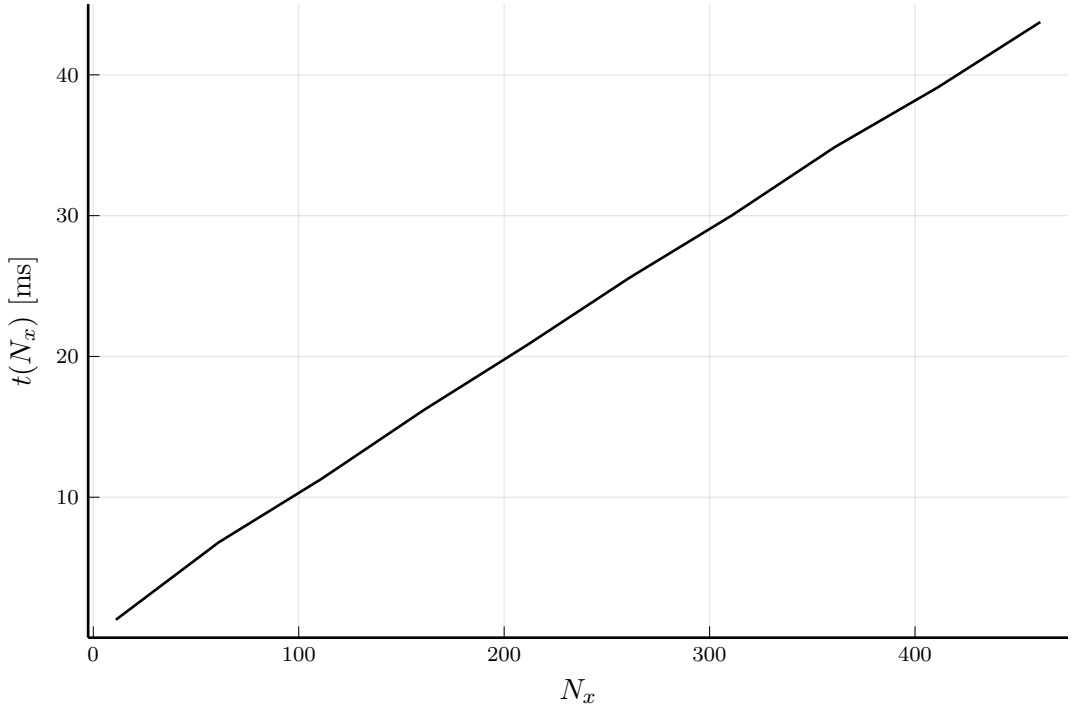

Рис. 7: График зависимости времени счёта от числа точек сетки $t(N_x)$ (в мс).

```julia
# Оптимальный по памяти алгоритм прогонки в случае, если его нужно запускать много раз
function tridiagonal_algorithm!(
        a::Vector{<:T}, b::Vector{<:T}, x::Vector{<:T},
        A::Vector{<:T}, B::Vector{<:T}, C::Vector{<:T}, D::Vector{<:T}
    ) where {T <: Number}

    a[1] = - C[1] / B[1]
    b[1] = D[1] / B[1]
    for i=2:(length(B)-1)
        a[i] = - C[i] / (B[i] + A[i-1] * a[i-1])
        b[i] = (D[i] - A[i-1] * b[i-1]) / (B[i] + A[i-1] * a[i-1])
    end
    b[end] = (D[end] - A[end] * b[end-1]) / (B[end] + A[end] * a[end-1])

    # обратный ход
    x[end] = b[end]
    for i = length(B)-1:-1:1
        x[i] = a[i]*x[i+1] + b[i]
    end
end
```

Listing 2: Реализация алгоритма прогонки

строен график зависимости среднего времени выполнения программы (как среднее время от $\sim 100 - 200$ запусков программы для одних и тех же входных данных) как функции от числа точек сетки по оси x (см. рис. 7).

Реализация неявной схемы требует написания алгоритма прогонки для решения систем линейных уравнений с трёхдиагональной матрицей. В листинге 2 представлен код алгоритма прогонки. Дополнительные переменные α, β, x передаются в функцию `tridiagonal_algorithm` с целью экономии памяти. Решение уравнения теплопроводности требует многократного выполнения алгоритма прогонки, поэтому общие временные переменные выделяются отдельно и сокращают количество ненужных аллокаций. В листинге 3 приведена реализация неявной схемы для одномерного квазилинейного уравнения теплопроводности. Работоспособность программы проверена на тех же тестовых задачах, что и для явной схемы. Из рисунка 8 видно, что, хоть абсолютная скорость счёта неявной схемы и уступает явной, асимптотика (то есть линейная зависимость) остаётся точно такой же.

Для проверки правильности учёта квазилинейности уравнения программа

```julia
function implicit_scheme(k, f, u0, mu1, mu2, T, Nx, Nt)
    x = range(0, 1, length = Nx) # Точки секти по оси `x`
    dx = step(x)
    t = range(0, T, length = Nt) # Точки сетки по оси `t`
    dt = step(t)

    u = zeros(Nx, Nt) # Матрица, хранящая решение разностной задачи
    u[:, 1] .= u0.(x) # Учёт начальных условий
    u[1, :] .= mu1.(t) # Учёт граничных условий на левом конце
    u[end, :] .= mu2.(t) # Учёт граничных условий на правом конце

    # Временные переменные для алгоритма прогонки
    A, C = (zeros(Nx) for i in 1:2)
    B, D, a, b, xt = (zeros(Nx) for i in 1:5)
    B[1] = B[end] = 1
    C[1] = A[end] = 0

    for j in 1:(Nt - 1) # Цикл по временным слоям
        # Обновление коэффициентов тридиагональной матрицы
        D[1] = u[1, j + 1]
        D[end] = u[end, j + 1]
        for i = 2:(Nx - 1)
            A[i - 1] = -dt * k(0.5 * (u[i - 1, j] + u[i, j]))
            B[i] = dx^2 + dt * (
                k(0.5 * (u[i, j] + u[i + 1, j])) +
                k(0.5 * (u[i - 1, j] + u[i, j]))
            )
            C[i] = -dt * k(0.5 * (u[i, j] + u[i + 1, j]))
            D[i] = dx^2 * u[i, j] + dt * dx^2 * f(x[i], t[j])
        end
        tridiagonal_algorithm!(a, b, xt, A, B, C, D) # Метод прогонки
        u[:, j + 1] .= xt
    end

    return u
end
```

Listing 3: Реализация неявной схемы для уравнения $u_t = (ku_x)_x + f$

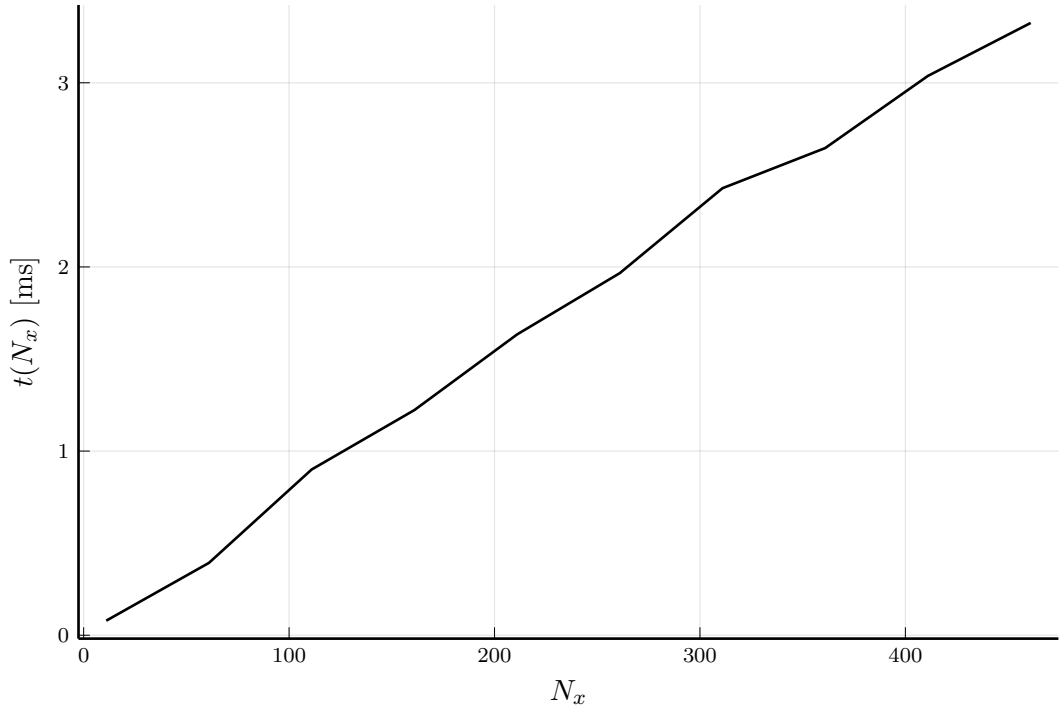

Рис. 8: График зависимости времени счёта от числа точек сетки для неявной схемы

проверена на задаче, рассмотренной в статье [20]:

$$\begin{cases} \dfrac{\partial u}{\partial t} = \dfrac{\partial}{\partial x}\left(\dfrac{1}{2}u^2\dfrac{\partial u}{\partial x}\right), & (x,t) \in (0,1)\times(0.1,0.4) \\ u(0,t) = 10\sqrt{t} \\ u(1,t) = 0 \end{cases}, \quad t \in [0.1, 0.4]$$

$$u(x,0.1) = \begin{cases} 2\sqrt{5(0.5-x)}, & x \leqslant 0.5 \\ 0, & x \geqslant 0.5 \end{cases}, \quad x \in [0,1]$$

На рис. 9 приведено численное решение, полученное неявной схемой на сетке с $N_x = 50$, $N_t = 50$. Также на рис. 10 представлен график зависимости локальной ошибки, посчитанной с помощью аналитического решения:

$$u(x,t) = \begin{cases} \left[\sigma c \varkappa_0^{-1}(ct + x_1 - x)\right]^{1/\sigma}, & x \leqslant x_1 + ct \\ 0, & x \geqslant x_1 + ct \end{cases} \tag{4}$$

где $\sigma = 2$, $\varkappa_0 = 0.5$, $x_1 = 0$, $c = 5$. Для большей наглядности представлены также два одномерных графика, показывающие профиль решения и локальную погрешность в некоторые моменты времени (см. рис. 11 и рис. 12)

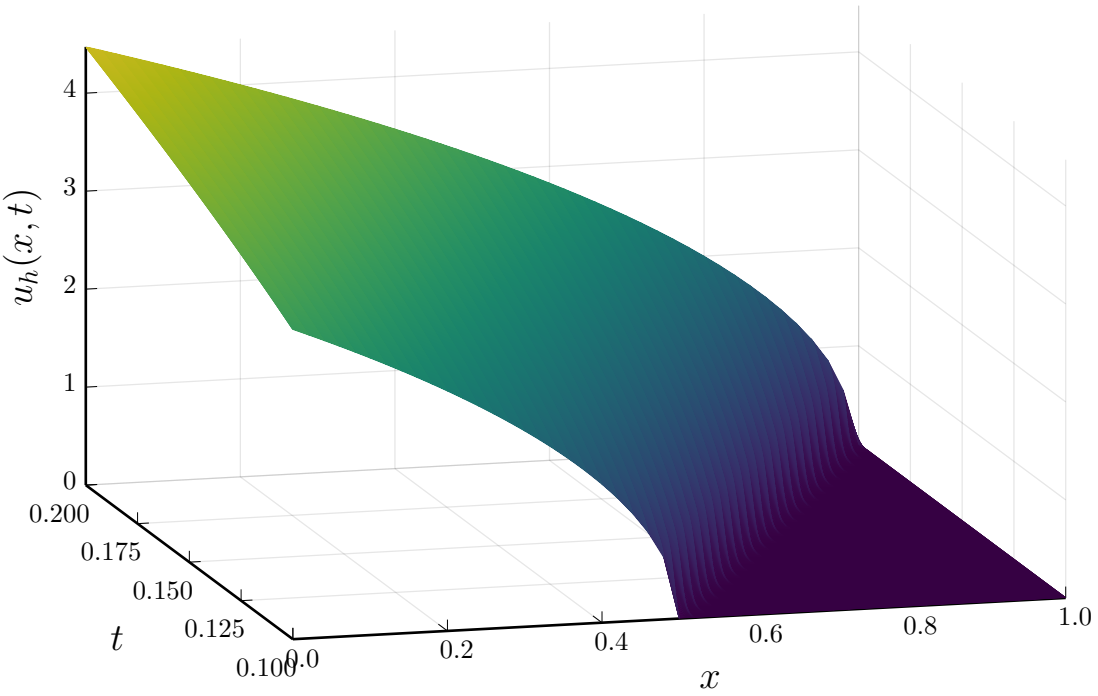

Рис. 9: Численное решение задачи (4) с параметрами сетки $N_x = 50$, $N_t = 500$

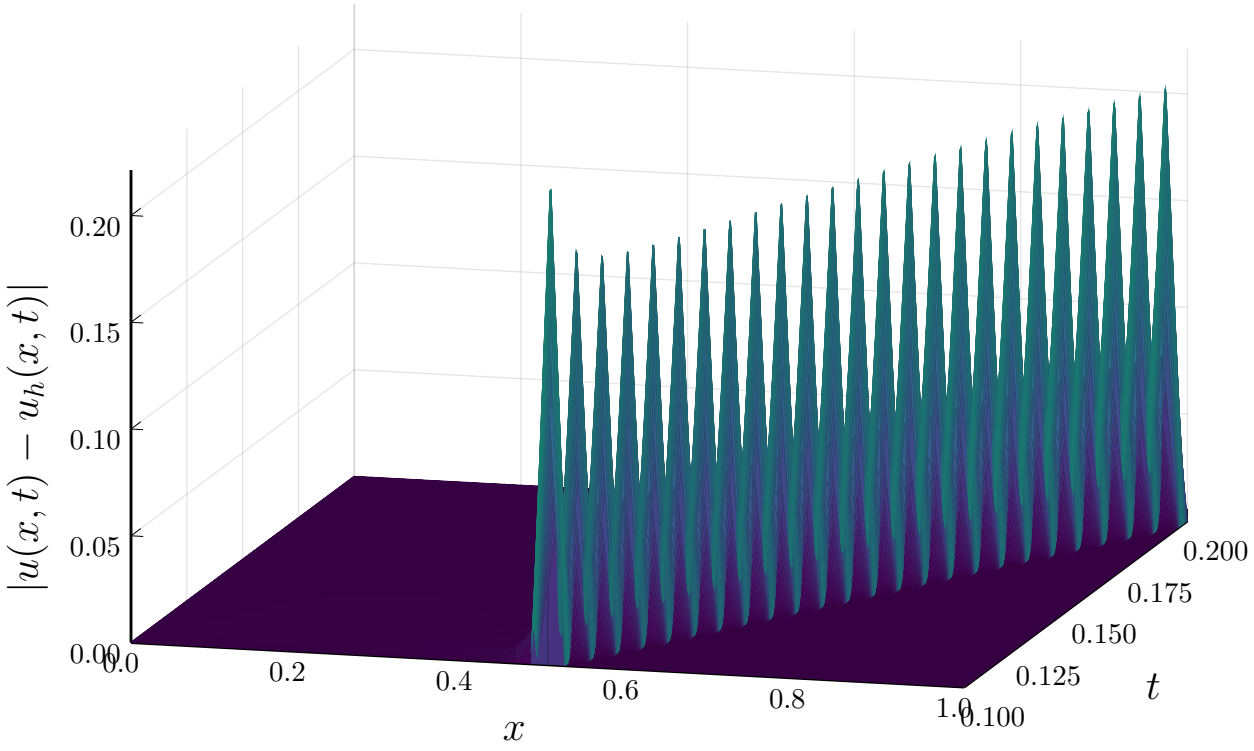

Рис. 10: График локальной ошибки численного решения

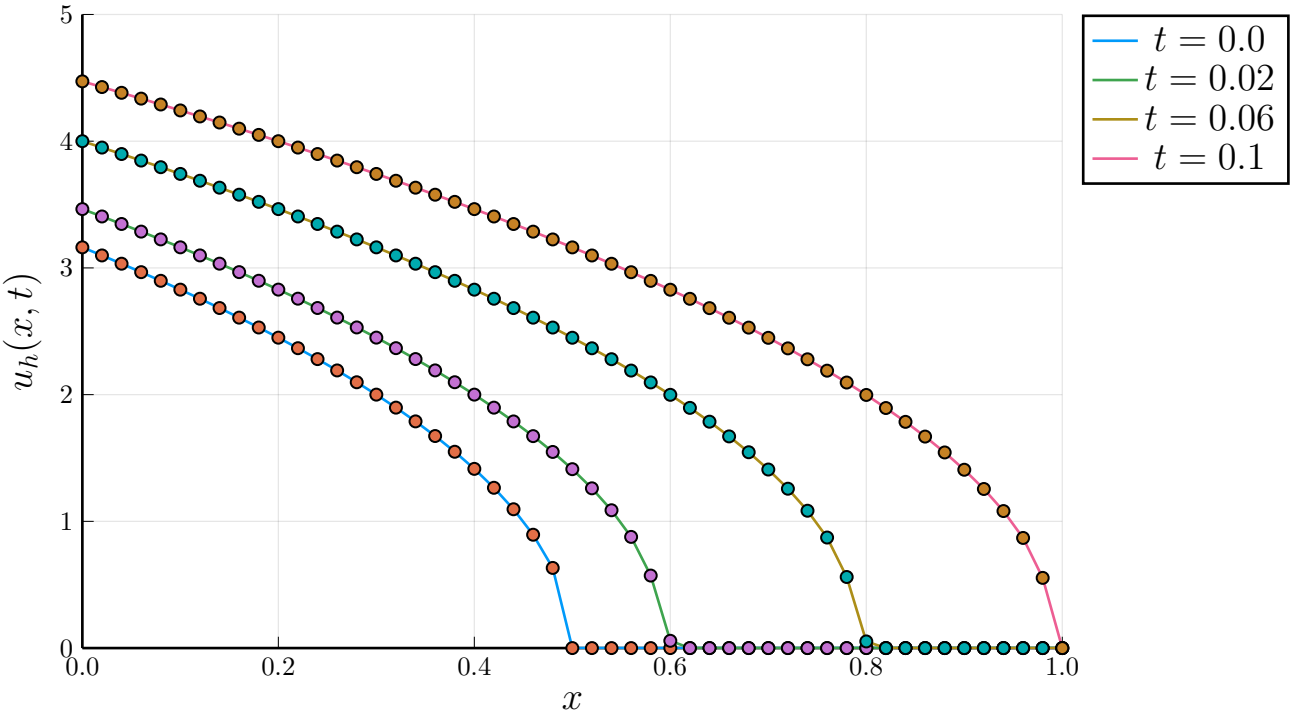

Рис. 11: Профили волны в некоторые моменты времени

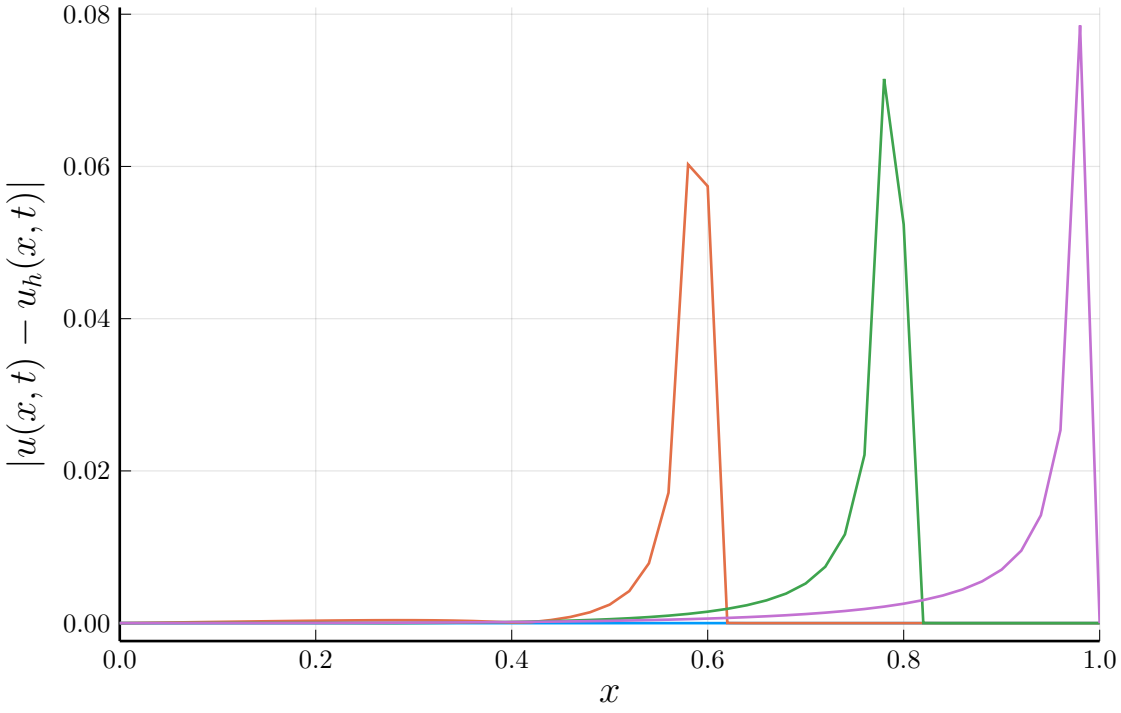

Рис. 12: Графики локальных погрешностей для некоторых моментов времени.

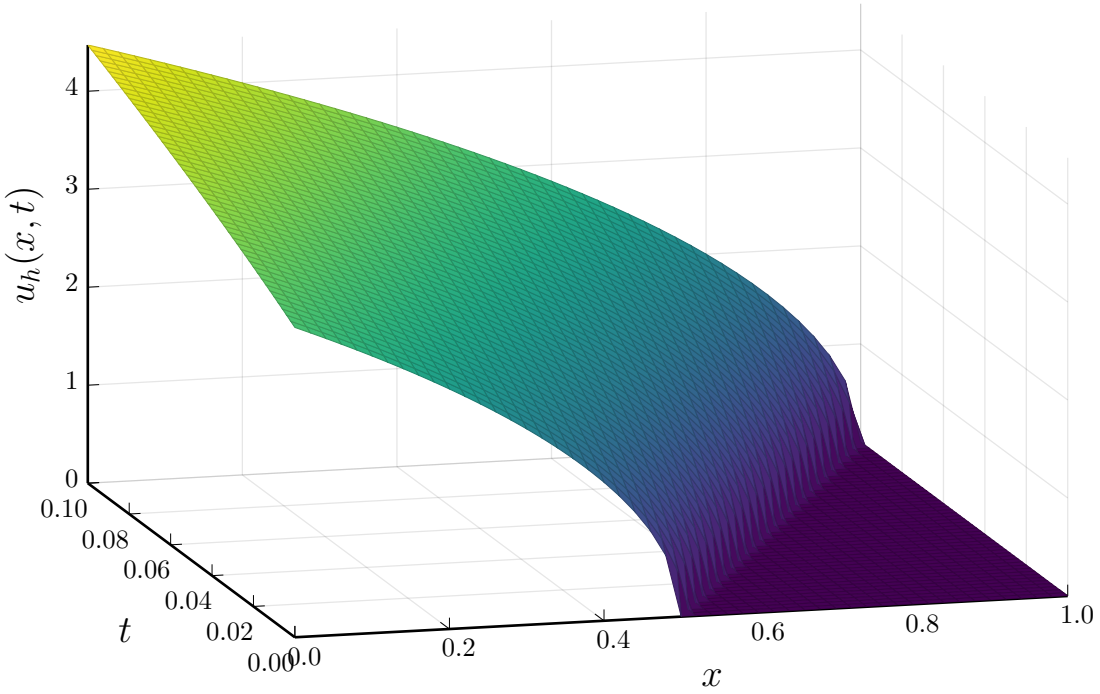

Рис. 13: Более точное решение задачи (4), полученное для параметров сетки $N_x = 5000$, $N_t = 5000$.

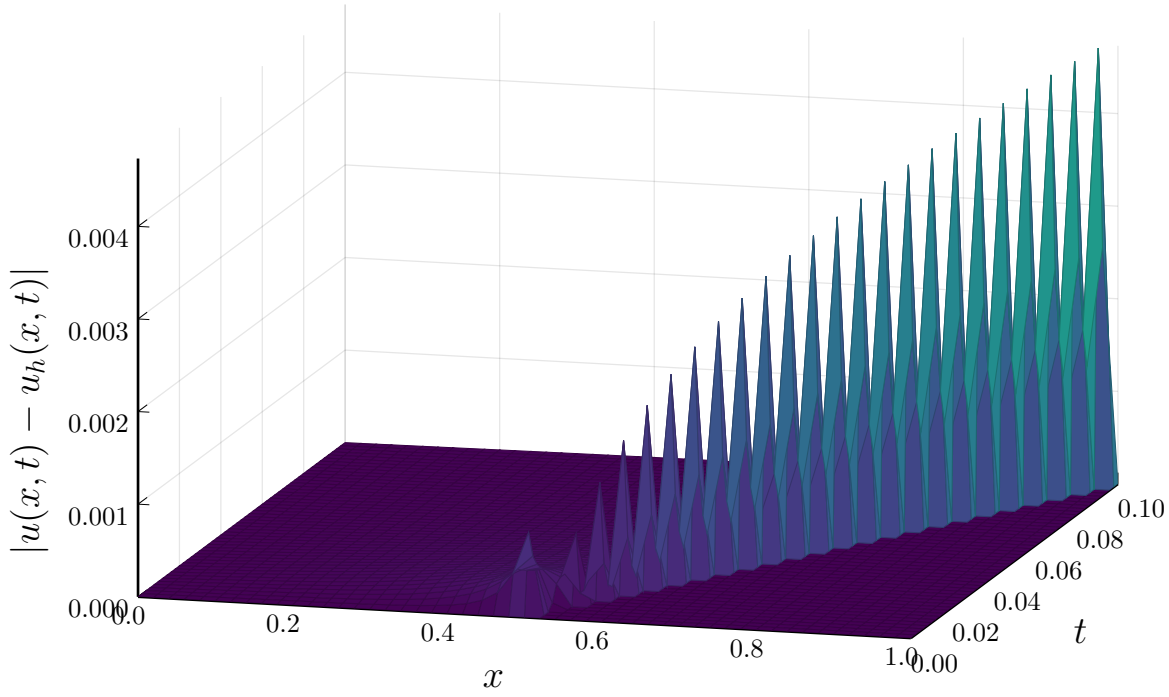

Рис. 14: График локальной погрешности, полученный для параметров сетки $N_x = 5000$, $N_t = 5000$. Видно, что относительно все ошибки сосредоточены в окрестности фронта волны.

Видно, что основная погрешность сосредоточена в точках *температурного фронта волны*: всюду, кроме некоторых точек около фронта, в которых производные решения терпят разрыв, отклонение приближённого решения от аналитического не превосходит $2 \cdot 10^{-3}$. Для достижения такой же точности и в *вблизи особых точек* решения необходима достаточно мелкая сетка. Были выбраны параметры $N_x = 5000$, $N_t = 5000$, график численного решения приведён на рис. 13. Однако из рис. 14 видно, что даже с настолько мелкой сеткой ошибки в близи особых точек решения остаются весьма заметными. Более того, решение во всей остальной области оказывается *на порядки* точнее решения вблизи особых точек. Дальнейшее измельчение сетки будет приводить к огромным, бесполезным расчётным затратам.

Данный пример хорошо демонстрирует причину возникновения метода локально-адаптивных сеток. В одномерном случае с одной особой точкой эта ситуация может показаться не столь плохой. Для многомерных же задач с большим числом особенностей и сильно меняющимися масштабами решения, где для получения заданной точности приходится использовать минимально необходимый шаг сетки во всей области сразу, число ненужных операций алгоритма значительно возрастает до недопустимых значений.

Как уже отмечалась, в задачах с размерностью $p > 1$ получающиеся в неявной схеме системы не будут трёхдиагональными, однако по-прежнему останутся достаточно разреженными. Реализация такой схемы заключается в прямом создании матрицы системы на каждом временном слое и решении с помощью общего алгоритма исключения переменных Гаусса.

Локально-одномерная схема в сущности решает N^{p-1} одномерных задач, поэтому её реализация похожа на неявную схему, но с некоторыми отличиями. В листинге 4 приведёт код этой схемы для двумерного квазилинейного уравнения теплопроводности (на случай больших размерностей код модифицируется очевидным образом).

```julia
function local_scheme(
    k1, k2, f,
    u0, mu_1, mu1, mu_2, mu2,
    Lx, Ly, T, Nx, Ny, Nt
    )

    # Создание сетки
    x = range(0, Lx, length = Nx); dx = step(x)
    y = range(0, Ly, length = Ny); dy = step(y)
    t = range(0, T, length = Nt);  dt = step(t)

    u = zeros(Nx, Ny, Nt) # Массив, хранящий решение

    # Временные переменные для прогонки
    A, C = (Tuple(zeros(N - 1) for N in [Nx Ny]) for i in 1:2)
    B, D, a, b, xt = (Tuple(zeros(N) for N in [Nx Ny]) for i in 1:5)
    B[1][1] = B[1][end] = B[2][1] = B[2][end] = 1
    C[1][1] = C[2][1] = A[1][end] = A[2][end] = 0
    w = zeros(Nx, Ny)

    for i in 1:Nx, j in 1:Ny u[i, j, 1] = u0(x[i], y[j]) end # н.у.
    for j in 1:(Nt - 1) # Цикл по временным слоям
        for k in 1:Ny # Сначала решаем задачу вдоль оси x для всех y_k
            for i in 2:(Nx - 1) # Создание трёхдиагональной матрицы
                A[1][i - 1] = -k1(0.5 * (u[i - 1, k, j] + u[i, k, j])) / dx^2
                B[1][i] = 1/dt + (
                    k1(0.5 * (u[i - 1, k, j] + u[i, k, j])) +
                    k1(0.5 * (u[i + 1, k, j] + u[i, k, j]))
                ) / dx^2
                C[1][i] = -k1(0.5 * (u[i + 1, k, j] + u[i, k, j])) / dx^2
                D[1][i] = u[i, k, j] / dt + 0.5 * f(x[i], y[k], t[j] + 0.5 * dt)
            end
            D[1][1] = mu_1(y[k], t[j] + 0.5 * dt); D[1][end] = mu1(y[k], t[j] + 0.5 * dt)
            tridiagonal_algorithm!(a[1], b[1], xt[1], A[1], B[1], C[1], D[1])
            w[:, k] .= xt[1]
        end
        for i in 1:Nx # Теперь решаем задачу вдоль оси y для всех x_i
            for k in 2:(Ny - 1) # Вся суть далее та же самая
                A[2][k - 1] = -k2(0.5 * (w[i, k] + w[i, k - 1])) / dy^2
                B[2][k] = 1/dt + (
                    k2(0.5 * (w[i, k] + w[i, k - 1])) +
                    k2(0.5 * (w[i, k + 1] + w[i, k]))
                ) / dy^2
                C[2][k] = -k2(0.5 * (w[i, k] + w[i, k + 1])) / dy^2
                D[2][k] = w[i, k] / dt + 0.5 * f(x[i], y[k], t[j + 1])
            end
            D[2][1] = mu_2(x[i], t[j + 1]); D[2][end] = mu2(x[i], t[j + 1])
            tridiagonal_algorithm!(a[2], b[2], xt[2], A[2], B[2], C[2], D[2])
            u[i, :, j + 1] .= xt[2]
        end
    end
    return u
end
```

Listing 4: Локально-одномерная схема для двумерного уравнения теплопроводности $u_t = (k_1 u_x)_x + (k_2 u_y)_y + f$

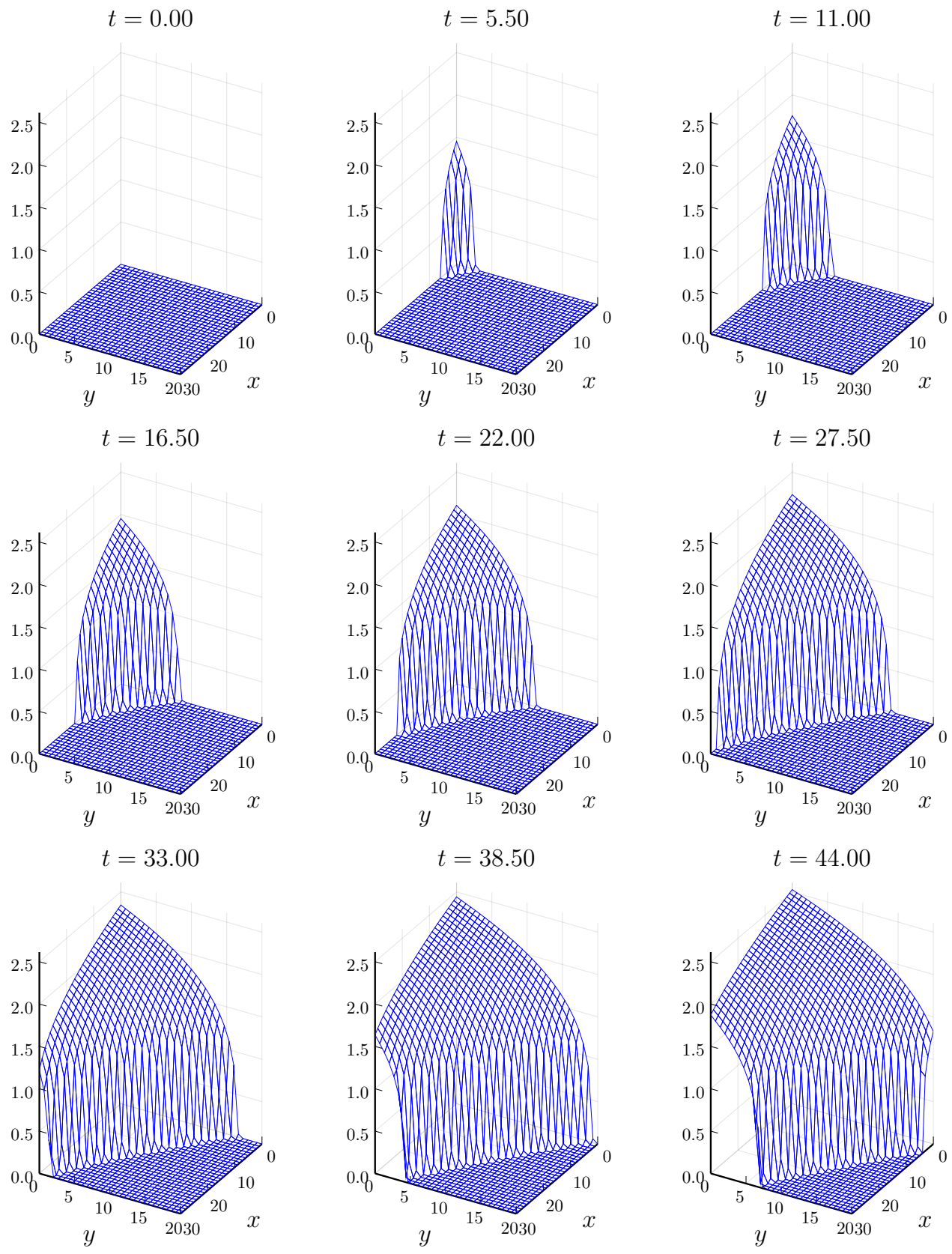

Рис. 15: Серия графиков зависимости численного решения $u_h(x, y, t)$ задачи (5) для фиксированных моментов времени $t \in [0, T]$.

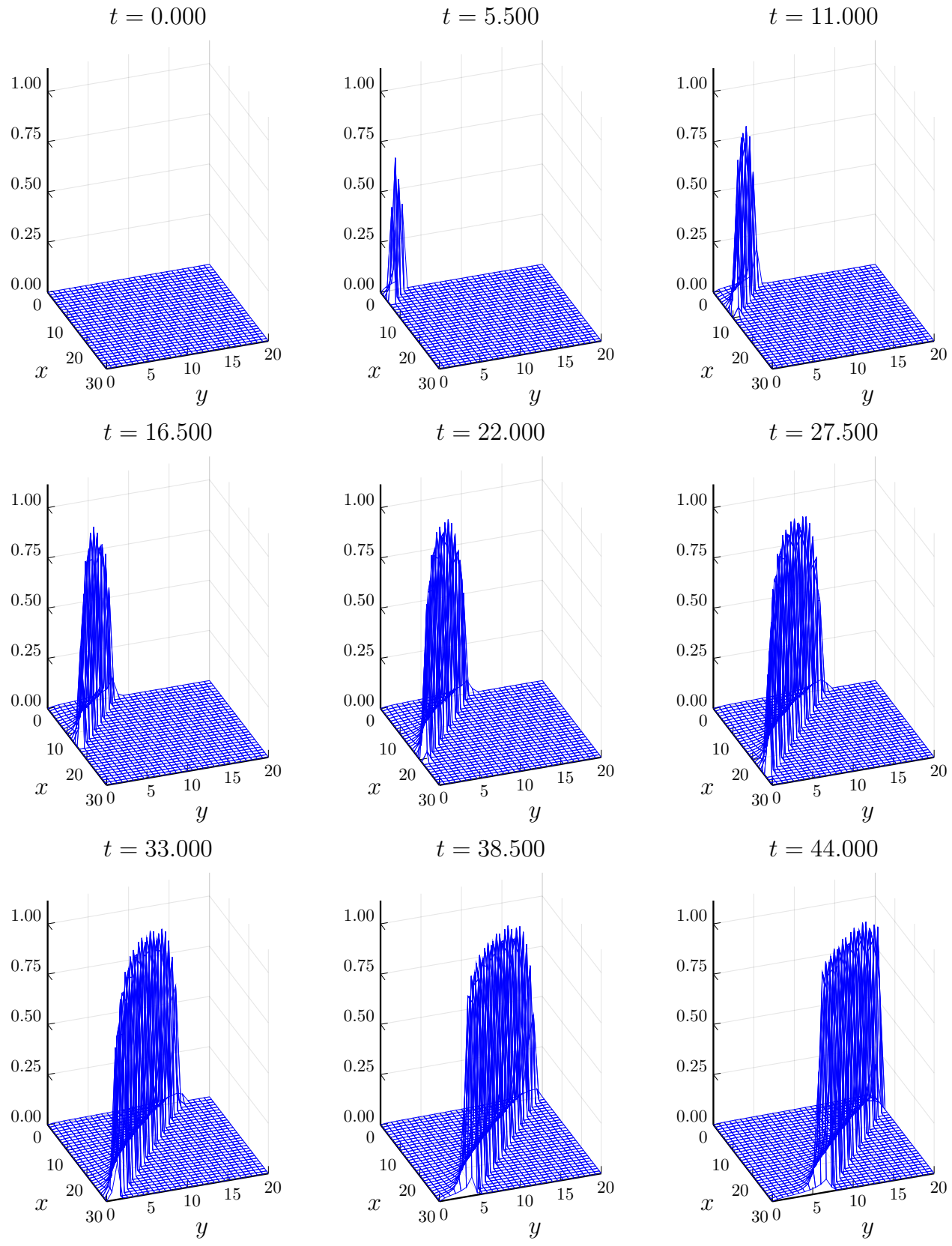

Рис. 16: Серия графиков зависимости $|u(x, y, t) - u_h(x, y, t)|$ для фиксированных моментов времени $t \in [0, T]$.

Работоспособность проверена на задаче, рассмотренной в статье [20]:

$$
\begin{cases}
u_t = (4u^4 u_x)_x + \left(\frac{1}{4}u^2 u_y\right)_y, \quad (x,y,t) \in (0,30)\times(0,20)\times(0,50) \\
u(x,y,0) = 0, \\
u(0,y,t) = \mu_{-1}(y,t) = \begin{cases} \frac{1}{2}\sqrt{-1+\sqrt{1+16(t-2y)}}, & t > 2y \\ 0, & t < 2y \end{cases} \\
u(30,y,t) = \mu_{1}(y,t) = \begin{cases} \frac{1}{2}\sqrt{-1+\sqrt{1+16(t-30-2y)}}, & t > 30+2y \\ 0, & t < 30+2y \end{cases} \\
u(x,0,t) = \mu_{-2}(x,t) = \begin{cases} \frac{1}{2}\sqrt{-1+\sqrt{1+16(t-x)}}, & t > x \\ 0, & t < x \end{cases} \\
u(x,20,t) = \mu_{2}(x,t) = \begin{cases} \frac{1}{2}\sqrt{-1+\sqrt{1+16(t-x-40)}}, & t > x+40 \\ 0, & t < x+40 \end{cases}
\end{cases}
\tag{5}
$$

На рис. 15 представлено численное решение задачи (5). Решение этой задачи также представляет собой вид бегущей волны с разрывом производных решения в точках фронта. Было проведено сравнение с аналитическим решением:

$$
u(x,y,t) = \begin{cases} \frac{1}{2}\sqrt{-1+\sqrt{1+16(t-x-2y)}}, & t \geqslant x+2y \\ 0, & t < x+2y \end{cases}
$$

На рис. 16 отражена локальная погрешность решения. Видно, что ситуация аналогична той, которая была описана выше для одномерного уравнения, а именно: погрешность в окрестности особых точек на порядки превышает погрешность в остальной области. Число этих точек, по сравнению с общим числом точек сетки невелико, а именно, отношение $\frac{N_{\text{особ.}}}{N_{\text{общ.}}} \to 0$ при $N_{\text{общ.}} \to 0$ Это утверждение прямо говорит о том, что просто измельчать равномерную сетку для достижения заданной точности в окрестности особых точек *нецелесообразно.*

3 Теория блочных локально-адаптивных сеток

Производить локальное измельчение сетки можно различными путями, основываясь на геометрии рассматриваемой области $G \subset \mathbb{R}^p$, геометрии мно-

жества особых точек уравнения, принятой неравномерной/криволинейной сетки. Подход, который используется в работе, основан на декартовых прямоугольных сетках. Такой выбор является оптимальным с точки зрения компьютерных вычислений — данные легче и удобнее всего хранятся и обрабатываются в областях прямоугольной формы, в то время как неструктурированные локальные измельчения приводят к нерегулярному доступу к памяти компьютера и, тем самым, к замедлению счёта. Сущность такого подхода можно описать следующим образом: исходная прямоугольная область $G = \prod_{k=1}^{p}[0, L_p]$ дискретизуется равномерной сеткой. Далее в соответствии с некоторыми критериями, выделяются подобласти, в которых содержатся особенности. Эти подобласти покрываются прямоугольными областями, которые в свою очередь дискретизуются более мелкой сеткой. Такой процесс продолжается до тех пор, пока нужные зоны „не покроются достаточно хорошо “, настолько, чтобы численное решение в этих областях достигало заданной точности. После выбора такой сетки производится численное интегрирование нескольких временных шагов. После такого обновления решения, его особые точки могут сместиться, соответственно, придётся перестраивать сетку под особенности. Данные переинтерполируются со старой сетки на новую, перестроенную, и процесс продолжается снова.

Основная теория черпалась из работ [11; 12; 21]. Однако стоит отметить следующие обстоятельства. Во-первых, в этих статьях исследовался подход локально-адаптивных сеток применительно к уравнениям гиперболического типа. Это позволяло использовать явную вычислительную схему, в то время как для параболических систем это ведёт к определённым условиям на выбор временного и пространственного шага. Во-вторых, использовался так называемый клеточный подход (см. рис. 17), в котором область G представлялась в виде ячеек (cells) прямоугольной формы,

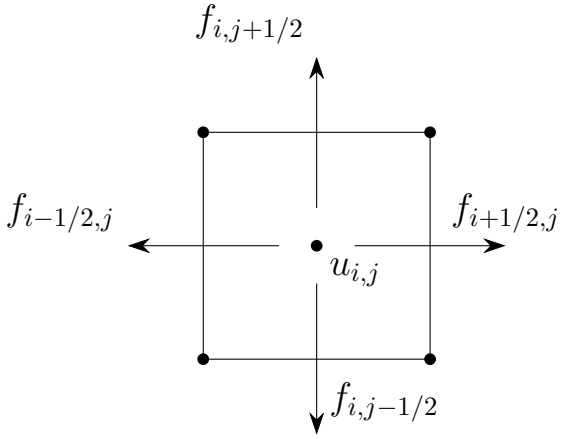

Рис. 17: Клеточный подход, векторы f — численные потоки

в цетре которых хранилось среднее значение решения (то есть использовался проекционный оператор вида $P_h = \frac{1}{\mu(C)} \int_C u(x)\,dx$), а также на границах ячейки хранилась информация о потоках (поток физ. величин, входящих в уравнение) $\vec{f} = \vec{f}(u)$. Затем эти величины использовались для модификации разностной схемы в соответствии с законом сохранения. Это привносит определённые трудности в реализацию данного алгоритма. В работе вместо явной схемы используется устойчивая, консервативная локально-одномерная схема, которая устраняет трудности с пересчётом потоков. Также в работе вместо клеточного формализма используется т.н. узельный, основные понятия которого были описаны выше.

3.1 Топология сетки

Далее для простоты и определённости рассматривается двумерный случай. На случай больших размерностей все определения непосредственно переносятся.

Адаптивная сетка состоит из последовательности *уровней* $l = 1, 2, \ldots, l_{\max}$. Каждый уровень представляет из себя набор прямоугольных областей $G_{l,m}$, где l нумерует уровень, m — номер прямоугольной области (блока) на данном уровне. Считаем, что на уровне l находится M_l блоков. Общее множество, занимаемое уровнем l есть

$$G_l = \bigcup_{m=1}^{M_l} G_{l,m}$$

Уровень $l = 1$ будем называть основным, главным или корневым. Этот уровень считаем состоящим из $M_1 = 1$ прямоугольной области $[0, L_x] \times [0, L_y]$, которая дискретизуется равномерной сеткой:

$$\omega_{1,1} = \{(x_i, y_j) \mid x_i = (i-1) \cdot h_x, y_j = (j-1) \cdot h_y, i = 1, \ldots, N_x, j = 1, \ldots, N_y\}.$$

Каждый последующий уровень дискретизуется более мелкой сеткой, а именно шаг и по оси x и по оси y при переходе с уровня $l-1$ на уровень l уменьшается в r_l раз. Чаще всего рассматривают $r_l = 2, 4$. В этой работе для определённости и простоты реализации рассматривается фиксированный $r_l = 2 \quad \forall l > 1$. Также на такую иерархию накладывается ограничение

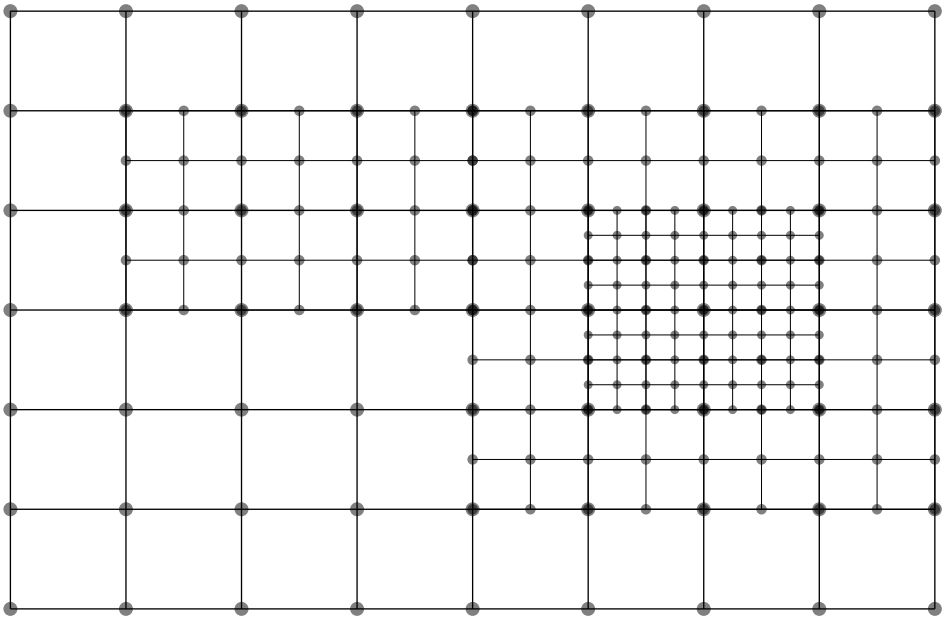

Рис. 18: Пример корректной блочно-структурированной декартовой сетки

корректно-вложенных уровней:

$$G_l \cap G_{l-1} = G_l,$$

то есть уровень $l+1$ должен целиком лежать в уровне l. (В ячеечном формализме это утверждение может быть сформулировано следующим образом: в окрестности каждой ячейки не должно быть ячеек, отличающихся от неё по размерам более чем в два раза). Множество точек на каждом блоке нумеруется (локально) индексной системой координат (i, j). Таким образом, уровень l должен хранить положения блоков $G_{l,1}, \ldots, G_{l,M_l}$ в индексных координатах надблока уровня $l-1$. Пример корректной блочно-стурктурированной декартовой сетки приведён на рис. 18

3.2 Численное обновление

Пускай известно решение на временном слое j, то есть известно $u^j_{i,k} \quad \forall i, k$. Решение на новом временном слое ищется постепенно, от грубого к мелкому, используя какую-либо ранее рассмотренную разностную схему для статических сеток. Как уже отмечалось, использование явной схемы нецелесообразно ввиду отсутствия устойчивости. В работах [22] используется явно-итерационная схема ЛИ-М, опирающаяся на оптимальные свойства много-

члена Чебышева специальной конструкции, о чём подробнее можно прочитать в [23—25] Предпочтение было отдано локально-одномерной схеме, ввиду всех её преимуществ, описанных ранее в разделе 2.1.3

Сначала обновляется решение на самой крупной сетке, то есть на уровне l. Затем ищется решение на более точной сетке. Так как в общем случае $\partial G_l \not\subset \partial G_1$, то сразу не понятно, как учитывать граничные условия в схеме обновления решения.

3.3 Граничные условия

Поступают следующим образом: при решении на уровне l в качестве граничных значений для обновления $t_j \mapsto t_{j+1}$ берутся уже найденные приближённые значения на слое t_{j+1} на уровне $l-1$ (то есть на предыдущем уровне). Значения в точках $\partial\omega_l \cap \omega_{l-1} \not\subset \omega_l$ задаются путём линейной интерполяции со значений в точках $\partial\omega_l \cap \omega_{l-1} \subset \omega_l$. Рисунок 19 иллюстрирует вышесказанное. Так, точки A, B принадлежат уровню $l-1$. Точки C, D, E принадлежат

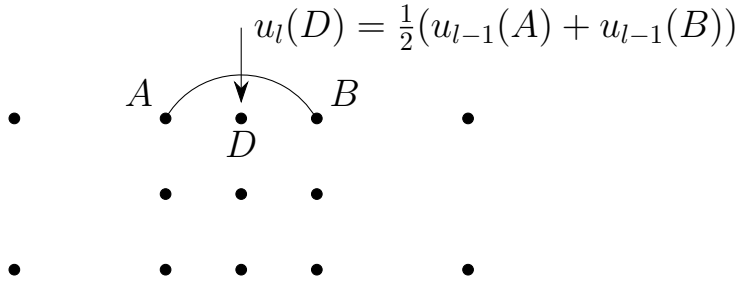

Рис. 19: Интерполяция с грубой сетки

уровню l. Точки C и E совпадают с точками A и B, и решение в них задаётся с уровня $l-1$, а значение в „промежуточной“точке D задаётся линейной интерполяцией значений с точек A и B:

$$u_l(D) = \frac{u_{l-1}(A) + u_{l-1}(B)}{2}$$

Для большей точности расчёта помимо измельчения пространственной сетки происходит и измельчение временной сетки, а именно:

$$\frac{\Delta x_l}{\Delta x_{l-1}} = \frac{\Delta t_l}{\Delta t_{l-1}}. \tag{6}$$

Это позволяет, например, избежать появления неустойчивости в случае использования явной схемы.

3.4 Данные на сетке

Очевидно использовать обыкновенные многомерные массивы для хранения данных численного решения на таких блочных сетках невозможно, а точнее, неэффективно. Задача отыскания наиболее удобного и оптимального формата хранения данных на сетках с локальной адаптацией — одна из ключевых. Существует несколько методов. Так, в [9] используются древовидные структуры (четвертичные и восьмеричные деревья). В этой работе для хранения данных используется двусвязный список. Каждый уровень `Level` описывается следующей структурой (несущественные в идеологическом плане тонкости реализации опускаются в коде):

```
struct Level
    level_number::Int # номер уровня в иерархии уровней
    sublevel::Union{Nothing, Level} # подуровень
    suplevel::Union{Nothing, Level} # надуровень

    blocks::Vector{Block} # массив блоков сеток
    M::Int # число блоков сетки

    t_curr::Real # Текущее время на уровне
    delta_t::Real # Временной шаг уровня
end
```

Поля `sublevel` и `suplevel` имеют тип `Union{Nothing, Level}`, что означает, что эти переменные могут быть как типа `Level` (то есть быть ссылкой на под- или над- уровни соответственно), так и типа `Nothing` (то есть не существовать, а значит, уровень не имеет подуровня (последний уровень измельчения) или надуровня (уровень является корневым)). Тип блок имеет вид:

```
struct Block
    spacial_grid::UniformGrid # Сетка
```

```
    # Координаты `((x1, x2), (y1, y2))` положения блока в
    # надблоке (в индексных координатах надблока)
    supblock_position::Tuple{Tuple{Int, Int}, Tuple{Int, Int}}

    u::Matrix{Real} # Решение
end
```

`UniformGrid` — удобное представление равномерной сетки, которая использовалась ранее.

3.5 Рекурсивный алгоритм

Основной алгоритм записывается достаточно просто после того, как учтено, каким образом хранятся данные, и реализована схема для равномерных сеток. Основные детали алгоритма могут быть записаны в следующем виде:

```
function solve(problem::HeatProblem, grid_params)
    # учёт начальных условий
    initial_conditions!(problem, grid_params)

    # Идём по всем временным слоям
    for j in 1:(grid_params.Nt - 1)
        levels[j + 1] = deepcopy(levels[j])
        advance_level!(problem, levels[j + 1])
    end
end
```

Функция `initial_conditions!` устанавливает значение в момент времени t_1, то есть для каждой точки (x, y) блочной сетки, для каждого уровня считается значение $u_0(x, y)$.

Затем выполняется послойное обновление решения: зная $u(x, y, t_j)$ для любых $(x, y) \in \omega$ (то есть зная решение на временном слое t_j), находится решение на следующем временном слое t_{j+1} с помощью функции `advance_level!`. Эта функция вызывается на коневом уровне $l = 1$, а затем рекурсивно на всех подуровнях. Это иллюстрирует листинг:

```
function advance_level!(problem::HeatProblem, level::Level)
    for i in 1:r_l # r_l --- refinement ratio
        numerical_update_u(problem, level)
        if has_sublevel(level)
            advance_level!(problem, level.sublevel)
        end
        level.t_curr += level.delta_t
    end
    interpolate_fine_to_coarse(level)
end
```

С помощью локально-одномерной схемы функция `numerical_update_u` обновляет решение на уровне `level`, и если уровень имеет подуровень (`has_sublevel(level) == true`), выполняется обновление на этом подуровне. Причём обновление на подуровне происходит r_l раз, то есть соответствует (6). В нашем случае $r_l = 2 \quad \forall l > 1$, что значит, что переход $t_j \mapsto t_{j+1}$ от старого временного слоя к новому на уровне l будет осуществляться за 2^{l-1} шагов.

Отметим ещё одну особенно важную часть алгоритма, требующую отдельного подробного исследования, и поэтому не рассмотренную в работе: перестройку сетки с учётом особенностей решения. После того как получено решение на новом временном слое, необходимо проанализировать полученное решение и найти новое положение особенностей решения. Запускается так называемый процесс индикации перестройки (refinement indication). Индикатор оценивает текущую точность численного решения и отмечает области сетки, в которых нужна большая/меньшая точность. Было предложено достаточно большое количество индикаторов (например, в работах [11; 12; 26]). После того как были отмечены точки, в которых требуется большая точность, запускается процесс перестройки сетки (regridding). Задача состоит в том, чтобы покрыть отмеченные точки блоками мелкой сетки таким образом, чтобы неотмеченных точек, попадающих в мелкую сетку было как можно меньше (то есть покрытие особенностей решения оптимальным образом, так называемый clustering algorithm). Также как и с индикаторами, существует множество различных алгоритмов кластеризации, с некоторыми вариантами можно ознакомиться в [12; 21].

3.6 Результаты моделирования

Тестирование реализованного алгоритма на блочно-структурированных сетках проводилось на задаче (5). Так как проводилось исследование работоспособности именно блочности сетки, а не её динамической адаптации, был выбран отрезок времени $t \in [40, 42]$. На рис. 20 представлена сетка, на которой производился расчёт. Из рисунка видно, что самая грубая сетка со-

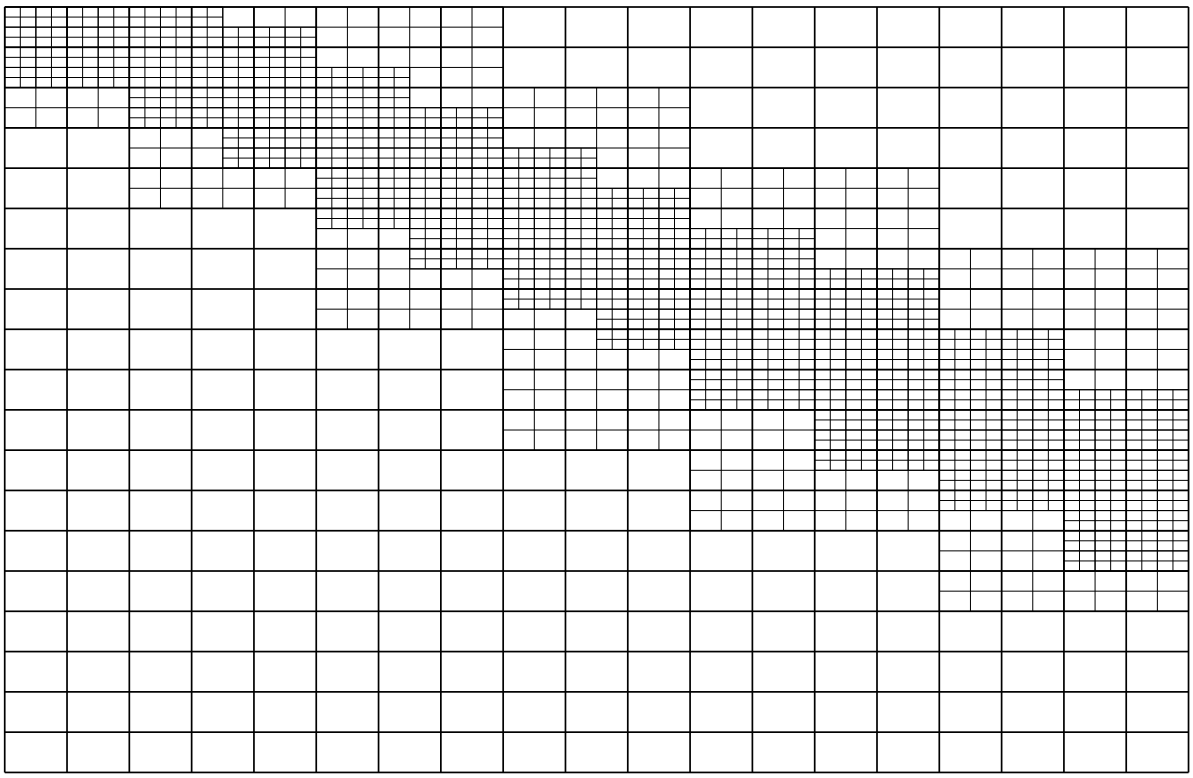

Рис. 20: Блочно-структурированная сетка, учитывающая особенность решения

ответствует выбору числа точек $N_x = 41$ и $N_y = 21$. График ошибок численного решения, полученного только на грубой сетке, приведён на рис. 21 График ошибок численного решения, полученного при использовании описанного выше алгоритма на структурированной сетке приведён на рис. 22 Из него непосредственно видно, что ошибка вблизи особенностей заметно уменьшилась. Также видно большее „размытие“окрестности особенности решения. Это указывает на то, что такая сетка „размывает“пики вблизи разрывов, делая погрешность решения более равномерной по всей области (что и соответствует тому, что мы находим решение с заданной точностью во всей области сразу). Также сравнивалось время, затраченное на расчёт на такой

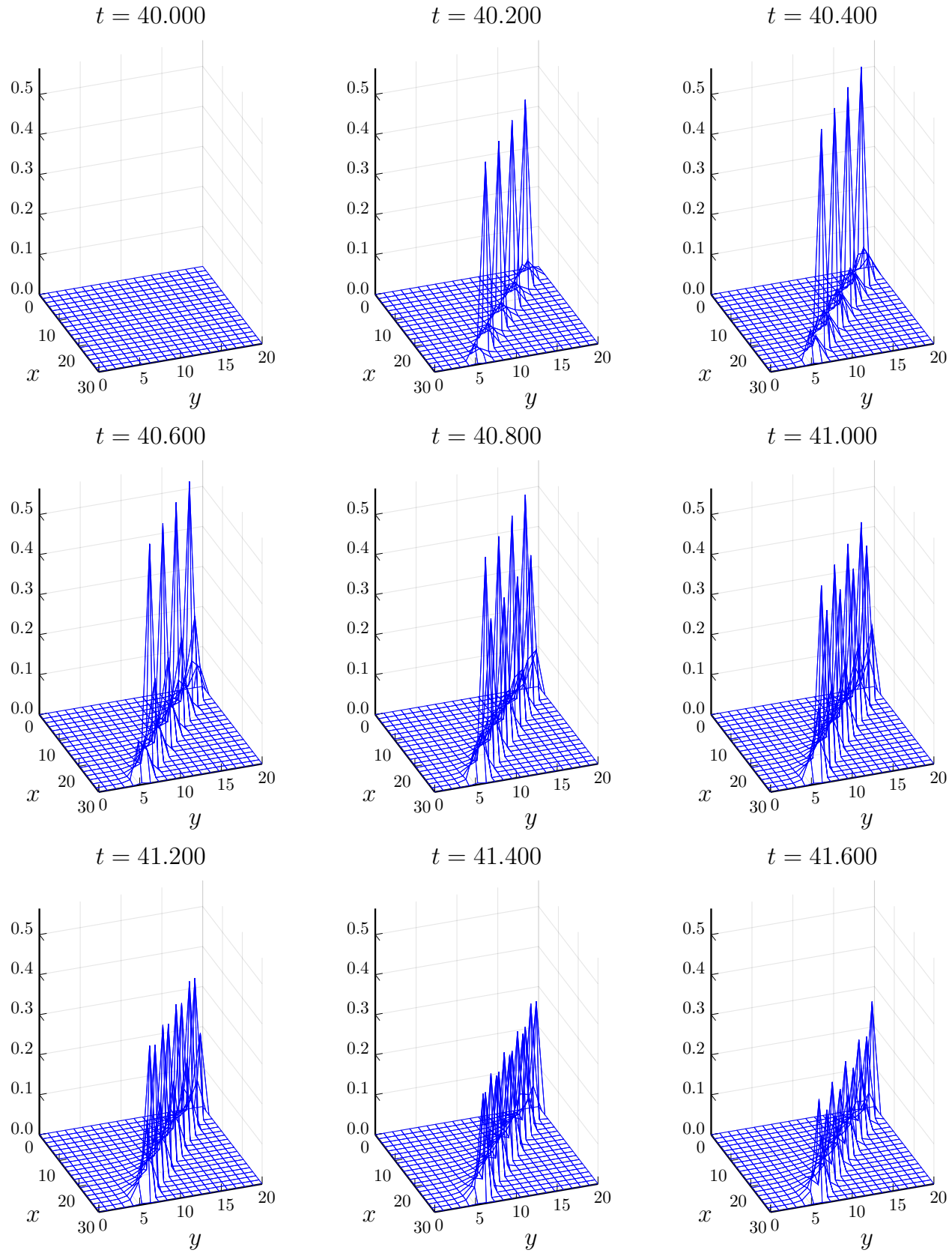

Рис. 21: График ошибок численного решения на грубой сетке $N_x = 41, N_y = 21$.

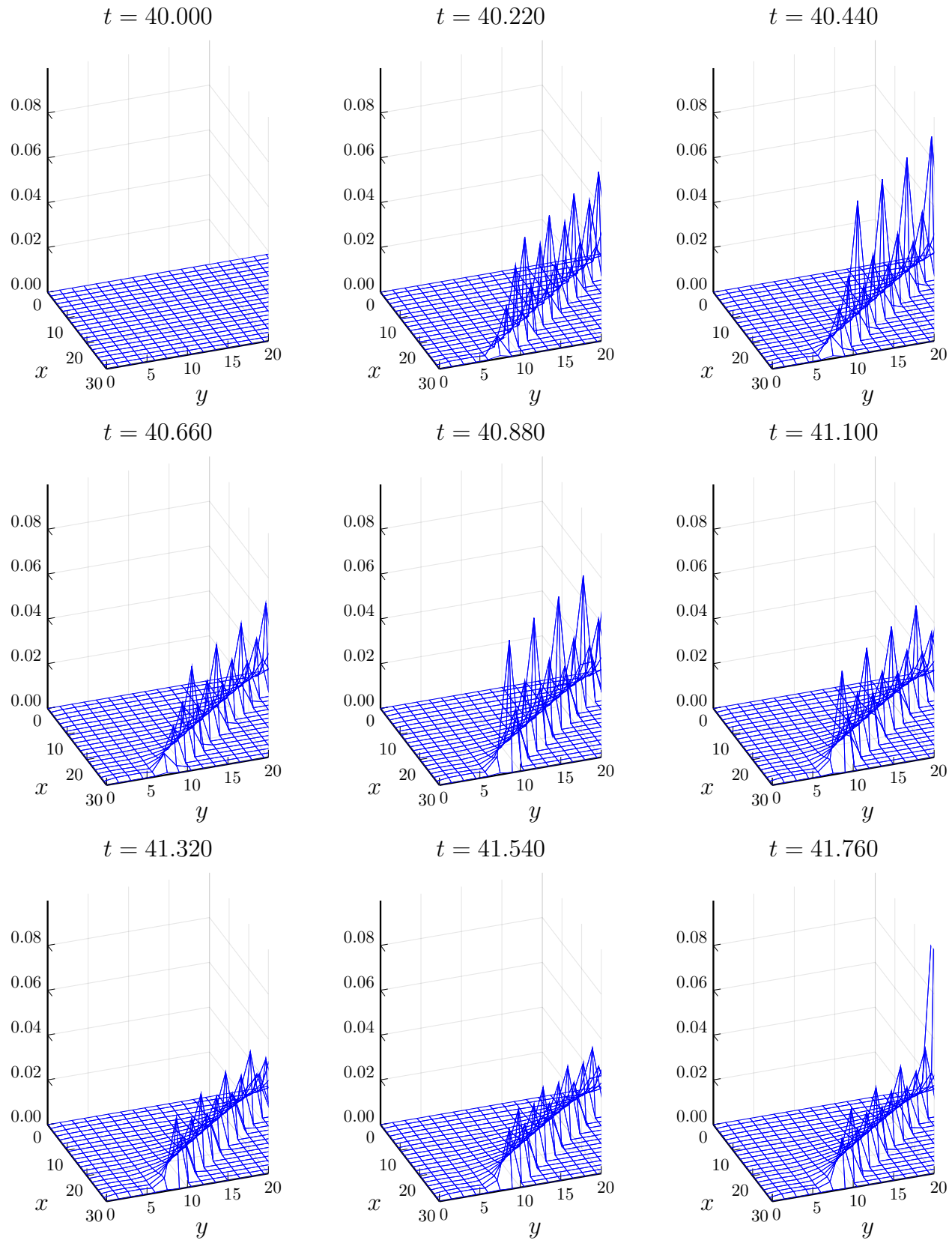

Рис. 22: График ошибок численного решения на блочно-структурированной сетке с тремя уровнями измельчения

блочно-структурированной сетке и на равномерной сетке, соответствующей самому мелкому разбиению из блочно-структурированной сетки: производительность первой программы оказалась $\approx$ в 10 раз выше. Таким образом, показаны основные достоинства реализованного алгоритма.

В процессе разработки и тестирования были найдены ошибки и недочёты программы, а именно, при достаточно большом количестве блоков и при их небольшом размере возможно возникновение так называемых артефактов решения (см. рис. 23), появляющихся на стыках двух блоков одного уровня. Скорее всего, данная особенность возникает из-за не совсем корректного учёта синхронизации данных на двух блоках. Точная причина и пути её устранения ещё подлежат дальнейшему анализу.

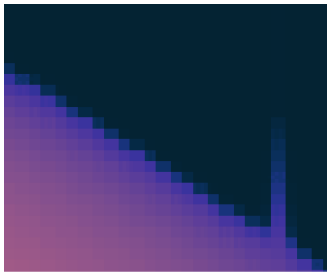

Рис. 23: Недочёты алгоритма

4 Заключение

В работе был проведён подробный анализ некоторых разностных схем численного решения уравнения теплопроводности, исследование их устойчивости, точности и скорости счёта, а именно:

- явной разностной схемы
- однопараметрического семейства неявных разностных схем, в том числе чисто неявной схемы для квазилинейного уравнения
- локально-одномерной схемы

Проделанный анализ сопровождался проведением большого количества численных расчётов с помощью написанного программного обеспечения, реализующего все вышеперечисленные разностные схемы и позволяющего решать задачу Дирихле для одномерного и многомерного уравнения теплопроводности с произвольными граничными и начальными данными, а также позволяющего демонстрировать точность и скорость счёта разностных схем. Показаны преимущества и недостатки использования равномерных статических сеток на конкретных примерах из различных статей.

Сделан обзор литературы, посвящённой методу блочно-структурированных локально-адаптивных сеток (block-structured adaptive mesh refinement), разработан алгоритм решения квазилинейного уравнения теплопроводности на блочно-структурированной сетке с использованием локально-одномерной схемы для обеспечения устойчивости метода, который реализован в виде программного кода. Исследована работоспособность алгоритма, показаны преимущества использования локального измельчения сетки.

Разработка численных методов локальной адаптации для решения прикладных задач является актуальной и неослабевающей научной проблемой, привлекающей как исследователей из области вычислительной математики, так и физиков экспериментаторов благодаря возможности поставить дорогостоящие эксперименты виртуально на компьютере. В настоящее время проблема полностью не решена и в научном мире ведутся активные исследования на эту тему.

В будущем планируется разработка программного обеспечения с переносом функционала перестройки сеток, реализованного в [27—29] для гипер-

болических проблем, на случай параболических уравнений, областей произвольной формы и возможностью распараллеливания программы.

Список литературы

1. *Тихонов А.*, *Самарский А.* Уравнения математической физики.–изд. 8-е, стереотипное. — 2007.
2. *Ландау Л.*, *Лифшиц Е.* Теоретическая физика: гидродинамика. 5-е изд., стереот // М.: Физмат-лит. — 2001. — Т. 5. — С. 736.
3. Квазилинейное уравнение теплопроводности с источником: обострение, локализация, симметрия, точные решения, асимптотики, структуры / В. А. Галактионов [и др.] // Итоги науки и техники. Серия «Современные проблемы математики. Новейшие достижения». — 1986. — Т. 28, № 0. — С. 95—205.
4. *Самарский А. А.* Теория разностных схем. — "Наука,"Глав. ред. физико-математической лит-ры, 1989.
5. *Самарский А. А.* Локально-одномерные разностные схемы на неравномерных сетках // Журнал вычислительной математики и математической физики. — 1963. — Т. 3, № 3. — С. 431—466.
6. *Зельдович Я.*, *Компанеец А.* К теории распространения тепла при теплопроводности, зависящей от температуры // Сборник, посвященный. — 1950. — Т. 70. — С. 61—71.
7. *Баренблатт Г.* О некоторых неустановившихся движениях жидкости и газа в пористой среде // Прикл. матем. и мех.—1952.—16. — 1952. — № 1. — С. 67—78.
8. *Баренблатт Г.*, *Вишик М.* О конечной скорости распространения в задачах нестационарной фильтрации жидкости и газа // Прикладная математика и механика. — 1956. — Т. 20, № 3. — С. 411—417.
9. Метод адаптивных декартовых сеток для решения задач газовой динамики / А. Афендиков [и др.]. — "Российская академиня наук", 2017.
10. *Дарьин Н. А.*, *Мажукин В. И.*, *Самарский А. А.* Конечно-разностный метод решения уравнений газовой динамики с использованием адаптивных сеток, динамически связанных с решением // Журнал вычислительной математики и математической физики. — 1988. — Т. 28, № 8. — С. 1210—1225.

11. *Berger M.* Adaptive mesh refinement for hyperbolic partial differential equations[P D. Thesis]. — 1982.

12. *Berger M. J.*, *Colella P.* Local adaptive mesh refinement for shock hydrodynamics Journal of computational Physics. — 1989. — Т. 82, № 1. — С. 64—84.

13. *Peaceman D. W.*, *Rachford Jr H. H.* The numerical solution of parabolic and elliptic differential equations // Journal of the Society for industrial and Applied Mathematics. — 1955. — Т. 3, № 1. — С. 28—41.

14. *Douglas Jr J.* On the Numerical Integration of $u_{xx} + u_{yy} = u_t$ by Implicit Methods // Journal of the society for industrial and applied mathematics. — 1955. — Т. 3, № 1. — С. 42—65.

15. *Яненко Н.* Об одном разностном методе счета многомерного уравнения теплопроводности // Докл. АН СССР. Т. 125. — 1959. — С. 1207.

16. *Дьяконов Е. Г.* Разностные схемы с расщепляющимся оператором для многомерных нестационарных задач // Журнал вычислительной математики и математической физики. — 1962. — Т. 2, № 4. — С. 549—568.

17. *Самарский А. А.* Об одном экономичном разностном методе решения многомерного параболического уравнения в произвольной области // Журнал вычислительной математики и математической физики. — 1962. — Т. 2, № 5. — С. 787—811.

18. Julia: A fresh approach to numerical computing / J. Bezanson [и др.] // SIAM review. — 2017. — Т. 59, № 1. — С. 65—98.

19. *Горюнов А. Ф.* Методы математической физики в примерах и задачах. — 2015.

20. *Самарский А. А.*, *Соболь И. М.* Примеры численного расчета температурных волн // Журнал вычислительной математики и математической физики. — 1963. — Т. 3, № 4. — С. 702—719.

21. *Deiterding R.* Block-structured adaptive mesh refinement-theory, implementation and application // Esaim: Proceedings. Т. 34. — EDP Sciences. 2011. — С. 97—150.

22. Численное решение параболических уравнений на локально-адаптивных сетках чебышевским методом / В. Т. Жуков [и др.] // Препринты Института прикладной математики им. МВ Келдыша РАН. — 2015. — № 0. — С. 87—26.

23. *Жуков В. Т.* О явных методах численного интегрирования для параболических уравнений // Математическое моделирование. — 2010. — Т. 22, № 10. — С. 127—158.

24. *Жуков В. Т.*, *Новикова Н. Д.*, *Феодоритова О. Б.* О решении эволюционных уравнений многосеточным и явно-итерационным методами // Журнал вычислительной математики и математической физики. — 2015. — Т. 55, № 8. — С. 1305—1319.

25. Параллельный многосеточный метод: сравнение эффективности на современных вычислительных архитектурах / В. Т. Жуков [и др.] // Препринты Института прикладной математики им. МВ Келдыша РАН. — 2014. — № 0. — С. 31—22.

26. *Löhner R.* An adaptive finite element scheme for transient problems in CFD // Computer methods in applied mechanics and engineering. — 1987. — Т. 61, № 3. — С. 323—338.

27. Adaptive numerical simulations with Trixi.jl: A case study of Julia for scientific computing / H. Ranocha [и др.] // Proceedings of the JuliaCon Conferences. — 2022. — Т. 1, № 1. — С. 77. — DOI: 10.21105/jcon.00077. — arXiv: 2108.06476 [cs.MS].

28. A purely hyperbolic discontinuous Galerkin approach for self-gravitating gas dynamics / M. Schlottke-Lakemper [и др.] // Journal of Computational Physics. — 2021. — Июнь. — Т. 442. — С. 110467. — DOI: 10.1016/j.jcp.2021.110467. — arXiv: 2008.10593 [math.NA].

29. Trixi.jl: Adaptive high-order numerical simulations of hyperbolic PDEs in Julia / M. Schlottke-Lakemper [и др.]. — 09.2021. — DOI: 10.5281/zenodo.3996439. — https://github.com/trixi-framework/Trixi.jl.