ФИЛИАЛ
МОСКОВСКОГО ГОСУДАРСТВЕННОГО УНИВЕРСИТЕТА
имени М. В. ЛОМОНОСОВА
в городе БАКУ

ФИЗИЧЕСКИЙ ФАКУЛЬТЕТ

Д. В. Лукьяненко, А. А. Панин

ЧИСЛЕННЫЕ МЕТОДЫ ДИАГНОСТИКИ РАЗРУШЕНИЯ РЕШЕНИЙ УРАВНЕНИЙ МАТЕМАТИЧЕСКОЙ ФИЗИКИ

учебно-методическое пособие
(с примерами на языке MatLab)

Баку
Физический факультет
Бакинского филиала МГУ
2018

Лукьяненко Д. В., Панин А. А.
Численные методы диагностики разрушения решений уравнений математической физики (с примерами на языке MatLab): учебно-методическое пособие. — Баку: Физический факультет Бакинского филиала МГУ, 2018. 125 с.
ISBN 978-9952-8353-1-1

В пособии изложены базовые численные методы диагностики разрушения решений уравнений математической физики, с примерами их программной реализации на языке программирования MatLab.

Материал пособия используется в курсе «Численные методы», который один из авторов читает для студентов бакалавриата и магистратуры кафедры математики физического факультета МГУ и студентов бакалавриата физического факультета Бакинского филиала МГУ.

Данный материал представляет интерес для широкого круга студентов и аспирантов, специализирующихся в области разрушения решений уравнений математической физики.

Библиогр. 12 назв., рис. 21.

Рекомендовано Советом отделения прикладной математики в качестве пособия для студентов физического факультета МГУ имени М. В. Ломоносова, обучающихся по направлениям «Физика» и «Астрономия».

ISBN 978-9952-8353-1-1

Оглавление

Предисловие 5

1 Диагностика разрушения решения задачи Коши для ОДУ первого порядка 7
1.1 Поиск численного решения 9
1.2 Численная диагностика разрушения решения . . . 12
1.3 Практические задания 28

2 Диагностика разрушения решения начально-краевой задачи для уравнения в частных производных псевдопараболического типа 30
2.1 Поиск численного решения 31
2.2 Оптимизация численного счёта 41
2.3 Контроль правильности реализации численного алгоритма в виде программного кода 47
2.4 Численная диагностика разрушения решения . . . 52
2.5 Практические задания 71

3 Диагностика разрушения решения начально-краевой задачи для уравнения в частных производных псевдогиперболического типа 73
3.1 Поиск численного решения 74
3.2 Оптимизация численного счёта 85
3.2.1 Использование вырожденного метода Гаусса 87
3.2.2 Использование разреженных матриц 95

3.3 Численная диагностика разрушения решения . . . 99
3.4 Построение схемы со 2-м порядком точности по времени 112
3.5 Практические задания 120
Литература 124

Предисловие

Данное пособие обобщает наработки авторов по численно-аналитическому исследованию явления разрушения решения (blow-up), а также использует многолетний опыт чтения курса «Численные методы» первым автором на кафедре математики отделения прикладной математики физического факультета МГУ имени М. В. Ломоносова. Авторы убеждены, что эффективное изложение любых численных методов невозможно без сопровождения теоретического материала практическими заданиями, выполнение которых студентами позволит прочувствовать все преимущества и недостатки рассматриваемых численных подходов, а также приобрести и закрепить соответствующие практические навыки. В связи с этим ключевой особенностью данной учебно-методической работы является сопровождение рассматриваемых численных подходов примерами их программной реализации на языке программирования MatLab, которые позволят учащимся не только успешно выполнить предлагаемые в конце каждого раздела практические задания, но и применить рассмотренные методы для решения своих собственных задач, возникающих в повседневной научной работе.

В пособии изложен метод численного исследования разрушения решений задач математической физики, основанный на оценке эффективного порядка точности по Ричардсону, основные идеи которого предложены в [2, 9]. Впервые о возможности диагностики разрушения таким методом авторы узнали из [8,

глава 8].

Пособие разделено на три главы по принципу увеличения сложности материала. В первой главе рассматриваются задачи Коши для обыкновенных дифференциальных уравнений. Во второй — метод распространяется на простые начально-краевые задачи для дифференциальных уравнений первого порядка по времени. В третьей рассматриваются более сложные начально-краевые задачи (уравнения второго порядка по времени, уравнения с нелинейностью под знаком производной по времени и т.д.). Подробно обсуждаются вопросы, связанные с повышением эффективности численной реализации методов (экономичные способы обращения матриц, рациональная организация промежуточных вычислений и хранения данных и т.п.). Описание всех методов сопровождается примерами их программной реализации на языке программирования MatLab.

Материал пособия используется в курсе «Численные методы», которые первый автор читает для студентов бакалавриата и магистратуры кафедры математики физического факультета МГУ и студентов бакалавриата физического факультета Бакинского филиала МГУ.

Основной целью пособия является предоставление читателям инструментария, который позволит эффективно применять изложенный материал (включая программный комплекс) для решения других современных задач, возникающих в научной работе. Также авторы выражают надежду, что выбранный ими стиль изложения материала позволит читателям обобщить предложенные методы для использования при решении более сложных задач, в том числе и в многомерных постановках.

1| Диагностика разрушения решения задачи Коши для ОДУ первого порядка

Рассмотрим следующую задачу, которая является простейшей моделью тепловыделения в одной из задач химической кинетики:

$$\begin{cases} \dfrac{du}{dt} = u^2, \quad t \in (t_0, T], \\ u(t_0) = u_0 > 0. \end{cases} \tag{1.1}$$

В качестве тестовой будем рассматривать задачу (1.1) для следующего набора параметров:

$$t_0 = 0, \quad T = 2, \quad u_0 = 1. \tag{1.2}$$

Функция, которая удовлетворяет уравнению и начальному условию из задачи (1.1), будет иметь вид

$$u(t) = \frac{1}{u_0 - t} \tag{1.3}$$

и для u_0 из набора параметров (1.2) изображена на рис. 1.1. Как видно на рисунке, функция (1.3) обращается в бесконечность за конечное время (в точке $t = 1$) и не определена в точке $t = 1$. Однако решением задачи Коши мы считаем лишь функцию, определённую на связном множестве (промежутке). Поэтому решением задачи (1.1) является ограничение функции (1.3) на $t \in [0, 1)$. О подобной ситуации говорят, что решение претерпевает разрушение (или «blow-up»).

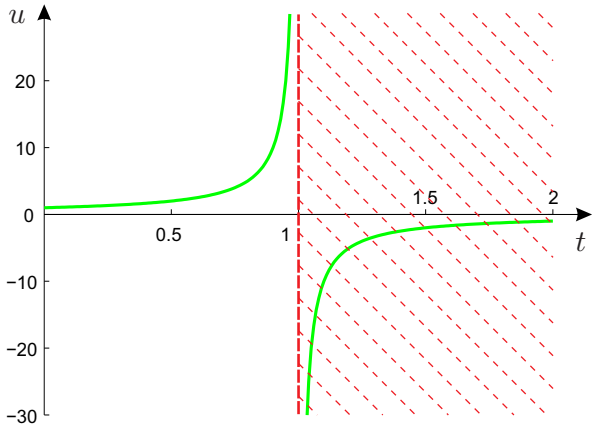

Рис. 1.1: Функция, которая удовлетворяет системе (1.1) для набора параметров (1.2). Штриховкой отмечена область, в которой функция $u(t)$ не является решением соответствующей задачи Коши.

Важно отметить, что в реальных задачах из всех областей науки ситуация, когда какой-либо параметр устремляется к бесконечности за конечное время, невозможна. Однако этот параметр может устремляться к бесконечности при решении соответствующей упрощённой математической модели, что в этом случае говорит о выходе за рамки применимости этой модели. Например, в случае задачи тепловыделения в химической реакции уход на бесконечность в окрестности $t = 1$ возможен только в предположении о неограниченном количестве реагирующего вещества, сосредоточенного в достаточно малом объёме. Очевидно, что такая ситуация на практике встретиться не может, так как количество реагирующего вещества всегда ограничено, что в итоге приведёт к прекращению хода реакции и, соответственно, прекращению тепловыделения в какой-то фиксированный момент времени $t < 1$ (при этом суммарное выделение тепла будет огра-

ничено). Поэтому если уточнить предметную модель рассматриваемой задачи и учесть как закон сохранения, так и скорость распространения реакции, то полученная математическая модель будет давать решение, которое существует во все моменты времени, начиная с начального. Однако на практике может оказаться невозможным или составление точной модели (ввиду сложности физического процесса, отсутствия точных данных и т. п.), или её решение, даже численное (ввиду чрезвычайной сложности уравнений). Поэтому нередко ограничиваются упрощённой моделью, описывающей характерные особенности процесса. Так, реальному взрыву (очень большому тепловыделению) может отвечать обращение решения приближённой модели в бесконечность.

Но даже упрощённые модели могут быть столь сложны, что единственным конструктивным подходом к их решению является использование численных методов для получения приближённого численного решения. И уже численное решение будет существовать для любых значений t (при надлежащем выборе аппроксимации, см. об этом замечание 2 на с. 19). Продемонстрируем это на примере рассматриваемой нами тестовой задачи (1.1)–(1.2).

1.1. Поиск численного решения

Перепишем задачу (1.1) в виде

$$\begin{cases} \dfrac{du}{dt} = f(u), \quad t \in (t_0, T], \\ u(t_0) = u_0, \end{cases} \tag{1.4}$$

где $f(u) = u^2$.

Введём равномерную сетку T_M по времени t с шагом $\tau = (T - t_0)/M$: $T_M = \{t_m,\ 0 \leqslant m \leqslant M :\ t_m = t_0 + m\tau\}$. Отметим, что эта сетка имеет $M+1$ узлов, или, что эквивалентно, M интервалов. В узлах сетки T_M введём обозначения для сеточных значений функции $u(t)$: $u_m \equiv u(t_m)$, $0 \leqslant m \leqslant M$.

Запишем семейство одностадийных схем Розенброка ROS1 [5, 10] для решения автономной системы (1.4):

$$
\begin{aligned}
&u_{m+1} = u_m + (t_{m+1} - t_m)\,\mathrm{Re}\, w_1\,, \quad 0 \leqslant m \leqslant M-1,\\
&\text{где } w_1 \text{ определяется из уравнения}\\
&\Bigl(1 - a_{11}\,(t_{m+1} - t_m)\,f_u(u_m)\Bigr) w_1 = f(u_m).
\end{aligned}
\tag{1.5}
$$

Здесь a_{11} — параметр, определяющий свойства схемы (подробности смотрите в п. 1.2.3 главы 1 [5]). В частности, при $a_{11} = 0$ схема (1.5) вырождается в явную одностадийную схему Рунге–Кутты ERK1 (схема Эйлера); при $a_{11} = 1$ — в обратную схему Эйлера DIRK1; при $a_{11} = 1/2$ — в схему «с полусуммой»; при $a_{11} = (1+i)/2$ — в одностадийную схему Розенброка с комплексным коэффициентом CROS1.

Далее приводится пример MatLab-функции, которая реализует поиск численного решения задачи (1.4) по схеме (1.5).

```
function u = ODESolving(t_0,T,M,u_0,f,f_u,a_11)

    % Функция находит приближённое численное решение
    % обыкновенного дифференциального уравнения (ОДУ/ODE)

    % Входные параметры:
    % t_0, T - начальный и конечный моменты счёта t_0 и T
    % M - число интервалов сетки по времени
    % u_0 - начальное условие
    % f и f_u - функции, определяющие неоднородность
    % решаемого ОДУ и её производную по u
    % a_11 - параметр схемы (0 - ERK1, 1 - DIRK1,
    % 1/2 - схема "с полусуммой", (1 + 1i)/2 - CROS1)

    % Выходной параметр:
    % u- массив, содержащий сеточные значения решения ОДУ

    tau = (T - t_0)/M;  % Определение шага сетки
    t = t_0:tau:T;      % Определение сетки

    % Выделение памяти под массив сеточных значений u(t)
```

```
22      u = zeros(1,M + 1);
23
24      u(1) = u_0;             % Задание начального условия
25
26      for m = 1:M
27
28          % Реализация схемы ROS1
29          % (для автономного уравнения)
30          w_1 = (1 - a_11*(t(m + 1) - t(m))*...
31                f_u(u(m)))^(-1)*f(u(m));
32          u(m + 1) = u(m) + (t(m + 1) - t(m))*real(w_1);
33
34      end
35
36  end
```

Замечание. Необходимо обратить внимание на то, что при обращении к компонентам векторов u и t все индексы сдвинуты на $+1$ (по сравнению с аналитическими формулами выше), так как в MatLab'е нумерация элементов массивов начинается с 1 (поэтому $u_0 \equiv u(1)$, $u_1 \equiv u(2)$, $\ldots$, $u_M \equiv u(M+1)$).

Запуск этой функции и последующая отрисовка решения могут быть осуществлены с помощью следующего набора команд:

```
1  % Определение начального и конечного времени счёта
2  t_0 = 0; T = 2;
3
4  M = 50; % Определение числа интервалов сетки
5
6  f = @(u) u^2;           % Определение функции  f(u) = u^2
7  f_u = @(u) 2*u;         % Определение функции  f_u(u) = 2u
8
9  u_0 = 1; % Определение начального условия
10
11 % Определение параметра  a_11  схемы ROS1
12 % (необходимо раскомментировать нужный)
13 a_11 = (1 + 1i)/2;  % CROS1
14 % a_11 = 1;          % DIRK1
15 % a_11 = 1/2;        % схема "с полусуммой"
16 % a_11 = 0;          % ERK1
17
```

```
% Запуск функции ODESolving с целью вычисления решения
u = ODESolving(t_0,T,M,u_0,f,f_u,a_11);

% Отрисовка решения
figure;
% Рисуется график точного решения
plot([0:0.01:2],1./(u_0 - [0:0.01:2]),'--g',...
    'LineWidth',2); hold on;
% Рисуется найденное приближённое решение
plot(t_0:(T - t_0)/M:T,u,'-ok','MarkerSize',3,...
    'LineWidth',1); hold on;
axis([0 2 -30 30]); xlabel('t'); ylabel('u');
```

Результаты работы этой функции по решению тестовой задачи (1.1)–(1.2) по схеме (1.5) для $M = 50$ при различных значениях параметра a_{11} приведены на рис. 1.2.

Как видно, численное решение существует при всех $t \in [t_0, T] \equiv [0, 2]$, поэтому возникает резонный вопрос, как диагностировать факт разрушения решения и определить, какой части численного решения мы можем доверять, а какой нет. Этот вопрос особо актуален при решении гораздо более сложных задач, аналитическое решение которых получить невозможно. Далее мы подробно рассмотрим этот вопрос.

1.2. Численная диагностика разрушения решения

Методика численной диагностики разрушения решения, которую мы рассматриваем в данном учебном пособии, основана на вычислении апостериорной асимптотически точной оценки погрешности (см., например, § 2 главы II в [6]).

Предположим, что мы нашли сеточное решение задачи Коши (1.1) с использованием схемы с порядком точности p на равномерной по времени t сетке T_M с шагом $\tau = (T - t_0)/M$: $T_M = \{t_m,\ 0 \leqslant m \leqslant M :\ t_m = t_0 + m\tau\}$. Это означает, что для всех узлов $t \in T_M$ верно равенство

$$u(t) = u^{(M)}(t) + O(\tau^p). \tag{1.6}$$

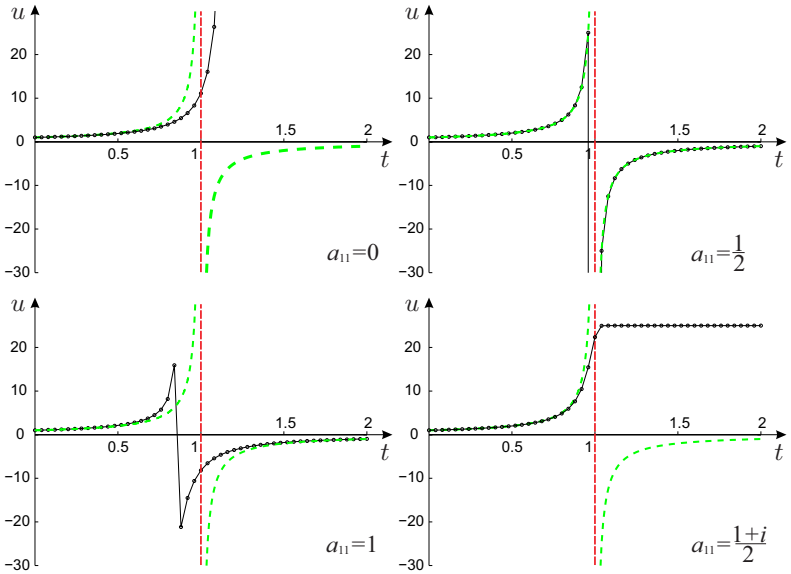

Рис. 1.2: Результат применения схемы ROS1 (1.5) для решения задачи (1.1)–(1.2) при различных значениях параметра a_{11} и одном и том же числе интервалов сетки $M = 50$.

Здесь верхний индекс вида (M) означает, что $u^{(M)}(t)$ найдено численно по соответствующей схеме на сетке с M интервалами.

Теперь перепишем соотношение (1.6) в виде

$$u(t) = u^{(M)}(t) + c(t)\tau^p + O(\tau^{p+1}), \tag{1.7}$$

явно выделяя главный член $R^{(M)}(t) \equiv c(t)\tau^p$ разложения ошибки $u(t) - u^{(M)}(t)$ приближённого вычисления $u(t)$ в ряд Тейлора. При этом будем исходить из предположения о существовании соответствующих непрерывных производных.

Предполагается, что вычисление $u^{(M)}(t)$, $t \in T_M$, мы проводили на фиксированной сетке с M интервалами (через число

которых однозначно определяется шаг сетки τ) по схеме с известным порядком точности p. Таким образом, уравнение (1.7) содержит два неизвестных: $u(t)$ и $c(t)$. Для того чтобы найти два неизвестных, нам необходимо два уравения. Получим второе уравнение, проведя вычисления на сгущённой в r раз сетке (т.е. на сетке, содержащей rM интервалов):

$$u(t) = u^{(rM)}(t) + c(t)\left(\frac{\tau}{r}\right)^p + O\left(\left(\frac{\tau}{r}\right)^{p+1}\right). \tag{1.8}$$

Вычтем из уравнения (1.8) уравнение (1.7) и из полученного равенства выразим $c(t)$:

$$c(t) = \frac{u^{(rM)}(t) - u^{(M)}(t)}{r^p - 1}\frac{r^p}{\tau^p} + O(\tau^1).$$

Таким образом,

$$R^{(rM)}(t) \equiv c(t)\left(\frac{\tau}{r}\right)^p = \frac{u^{(rM)}(t) - u^{(M)}(t)}{r^p - 1} + O(\tau^{p+1}). \tag{1.9}$$

Дробь, стоящая в правой части равенства (1.9), является апостериорной асимптотически точной оценкой члена p-го порядка разложения в ряд Тейлора точного решения $u(t)$, или, что то же самое, апостериорной асимптотически точной оценкой главного члена разложения ошибки $u(t) - u^{(rM)}(t)$ приближённого вычисления $u(t)$ в ряд Тейлора. Таким образом,

$$u(t) = u^{(rM)}(t) + R^{(rM)}(t) + O(\tau^{p+1}).$$

Исходя из того, что при $\tau \to 0$ слагаемое $R^{(rM)}(t)$ превосходит все остальные члены разложения в ряд Тейлора ошибки $u(t) - u^{(rM)}(t)$, его можно воспринимать как апостериорную асимптотически точную оценку погрешности вычисления $u^{(rM)}(t)$. Далее будем под $R^{(rM)}(t)$ подразумевать, опуская $O(\tau^{p+1})$ (но при этом не забывая об асимптотическом характере соответствующей формулы), величину

$$R^{(rM)}(t) = \frac{u^{(rM)}(t) - u^{(M)}(t)}{r^p - 1}, \tag{1.10}$$

что является классической формулой Рунге–Ромберга.

Проведя расчёты ещё на одной сетке — с числом интервалов r^2M, — мы сможем вычислить

$$R^{(r^2M)}(t) \equiv c(t)\Big(\frac{\tau}{r^2}\Big)^p = \frac{u^{(r^2M)}(t) - u^{(rM)}(t)}{r^p - 1} + O(\tau^{p+1}).$$

Заметим, что

$$R^{(rM)}(t) \equiv c(t)\Big(\frac{\tau}{r}\Big)^p, \quad R^{(r^2M)}(t) \equiv c(t)\Big(\frac{\tau}{r^2}\Big)^p, \tag{1.11}$$

и из отношения норм этих двух выражений найдём выражение для эффективного порядка точности

$$p^{eff} = \log_r \frac{\|R^{(rM)}(t)\|}{\|R^{(r^2M)}(t)\|}, \tag{1.12}$$

с которым вычисляется приближённое решение на промежутке времени $t \in [t_0, T]$ и которое обладает свойством: $p^{eff} \to p^{theor} \equiv p$ при $\tau \to 0$. Отметим, что мы здесь говорим именно об *эффективном* порядке точности, потому что он вычисляется исходя из имеющихся приближённых решений в пренебрежении приближённым характером формулы (1.10) (сравните с (1.9)).

Также эффективный порядок точности может быть вычислен и поточечно для каждого отдельного узла сетки $t_m \in T_M$, $0 \leqslant m \leqslant M$. Сооветствующую формулу можно получить из отношения выражений в (1.11):

$$p^{eff}(t) = \log_r \frac{R^{(rM)}(t)}{R^{(r^2M)}(t)}. \tag{1.13}$$

Следует понимать, что аргумент логарифма в (1.13) может оказаться отрицательным, если члены более высокого порядка в разложении ошибки по Тейлору преобладают над членом порядка p. Такое может произойти, например, если коэффициент при p-м члене случайно оказался равным нулю в конкретной задаче

в конкретной точке либо если сетка ещё слишком грубая. Поэтому на практике, чтобы избежать аварийной остановки расчёта, имеет смысл в (1.13) взять модули числителя и знаменателя.

Теперь с учётом сделанного выше замечания опишем практический алгоритм выполнения оценок эффективного порядка точности, что в дальнейшем позволит нам диагностировать факт разрушения решения в случае его наличия.

Для начала введём базовую сетку T_M: $\{t_m\}$, $0 \leqslant m \leqslant M$. После этого произведём последовательное сгущение сетки, начиная с базовой, и вычислим решения

$$u_{(s)}(t) \equiv u^{(r^{s-1}M)}(t)$$

на полученных сетках $T_{r^{s-1}M}$ (s — номер сетки из набора сеток $s = \overline{1,S}$) по схеме с порядком точности p. В этом случае, если r целое, каждая последующая сетка $T_{r^{s-1}M}$ имеет узлы, совпадающие с узлами базовой сетки $t_m \in T_M$, $0 \leqslant m \leqslant M$. В этих узлах t мы можем выполнить апостериорную асимптотически точную оценку погрешности на каждой сетке с номером s по формуле Рунге–Ромберга (1.10)

$$\Delta_{(s)}(t_m) \equiv R^{(r^{s-1}M)}(t_m) = \frac{u_{(s)}(t_m) - u_{(s-1)}(t_m)}{r^p - 1},$$

и оценить эффективный порядок точности (1.12) на всём промежутке времени $t \in [t_0, T]$

$$p_{(s)}^{eff} = \log_r \frac{\sqrt{\sum\limits_{m=0}^{M} \left(u_{(s-1)}(t_m) - u_{(s-2)}(t_m)\right)^2}}{\sqrt{\sum\limits_{m=0}^{M} \left(u_{(s)}(t_m) - u_{(s-1)}(t_m)\right)^2}} \tag{1.14}$$

(здесь выбрана наиболее удобная из возможных норм — евклидова). В случае, если на всём промежутке времени $t \in [t_0, T]$

решение задачи имеет непрерывные p-е производные по времени, имеет место сходимость

$$p_{(s)}^{eff} \xrightarrow[s\to\infty]{} p^{theor} \equiv p.$$

Нарушение этой сходимости говорит о потере гладкости точного решения на промежутке времени $t \in [t_0, T]$. Другими словами, теоретический порядок точности, равный p, характеризует способность соответствующей численной схемы передать ровно p членов (в случае их наличия!) разложения в ряд Тейлора точного решения. А эффективный порядок точности позволяет нам судить о том, сколько реально членов разложения в ряд Тейлора точного решения было передано. В частности, в случае $p^{eff} \leqslant 0$ можно сделать вывод об отсутствии возможности разложить точное решение в ряд Тейлора, что говорит об отсутствии решения (или, другими словами, о факте разрушения решения в какой-либо точке промежутка $t \in [t_0, T]$) или потере его гладкости.

С целью локализации конкретного момента времени, в который произошло разрушение решения, можно оценить эффективный порядок точности и поточечно (1.13) в каждом узле $t_m \in T_M$, $0 \leqslant m \leqslant M$,

$$p_{(s)}^{eff}(t_m) = \log_r \frac{|u_{(s-1)}(t_m) - u_{(s-2)}(t_m)|}{|u_{(s)}(t_m) - u_{(s-1)}(t_m)|}. \qquad (1.15)$$

В точках t, в которых решение исходной задачи имеет непрерывные p-е производные по времени, имеет место сходимость

$$p_{(s)}^{eff}(t) \xrightarrow[s\to\infty]{} p^{theor} \equiv p$$

и соответствующая оценка погрешности является асимптотически точной при $s \to \infty$ (или, что то же самое, $N, M \to \infty$). Нарушение этой сходимости говорит о потере гладкости точного решения. В частности, в случае степенной «сингулярности» $u(t) \sim (T_{bl} - t)^{-\beta}$, где T_{bl} — время разрушения (индекс «bl» —

сокращение от «blow-up», «разрушение»), для любого $t > T_{bl}$ эффективный порядок точности $p_{(s)}^{eff}(t) \xrightarrow[s\to\infty]{} -\beta$. Это позволяет найти соответствующую степень β. Если $p_{(s)}^{eff}(t) \xrightarrow[s\to\infty]{} -\infty$ для любого $t > T_{bl}$, мы можем утверждать, что решение экспоненциально возрастает, т.е. $u(t) = \infty$; если $p_{(s)}^{eff}(t) \xrightarrow[s\to\infty]{} 0$ для любого $t > T_{bl}$, то рост решения в окрестности «сингулярности» является логарифмическим: $u(t) \sim \ln(T_{bl} - t)$. Вывод соответствующих утверждений можно найти в [2, 9]. Момент разрушения решения T_{bl} может быть найден с точностью до величины интервала базовой сетки по времени T_M.

Далее приводится пример видоизменённой MatLab-функции ODESolving, которая реализует поиск приближённого численного решения $u_{(s)}(t) \equiv u^{(r^{s-1}M)}(t)$ задачи (1.4) по схеме CROS1 на сетке с номером s, включая выбор сеточных значений только из узлов, совпадающих с узлами базовой сетки T_M.

```
function u_basic = ODESolving(t_0,T,M_0,u_0,f,f_u,s,r)

    % Функция находит приближённое численное решение
    % обыкновенного дифференциального уравнения (ОДУ/ODE)

    % Входные параметры:
    % t_0, T - начальный и конечный моменты счёта  t_0  и  T
    % M_0 - число интервалов базовой сетки по времени
    % u_0 - начальное условие
    % f и f_u - функции, определяющие неоднородность
    % решаемого ОДУ и её производную по u
    % s - номер сетки, на которой вычисляется решение
    % (если s = 1, то решение ищется на базовой сетке)
    % r - коэффициент сгущения сетки

    % Выходной параметр:
    % u_basic - массив, содержащий сеточные значения
    % решения ОДУ только в узлах,
    % совпадающих с узлами базовой сетки

    % Формирование сгущённой в r^(s - 1) раз
    % сетки с номером s
```

```
23
24      M = M_0*r^(s - 1);  % Вычисление числа интервалов
25                          % на сетке с номером s
26      tau = (T - t_0)/M;  % Определение шага сгущённой сетки
27      t = t_0:tau:T;      % Определение сгущённой сетки
28
29      % Выделение памяти под массив u_basic, в котором
30      % будут храниться сеточные значения из узлов,
31      % совпадающих с узлами базовой сетки
32      u_basic = zeros(1,M_0 + 1);
33
34      % Выделение памяти под массив сеточных значений
35      % решения на сгущённой сетке
36      u = zeros(1,M + 1);
37
38      u(1) = u_0;         % Задание начального условия
39
40      for m = 1:M
41
42          % Реализация схемы CROS1
43          % (для автономного уравнения)
44          w_1 = (1 - (1 + 1i)/2*(t(m + 1) - t(m))* ...
45              f_u(u(m)))^(-1)*f(u(m));
46          u(m + 1) = u(m) + (t(m + 1) - t(m))*real(w_1);
47
48      end
49
50      % Из массива u выбираются сеточные значения из узлов,
51      % совпадающих с узлами базовой сетки
52      for m = 1:(M_0 + 1)
53          u_basic(1,m) = u((m - 1)*r^(s - 1) + 1);
54      end
55
56  end
```

Замечание 1. Напомним ещё раз, что здесь и далее при обращении к компонентам векторов u и t все индексы сдвинуты на $+1$, так как в MatLab'е нумерация элементов массивов начинается с 1 (поэтому $u_0 \equiv u(1)$, $u_1 \equiv u(2)$, $\ldots$, $u_M \equiv u(M+1)$ и $t_0 \equiv t(1)$, $t_1 \equiv t(2)$, $\ldots$, $t_M \equiv t(M+1)$).

Замечание 2. Необходимо отметить, что схема CROS1 ($a_{11} = \frac{1+i}{2}$) позволяет избежать численного переполнения решения в

случае его стремления к бесконечности и даёт возможность провести численную диагностику характера разрушения [2, 9]. Поэтому данная MatLab-функция и все нижеследующие примеры написаны только для случая схемы CROS1.

Запуск этой функции для получения набора сеточных решений $u_{(s)}(t) \equiv u^{(r^{s-1}M)}(t)$, $s = \overline{1,S}$, задачи (1.1)–(1.2) на различных сетках, начиная с сетки T_M с $M = 50$, может быть осуществлён с помощью следующего набора команд.

```
% Определение начального и конечного времени счёта
t_0 = 0; T = 2;

M = 50; % Определение числа интервалов базовой сетки

f = @(u) u^2;          % Определение функции  f(u) = u^2
f_u = @(u) 2*u;        % Определение функции  f_u(u) = 2u

u_0 = 1; % Определение начального условия

S = 10; % Число сеток, на которых ищется
        % приближённое решение
r = 2;  % Коэффициент сгущения сеток

% Выделение памяти под массивы сеточных значений
% решений ОДУ на разных сетках с номерами  s = 1,S
% Первый индекс - номер сетки s из последовательности
% сгущающихся сеток, на которых ищется решение
% Второй индекс определяет массив (для фиксированного s),
% в котором хранятся сеточные значения решения из узлов,
% совпадающих с узлами базовой сетки
array_of_u = zeros(S,M + 1);

% "Большой цикл", который пересчитывает решение S раз
% на последовательности сгущающихся сеток
% Массив сеточных значений решения содержит только
% сеточные значения из узлов,
% совпадающих с узлами базовой сетки
for s = 1:S
    u = ODESolving(t_0,T,M,u_0,f,f_u,s,r);
    array_of_u(s,:) = u;
```

```
    s
end

% Сохраняем необходимые для дальнейшей диагностики
% разрушения решения данные Workspace'а в файл
save('data.mat','array_of_u','M','r','S','t_0','T');
```

В связи с тем, что время расчётов на каждой очередной сетке возрастает в r раз, нахождение приближённых решений на разных сетках реализовано в виде вышеприведённого отдельного кода, который вычисляет все необходимые для дальнейшей численной диагностики данные и записывает их в файл data.mat. Его содержимое будет подгружаться вводимыми далее в ходе изложения функциями без повторного расчёта решений на последовательности сгущающихся сеток.

Далее приводится пример MatLab-кода, который вычисляет эффективные порядки точности приближённого решения на промежутке времени $t \in [t_0, T]$ (1.14), используя данные из уже сформированного предыдущей программой файла data.mat.

```
% Загрузка результата вычисления приближённого решения
% на последовательности из S сгущающихся в r раз сеток
load('data.mat');

% Выделение памяти под массив значений эффективных
% порядков точности расчёта приближённого решения
% на разных сетках
p_eff = zeros(S,1);

% Вычисление эффективных порядков точности
p_eff(1) = NaN; % Вычисление p_(1)^eff невозможно
p_eff(2) = NaN; % Вычисление p_(2)^eff невозможно
for s = 3:S
    p_eff(s) = log(...
        sqrt(sum((array_of_u(s - 1,:) - ...
        array_of_u(s - 2,:)).^2))/...
        sqrt(sum((array_of_u(s,:) - ...
        array_of_u(s - 1,:)).^2)))/...
        log(r);
```

```
end

% Вывод последовательности значений  p_(s)^eff
for s = 1:S
    X = ['p^eff_(',int2str(s),')=', ...
         num2str(p_eff(s),'%6.4f')];
    disp(X);
end
```

Отметим, что вычисление $p^{eff}_{(1)}$ и $p^{eff}_{(1)}$ невозможно потому, что для нахождения эффективных порядков точности необходимо знать приближённое решение на двух предыдущих сетках (см. (1.14)).

В таблицах 1.1 и 1.2 приведены примеры результатов вычислений последовательностей значений $p^{eff}_{(s)}$ для различных временных интервалов $t \in [t_0, T]$. Хорошо видно, что для тех временных промежутков, для которых решение задачи Коши (1.1) существует, последовательность эффективных порядков точности $p^{eff}_{(s)}$ сходится к теоретическому порядку точности $p^{theor} \equiv p$. В случае же отсутствия решения на соответствующем временном промежутке последовательность эффективных порядков точности быстро сходится к неположительному числу.

Далее приводится пример MatLab-кода, который вычисляет эффективные порядки точности (1.15) приближённого решения в узлах, совпадающих с узлами базовой сетки T_M, используя данные из уже сформированного файла data.mat, с целью определения конкретного момента времени, в который происходит разрушение решения.

```
% Загрузка результата вычисления приближённого решения
% на последовательности из S сгущающихся в r раз сеток
load('data.mat');

% Выделение памяти под массив значений эффективных
% порядков точности расчёта приближённого решения
% в каждом узле  t_m,  1 ⩽ m ⩽ M  (второй индекс массива),
% кроме  t_0, так как в нём решение задано точно,
```

s	$T = 0.50$	$T = 0.90$	$T = 0.95$	$T = 0.99$	$T = 2.00$
3	1.9994	1.9356	1.7304	0.1521	-0.9986
4	1.9998	1.9830	1.9203	0.9111	-0.9999
5	2.0000	1.9957	1.9789	1.5437	-1.0000
6	2.0000	1.9989	1.9946	1.8555	-1.0000
7	2.0000	1.9997	1.9986	1.9606	-1.0000
8	2.0000	1.9999	1.9997	1.9899	-1.0000
9	2.0000	2.0000	1.9999	1.9974	-1.0000
10	2.0000	2.0000	2.0000	1.9994	-1.0000
11	2.0000	2.0000	2.0000	1.9998	-1.0000
12	1.9999	2.0000	2.0000	2.0000	-1.0000

Таблица 1.1: Наборы значений $p_{(s)}^{eff}$ по итогам расчёта задачи (1.1) с начальным условием $u_0 = 1$ для разных временных промежутков $t \in [t_0, T] \equiv [0, T]$ с начальными сетками T_M с одинаковым числом интервалов $M = 50$ и коэффициентом сгущения сеток $r = 2$.

```
% и на разных сетках (первый индекс массива)
p_eff_ForEveryTime = zeros(S,M + 1);

% Вычисление эффективных порядков точности
for m = 2:(M + 1)

    % Вычисление p_(1)^eff(t_m) и p_(2)^eff(t_m) невозможно
    p_eff_ForEveryTime(1,m) = NaN;
    p_eff_ForEveryTime(2,m) = NaN;

    for s = 3:S
        p_eff_ForEveryTime(s,1) = inf;
        p_eff_ForEveryTime(s,m) = log(...
            abs(array_of_u(s-1,m)-array_of_u(s-2,m))/...
            abs(array_of_u(s,m)-array_of_u(s-1,m)))/...
            log(r);
    end
end
```

s	$T = 0.50$	$T = 0.90$	$T = 0.95$	$T = 0.99$	$T = 2.00$
3	1.9994	1.9803	1.9189	0.9294	-0.9996
4	1.9998	1.9950	1.9784	1.5443	-1.0000
5	2.0000	1.9987	1.9945	1.8547	-1.0000
6	2.0000	1.9997	1.9986	1.9603	-1.0000
7	2.0000	1.9999	1.9997	1.9898	-1.0000
8	2.0000	2.0000	1.9999	1.9974	-1.0000
9	2.0000	2.0000	2.0000	1.9994	-1.0000
10	2.0000	2.0000	2.0000	1.9998	-1.0000
11	2.0000	2.0000	2.0000	2.0000	-1.0000
12	1.9999	2.0000	2.0000	2.0000	-1.0000

Таблица 1.2: Наборы значений $p_{(s)}^{eff}$ по итогам расчёта задачи (1.1) с начальным условием $u_0 = 1$ для разных временных промежутков $t \in [t_0, T] \equiv [0, T]$ с сетками T_M с одинаковым начальным шагом $\tau = 0.01$ ($M = T/0.01$) и коэффициентом сгущения сеток $r = 2$.

```
27
28 % Отрисовка результатов расчётов для сетки с номером s
29 % S = 8;
30 figure;
31 t = t_0:(T - t_0)/M:T; % Определение базовой сетки
32 % Рисуется зависимость теоретического порядка точности
33 % от узла базовой сетки
34 plot(t,t*0 + 2,'-*k','MarkerSize',3); hold on;
35 % Рисуется зависимость эффективного порядка точности
36 % от узла базовой сетки
37 plot(t(2:M+1),p_eff_ForEveryTime(S,2:M + 1),...
38     '-sk','MarkerSize',5,'LineWidth',1);
39 axis([t(1) t(M + 1) -2.0 3.0]);
40 xlabel('t'); ylabel('p^{eff}');
```

На рис. 1.3 представлен результат работы данного программного кода для решения задачи (1.1)–(1.2). В качестве базовой сетки T_M была взята сетка с $M = 50$ интервалами, и было вы-

полнено последовательное сгущение сеток с коэффициентом сгущения $r = 2$. Представлен результат расчётов после получения решения на 8-й сетке ($s = 8$), для которой получен наглядный выход поточечных значений эффективных порядков точности $p^{eff}_{(s)}(t_m)$, $0 \leqslant m \leqslant M$, на асимптотически точные значения. На рис. 1.4 для сравнения показаны графики функций $p^{eff}_{(s)}(t)$ для различных s, которые экспериментальным образом обосновывают выход на асимптотику значений $p^{eff}_{(8)}(t_m)$, представленных на рис. 1.3.

Считая, что мы не знаем точного решения задачи (1.1)–(1.2), можно рассуждать так. После вычисления на $S = 8$ вложенных сетках поточечные значения эффективного порядка точности $p^{eff}_{(s)}(t)$ сходятся к $p^{theor} \equiv 2$ ($p = 2$ для используемой схемы CROS1) для каждого момента времени t_m с $m \leqslant 24$. Для значений t_m, $m \geqslant 25$, имеем: $p^{eff}_{(s)}(t_{25}) \to -1$. Значит, при $t = t_{25}$ точное решение уже не существует, а при $t = t_{24}$ оно, скорее всего, ещё существует. Таким образом, разрушение решения происходит в момент $T_{bl} \in (t_{24}, t_{25}] \equiv (0.960, 1.000]$. Кроме того, в силу сказанного выше о связи характера разрушения и эффективного порядка точности мы можем сделать вывод, что в точке T_{bl} решение имеет особенность типа полюса $u(t) \sim (T_{bl} - t)^{-1}$.

Замечание 3. К сожалению, численные методы редко могут дать полную гарантию результата в том смысле, в котором её даёт доказательство в классической математике. Даже нарушение сходимости p^{eff} к теоретическому порядку точности, строго говоря, нельзя наблюдать за конечное число шагов. Мы лишь исходим из того, что ситуация, когда после столь наглядного «притяжения» значений p^{eff} к числу -1 тенденция изменится, представляется крайне маловероятной на практике. А поскольку теория утверждает (на сей раз уже вполне строго), что при наличии точного классического решения разностная схема к нему сходится, то из отсутствия такой сходимости мы можем сделать вывод об отсутствии классического решения. Более того, как показано в [2], предельное при $s \to +\infty$ значение p^{eff} после момен-

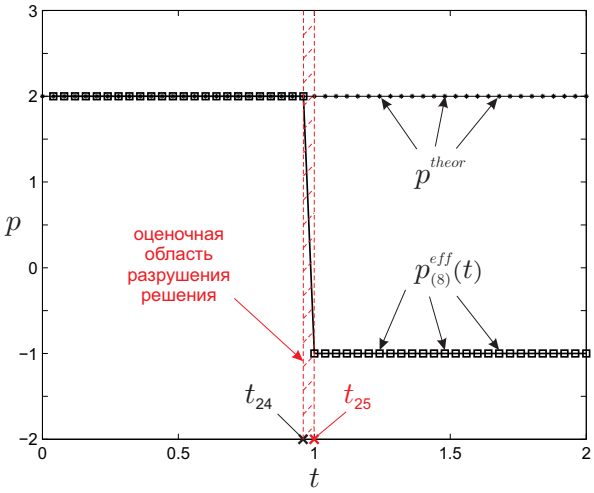

Рис. 1.3: Результат вычислений эффективных порядков точности $p^{eff}_{(S)}(t_m)$, $0 \leqslant m \leqslant M$, задачи (1.1)–(1.2) для следующего набора параметров: $M = 50$, $r = 2$, $S = 8$.

та разрушения позволяет делать вывод о характере особенности. С другой стороны, наглядное «притяжение» эффективного порядка точности к теоретическому позволяет надеяться как на существование классического решения в области такого притяжения, так и на сходимость к нему численного решения.

В итоге мы можем сделать вывод о том, какой части численного решения (см. рис. 1.5), полученного на 8-й вложенной сетке, мы можем доверять, а какой нет.

Решение на 8-й сетке ($s = 8$) было получено посредством набора следующих команд.

```
% Загрузка результата вычисления приближённого решения
% на последовательности из S сгущающихся в r раз сеток
load('data.mat');
```

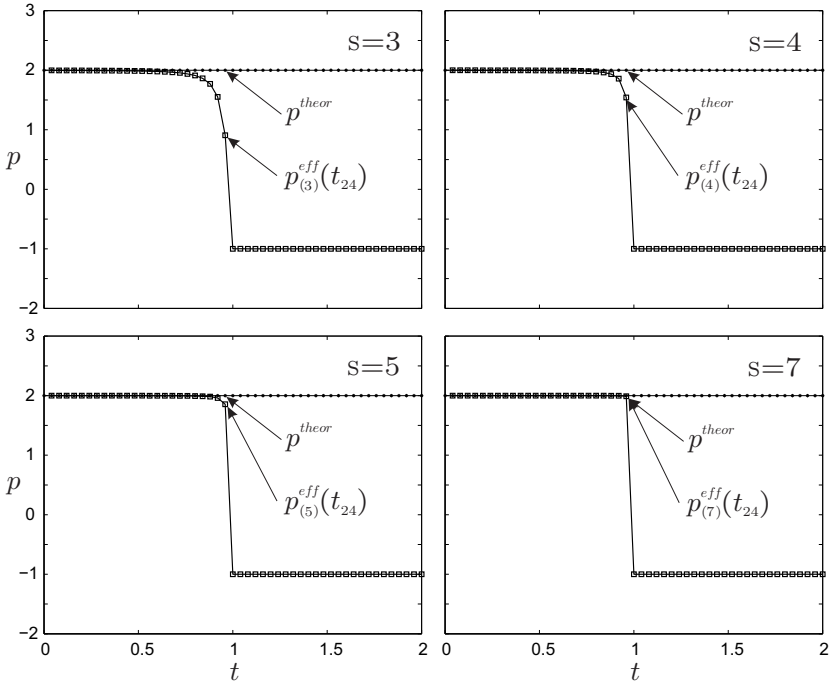

Рис. 1.4: Результат вычислений эффективных порядков точности $p^{eff}_{(s)}(t_m)$, $0 \leqslant m \leqslant M$, задачи (1.1)–(1.2) для $M = 50$, $r = 2$ и различных s: $s = \{3, 4, 5, 7\}$.

```
4
5  % Из массива решений на разных сетках выбирается
6  % сеточное решение на 8-й сетке
7  s = 8;
8
9  u = array_of_u(s,:);
10
11 % Отрисовка решения
12 figure;
```

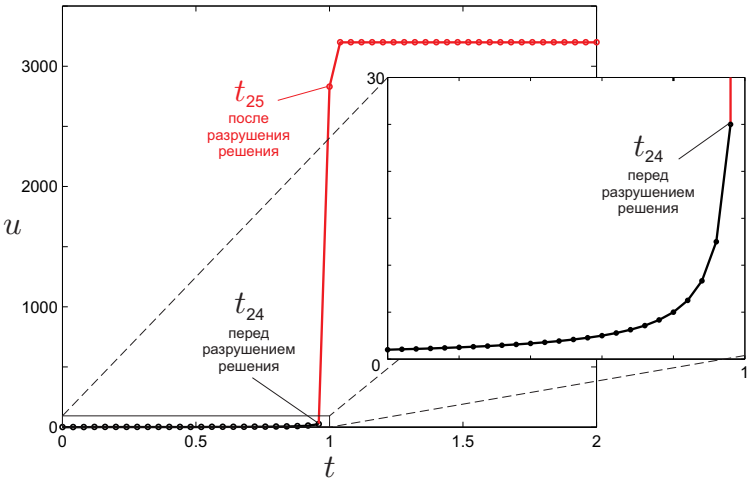

Рис. 1.5: Результат расчёта $u_{(s)}(t) \equiv u^{(r^{s-1}M)}(t)$ задачи (1.1)–(1.2) для следующего набора параметров: $M = 50$, $r = 2$, $s = 8$. Отмечены только узлы, совпадающие с узлами базовой сетки.

```
t = t_0:(T - t_0)/M:T; % Определение базовой сетки по t
plot(t,u,'-ok','MarkerSize',3,'LineWidth',1);
axis([0 2 -0 30]); xlabel('t'); ylabel('u');
```

1.3. Практические задания

Задача 1. Решите аналитически и численно (по схеме CROS1) задачу Коши

$$\begin{cases} \dfrac{du}{dt} = 1 + u^2, \quad t \in (t_0, T], \\ u(t_0) = u_0 \end{cases} \tag{1.16}$$

для следующего набора параметров:

$$t_0 = 0, \quad T = 1, \quad u_0 = 1.$$

Сравните найденное аналитически и оценённое численно время разрушения решения. Какое значение эффективного порядка точности наблюдается после разрушения? (Воспользуйтесь формулой (1.15) на с. 17.) Проведите соответствующую классификацию особенности.

Задача 2. Решите аналитически и численно (по схеме CROS1) задачу Коши

$$\begin{cases} \dfrac{du}{dt} = e^u, \quad t \in (t_0, T], \\ u(t_0) = u_0 \end{cases} \tag{1.17}$$

для следующего набора параметров:

$$t_0 = 0, \quad T = 2, \quad u_0 = 0.$$

Сравните найденное аналитически и оценённое численно время разрушения решения. Какое значение эффективного порядка точности наблюдается после разрушения? (Воспользуйтесь формулой (1.15) на с. 17.) Проведите соответствующую классификацию особенности.

Задача 3. Решите численно (по схеме CROS1) задачу Коши

$$\begin{cases} \dfrac{du}{dt} = e^{u^2}, \quad t \in (t_0, T], \\ u(t_0) = u_0 \end{cases} \tag{1.18}$$

для следующего набора параметров:

$$t_0 = 0, \quad T = 5, \quad u_0 = 0.$$

Наблюдается ли на данном отрезке разрушение решения? Если да, какое значение эффективного порядка точности наблюдается после разрушения? (Воспользуйтесь формулой (1.15) на с. 17.) При наличии разрушения проведите классификацию особенности.

Замечание. Если численная оценка времени разрушения значительно меньше времени счёта (скажем, $T_{bl}/T < 1/2$), то разумно повторить расчёты на промежутке меньшей длины, но взять при этом то же количество узлов базовой сетки.

2 Диагностика разрушения решения начально-краевой задачи для уравнения в частных производных псевдопараболического типа

В данной главе рассматриваются особенности численной диагностики разрушения решения начально-краевой задачи для уравнений в частных производных псевдопараболического типа. В качестве примера будем рассматривать уравнение Бенджамена–Бона–Махони–Бюргерса [7], которое возникает в теории длинных волн на воде. Требуется найти функцию $u(x,t)$, определённую в области $(x,t) \in [a,b] \times [t_0,T]$[1] и удовлетворяющую системе уравнений

$$\begin{cases} \dfrac{\partial}{\partial t}\big(u_{xx} - u\big) + u_{xx} + uu_x = 0, \quad x \in (a,b], \quad t \in (t_0,T], \\ u(a,t) = 0, \quad u_x(a,t) = 0, \quad t \in (t_0,T], \\ u(x,t_0) = u_{init}(x), \quad x \in [a,b], \end{cases} \tag{2.1}$$

[1] *Замечание.* Мы ставим задачу численного нахождения решения до момента времени T включительно, хотя знаем, что решение в этот момент времени может не существовать и, более того, даже может разрушиться ранее. Это связано с тем, что мы хотим диагностировать разрушение решения численно, а значит, нам необходимо найти *численное* решение вплоть до этого момента времени включительно.

а также диагностировать факт разрушения решения (в случае его наличия) и уточнить его локализацию как во времени, так и в пространстве, по сравнению с априорной оценкой (если таковая имеется), полученной аналитическими методами.

2.1. Поиск численного решения

Для численного решения задачи (2.1) мы применим метод прямых (MOL) [1, 3, 10]. Суть этого метода состоит в следующем. Сначала вводится сетка по пространственной переменной и аппроксимируются пространственные производные. В итоге исходная дифференциальная задача с уравнением в частных производных заменяется системой большого числа обыкновенных дифференциальных уравнений (и, возможно, некоторого количества алгебраических, аппроксимирующих граничные условия). Затем в полученной системе ОДУ аппроксимируются производные по времени, что эффективнее всего делать с помощью одностадийной схемы Розенброка с комплексным коэффициентом CROS1 [1, 11].

Итак, вначале введём равномерную сетку X_N с N интервалами только по пространственной переменной x с шагом $h = (b-a)/N$ (что соответствует $N+1$ узлу сетки): $X_N = \{x_n,\ 0 \leqslant n \leqslant N :\ x_n = a + nh\}$. Таким образом, после конечно-разностной аппроксимации пространственных производных со вторым порядком точности мы получим дифференциально-алгебраическую систему, из которой требуется определить $N+1$ неизвестную функцию $u_n \equiv u_n(t) \equiv u(x_n, t)$, $n = \overline{0, N}$:

$$
\begin{cases}
\dfrac{d}{dt}\Big(\dfrac{u_{n+1} - 2u_n + u_{n-1}}{h^2} - u_n\Big) + \dfrac{u_{n+1} - 2u_n + u_{n-1}}{h^2} + \\
\qquad\qquad + u_n \dfrac{u_{n+1} - u_{n-1}}{2h} = 0, \quad n = \overline{1, N-1}, \quad t \in (t_0, T], \\
u_0 = 0, \quad \dfrac{-\frac{3}{2}u_0 + 2u_1 - \frac{1}{2}u_2}{h} = 0, \quad t \in (t_0, T], \\
u_n(t_0) = u_{init}(x_n), \quad n = \overline{0, N}.
\end{cases}
$$

Для удобства последующих преобразований перепишем эту систему в следующем виде, уединяя дифференциальную часть в левой части каждого уравнения:

$$\begin{cases} \dfrac{du_{n-1}}{dt} - \big(2+h^2\big)\dfrac{du_n}{dt} + \dfrac{du_{n+1}}{dt} = -\big(u_{n+1} - 2u_n + u_{n-1}\big) - \\ \qquad\qquad - \dfrac{h}{2}u_n\big(u_{n+1} - u_{n-1}\big),\ n = \overline{1, N-1}, \quad t \in (t_0, T], \\ u_0 = 0, \quad u_1 = \dfrac{3}{4}u_0 + \dfrac{1}{4}u_2, \quad t \in (t_0, T], \\ u_n(t_0) = u_{init}(x_n), \quad n = \overline{0, N}. \end{cases}$$

Полученная система является дифференциально-алгебраической, поскольку включает в себя как дифференциальные уравнения, так и алгебраические (два уравнения, определяемые граничными условиями). Путём подстановки выражений для u_0 и u_1 ($u_0 = 0$, $u_1 = \frac{1}{4}u_2$) в первые два дифференциальные уравнения (для $n = \{1, 2\}$) система может быть сведена к чисто дифференциальной:

$$\begin{cases} \dfrac{1}{4}\big(2-h^2\big)\dfrac{du_2}{dt} = -\dfrac{1}{2}u_2 - \dfrac{h}{8}u_2^2, \quad t \in (t_0, T], \\ -\Big(\dfrac{7}{4}+h^2\Big)\dfrac{du_2}{dt} + \dfrac{du_3}{dt} = -\Big(u_3 - \dfrac{7}{4}u_2\Big) - \\ \qquad\qquad - \dfrac{h}{2}u_2\Big(u_3 - \dfrac{1}{4}u_2\Big), \quad t \in (t_0, T], \\ \dfrac{du_{n-1}}{dt} - \big(2+h^2\big)\dfrac{du_n}{dt} + \dfrac{du_{n+1}}{dt} = -\big(u_{n+1} - 2u_n + u_{n-1}\big) - \\ \qquad\qquad - \dfrac{h}{2}u_n\big(u_{n+1} - u_{n-1}\big),\ n = \overline{3, N-1}, \quad t \in (t_0, T], \\ u_n(t_0) = u_{init}(x_n), \quad n = \overline{0, N}. \end{cases}$$

Эта система, содержащая $N-1$ уравнение и $N-1$ неизвестную

функцию u_n, $n = \overline{2, N}$, может быть переписана в векторном виде:

$$\begin{cases} \boldsymbol{D}\dfrac{d\boldsymbol{y}}{dt} = \boldsymbol{f}(\boldsymbol{y}), \quad t \in (t_0, T], \\ \boldsymbol{y}(t_0) = \boldsymbol{y}_{init}, \end{cases} \tag{2.2}$$

где $\boldsymbol{y} = \begin{pmatrix} u_2 & u_3 & \dots & u_N \end{pmatrix}^T$, $\boldsymbol{f} = \begin{pmatrix} f_1 & f_2 & \dots & f_{N-1} \end{pmatrix}^T$ и $\boldsymbol{y}_{init} = \begin{pmatrix} u_2(t_0) & u_3(t_0) & \dots & u_N(t_0) \end{pmatrix}^T$.

Здесь вектор-функция $\boldsymbol{f}$ имеет следующую структуру:

$$f_n = \begin{cases} -\frac{1}{2}y_1 - \dfrac{h}{8}y_1^2, & \text{если } n = 1, \\ -\Big(y_2 - \frac{7}{4}y_1\Big) - \dfrac{h}{2}y_1\Big(y_2 - \frac{1}{4}y_1\Big), & \text{если } n = 2, \\ -\big(y_n - 2y_{n-1} + y_{n-2}\big) - \\ \quad -\dfrac{h}{2}y_{n-1}\big(y_n - y_{n-2}\big), & \text{если } n = \overline{3, N-1}. \end{cases}$$

Ниже приводится пример MatLab-функции, которая реализует вычисление компонент вектор-функции $\boldsymbol{f}$.

```
function f = f(y,h,N)

    % Функция вычисляет вектор правой части
    % решаемой системы ОДУ

    % Входные данные:
    % y - вектор решения системы ОДУ
    % на текущем временном слое
    % h - шаг сетки по переменной x
    % N - число интервалов сетки по переменной x

    % Выходные данные:
    % f - искомый вектор f

    f = zeros(N - 1,1);

    f(1) = -1/2*y(1) - h/8*y(1)^2;
    f(2) = -(y(2) - 7/4*y(1)) - ...
```

```
        h/2*y(1)*(y(2) - 1/4*y(1));
    for n = 3:(N - 1)
        f(n) = -(y(n) - 2*y(n - 1) + y(n - 2)) + ...
            -h/2*y(n - 1)*(y(n) - y(n - 2));
    end

end
```

А матрица $\boldsymbol{D}$ имеет следующие ненулевые элементы:

$$
\begin{aligned}
D_{n,n-2} &= \Big\{1, \quad \text{если } n = \overline{3, N-1},\\
D_{n,n-1} &= \begin{cases} -\left(\frac{7}{4} + h^2\right), & \text{если } n = 2,\\ -\left(2 + h^2\right), & \text{если } n = \overline{3, N-1}, \end{cases}\\
D_{n,n} &= \begin{cases} \frac{1}{4}\left(2 - h^2\right), & \text{если } n = 1,\\ 1, & \text{если } n = \overline{2, N-1}. \end{cases}
\end{aligned}
$$

Ниже приводится пример MatLab-функции, которая реализует вычисление компонент матрицы $\boldsymbol{D}$.

```
function D = D(h,N)

    % Функция вычисляет матрицу дифференциального
    % оператора решаемой системы ОДУ

    % Входные данные:
    % h - шаг сетки по переменной x
    % N - число интервалов сетки по переменной x

    % Выходные данные:
    % D - искомая матрица дифференциального оператора

    D = zeros(N - 1,N - 1);

    D(1,1) = 1/4*(2 - h^2);
    D(2,1) = -(7/4 + h^2);
    D(2,2) = 1;
    for n = 3:(N - 1)
        D(n,n - 2) = 1;
```

```
20          D(n,n - 1) = -(2 + h^2);
21          D(n,n) = 1;
22      end
23
24  end
```

Теперь введём равномерную сетку T_M по времени t с шагом $\tau = (T - t_0)/M$, которая имеет M интервалов (то есть $M + 1$ узел): $T_M = \{t_m,\ 0 \leqslant m \leqslant M :\ t_m = t_0 + m\tau\}$.

В результате мы можем применить схему Розенброка CROS1 для решения системы (2.2):

$$\begin{aligned} &\boldsymbol{y}(t_{m+1}) = \boldsymbol{y}(t_m) + (t_{m+1} - t_m)\,\mathrm{Re}\,\boldsymbol{w}_1\,, \\ &\text{где } \boldsymbol{w}_1 \text{ является решением СЛАУ} \\ &\left[\boldsymbol{D} - \frac{1+i}{2}(t_{m+1} - t_m)\,\boldsymbol{f_y}\Big(\boldsymbol{y}(t_m)\Big)\right]\boldsymbol{w}_1 = \boldsymbol{f}\Big(\boldsymbol{y}(t_m)\Big). \end{aligned} \tag{2.3}$$

Здесь $\boldsymbol{f_y}$ — матрица с элементами $(f_y)_{n,m} \equiv \frac{\partial f_n}{\partial y_m}$ (матрица Якоби), которая для рассматриваемой системы имеет следующие ненулевые элементы:

$$(f_y)_{n,n-2} = -1 + \frac{h}{2}y_{n-1}, \quad \text{если } n = \overline{3, N-1},$$

$$(f_y)_{n,n-1} = \begin{cases} \frac{7}{4} - \frac{h}{2}(y_2 - \frac{1}{2}y_1), & \text{если } n = 2, \\ 2 - \frac{h}{2}(y_n - y_{n-2}), & \text{если } n = \overline{3, N-1}, \end{cases}$$

$$(f_y)_{n,n} = \begin{cases} -\frac{1}{2} - \frac{h}{4}y_1, & \text{если } n = 1, \\ -1 - \frac{h}{2}y_{n-1}, & \text{если } n = \overline{2, N-1}. \end{cases}$$

Ниже приводится пример MatLab-функции, которая реализует вычисление компонент матрицы Якоби $\boldsymbol{f_y}$.

```
1  function f_y = f_y(y,h,N)
```

```
2
3       % Функция вычисляет матрицу Якоби правой части
4       % решаемой системы ОДУ
5
6       % Входные данные:
7       % y - вектор решения системы ОДУ
8       % на текущем временном слое
9       % h - шаг сетки по переменной x
10      % N - число интервалов сетки по переменной x
11
12      % Выходные данные:
13      % f_y - искомая матрица Якоби
14
15      f_y = zeros(N - 1,N - 1);
16
17      f_y(1,1) = -1/2 - h/4*y(1);
18      f_y(2,1) = 7/4 - h/2*(y(2) - 1/2*y(1));
19      f_y(2,2) = -1 - h/2*y(1);
20      for n = 3:(N - 1)
21          f_y(n,n - 2) = -1 + h/2*y(n - 1);
22          f_y(n,n - 1) = 2 - h/2*(y(n) - y(n - 2));
23          f_y(n,n) = -1 - h/2*y(n - 1);
24      end
25
26  end
```

Далее приводится пример MatLab-функции, которая реализует поиск численного решения задачи (2.1) в видоизменённой форме (2.2) по схеме (2.3), используя приведённые выше функции f, D и f_y.

```
1   function u = PDESolving(a,b,N_0,t_0,T,M_0, ...
2       u_init,s,r_x,r_t)
3
4       % Функция находит приближённое численное решение
5       % уравнения в частных производных (УрЧП/PDE)
6
7       % Входные параметры:
8       % a,b - границы области [a,b] по переменной x
9       % N_0 - число интервалов базовой сетки по пространству
10      % t_0, T - начальный и конечный моменты счёта t_0 и T
11      % M_0 - число интервалов базовой сетки по времени
```

```
12      % u_init - функция, определяющяя начальное условие
13      % s - номер сетки, на которой вычисляется решение
14      % (если s = 1, то решение ищется на базовой сетке)
15      % r_x и r_t - коэффициенты сгущения сетки по x и по t
16
17      % Выходной параметр:
18      % u- массив, содержащий сеточные значения решения УрЧП
19      % только в узлах, совпадающих с узлами базовой сетки
20
21      % Формирование сгущённой
22      % в r_x^(s - 1) раз по пространственной переменной x и
23      % в r_t^(s - 1) раз по временной переменной t
24      % сетки с номером s
25
26      N = N_0*r_x^(s - 1);     % Вычисление числа интервалов
27      M = M_0*r_t^(s - 1);     % на сетке с номером s
28
29      h = (b - a)/N;       % Определение шага сетки по x
30      x = a:h:b;           % Определение сгущённой сетки по x
31      tau = (T - t_0)/M; % Определение шага сетки по t
32      t = t_0:tau:T;       % Определение сгущённой сетки по t
33
34      % Выделение памяти под массив u
35      % В строке с номером (m + 1) этого массива хранятся
36      % сеточные значения решения, соответствующие
37      % моменту времени t_m базовой сетки по времени
38      u = zeros(M_0 + 1,N_0 + 1);
39
40      % Выделение памяти под массив сеточных значений
41      % решения системы ОДУ, соответствующего
42      % текущему моменту времени t_m
43      y = zeros(1,N - 1);
44
45      % Задание начального условия решаемой системы ОДУ
46      for n = 1:(N - 1)
47          y(1,n) = u_init(x(n + 2));
48      end
49
50      % Из первой строки массива u_init,
51      % соответствующего начальному условию,
52      % выбираются сеточные значения из узлов,
53      % совпадающих с узлами базовой сетки по пространству
54      for n = 1:(N_0 + 1)
```

```
55          u(1,n) = u_init(x((n - 1)*r_x^(s - 1) + 1));
56      end
57
58      % Введение индекса, отвечающего за выбор
59      % временного слоя на сетке с номером s,
60      % совпадающего с соответствующим
61      % временным слоем базовой сетки. На данный момент
62      % будем отслеживать совпадение t_m на сгущённой сетке
63      % с t_{m_basic} на базовой сетке
64      m_basic = 2;
65
66      for m = 1:M
67
68          % Реализация схемы CROS1
69
70          w_1 = (D(h,N) - (1+1i)/2*(t(m + 1) - ...
71              t(m))*f_y(y,h,N))\f(y,h,N);
72
73          y = y + (t(m + 1) - t(m))*real(w_1)';
74
75          % Выполнение проверки совпадения t_{m+1}
76          % на сгущённой сетке с t_{m_basic} базовой сетки
77          if (m + 1) == (m_basic - 1)*r_t^(s - 1) + 1
78
79              % Заполнение массива сеточных значений решения
80              % исходной задачи для УрЧП
81
82              % Учитываются левые граничные условия
83              u(m_basic,1) = 0;
84              if s==1
85                  u(m_basic,2) = 1/2*y(1);
86              else
87                  u(m_basic,2) = y(r_x^(s - 1) - 1);
88              end
89
90              % На текущем временном слое выбираются
91              % пространственные узлы, совпадающие
92              % с узлами базовой сетки
93              % (кроме граничных, которые уже учтены выше)
94              for n = 3:(N_0 + 1)
95                  u(m_basic,n) = y((n - 1)*r_x^(s - 1)- 1);
96              end
97
```

```
98                  % Теперь будет отслеживаться совпадение  t_{m+1}
99                  % на сгущённой сетке с очередным  t_{m_basic}
100                 % базовой сетки
101                 m_basic = m_basic + 1;
102
103             end
104
105         end
106
107 end
```

Замечание. Отметим некоторые особенности функции PDESolving.

1. В функции уже реализована возможность поиска приближённого численного решения на последовательности сгущающихся сеток, включая выбор сеточных значений только из узлов, совпадающих с узлами базовой сетки. Эта особенность нам потребуется при реализации численной диагностики разрушения решения, которая рассматривается в следующем разделе. Сейчас же мы будем использовать эту функцию для вычисления решения только на одной (базовой) сетке. Этот случай соответствует значению входного параметра $s := 1$, поэтому значения параметров r_x и r_t не существенны и ни на что на данный момент не влияют.

   На рис. 2.1 приведена схема, отображающая структуру вектора $\boldsymbol{y}(t_m)$ и поясняющая строки кода 82–88, в которых реализован выбор узлов, совпадающих с узлами базовой сетки.

2. С целью экономии памяти (что критично при больших значениях s) в памяти хранится только набор сеточных значений вектора $\boldsymbol{y}(t_m)$ в текущий расчётный момент времени.

3. Необходимо обратить внимание на то, что при обращении к компонентам вектора x все индексы сдвинуты на $+1$ (по сравнению с аналитическими формулами выше), так как

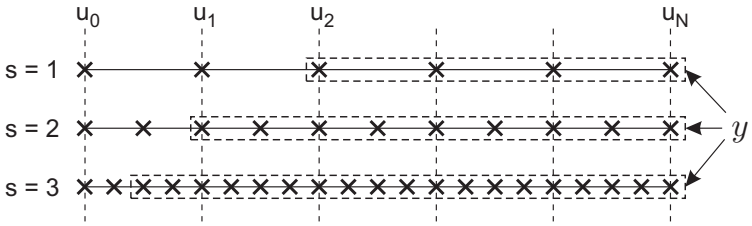

Рис. 2.1: Структура вектора $\boldsymbol{y}(t_m)$ на различных сетках.

в MatLab'е нумерация элементов массивов начинается с 1 (поэтому $x_0 \equiv x(1)$, $x_1 \equiv x(2)$, ..., $x_N \equiv x(N+1)$).

Запуск функции PDESolving может быть выполнен с помощью, например, следующего набора команд, оформленных в виде отдельного MatLab-файла test_2_1_PDESolving.m с расширением «.m», содержимое которого следующее:

```
% Определение начального и конечного времени счёта
t_0 = 0; T = 1.667;

% Определение границ отрезка  x ∈ [a,b]
a = 0; b = 1;

% Определение числа интервалов базовой сетки
N = 50; M = 50;

% Определение начального условия
u_init = @(x) -100*sin(pi*x)^100+100*x^2;

s = 1;   % Номер сетки (только базовая)
r_x = 2; % Коэффициент сгущения сетки по x
r_t = 2; % Коэффициент сгущения сетки по t

u = PDESolving(a,b,N,t_0,T,M,u_init,s,r_x,r_t);


% Отрисовка решения
figure;
```

```
x = a:(b - a)/N:b; % Определение базовой сетки по x
for m = 0:M
    % Рисуется график начального условия
    plot(x,u(1,:),'--k','LineWidth',1); hold on;
    % Рисуется график решения в момент времени t_m
    plot(x,u(m + 1,:),'-ok',...
        'MarkerSize',3,'LineWidth',1); hold on;
    axis([a b -120 120]); xlabel('x'); ylabel('u');
    hold off; drawnow; pause(0.1);
end
```

Данный набор команд позволит получить решение для следующего набора параметров задачи (2.1):

$$\begin{aligned} &a = 0, \quad b = 1, \quad t_0 = 0, \quad T = 1.667, \\ &u_{init}(x) = -100\sin(\pi x)^{100} + 100x^2, \end{aligned} \tag{2.4}$$

с параметрами сеток по пространству и времени:

$$N = 50, \quad M = 50. \tag{2.5}$$

Отметим, что время T взято как теоретическая верхняя оценка времени разрушения, т.е. нам a proiri известно, что при выборе параметров согласно (2.4) на отрезке $[t_0, T]$ произойдёт разрушение решения.

На рис. 2.2 приведено несколько наборов сеточных значений функций $u(x, t_m)$ для отдельных моментов времени t_m.

2.2. Оптимизация численного счёта

Для численной диагностики факта разрушения решения нам будет необходимо проводить вычисления на последовательности сгущающихся сеток (в общем случае сгущение каждый раз производится в r_x раз по пространственной переменной x и в r_t раз — по временной t). С увеличением номера s сетки из последовательности $s = \overline{1, S}$ размеры сетки $X_{r_x^{s-1}N} \times T_{r_t^{s-1}M}$ быстро растут, что приводит к значительному увеличению времени работы

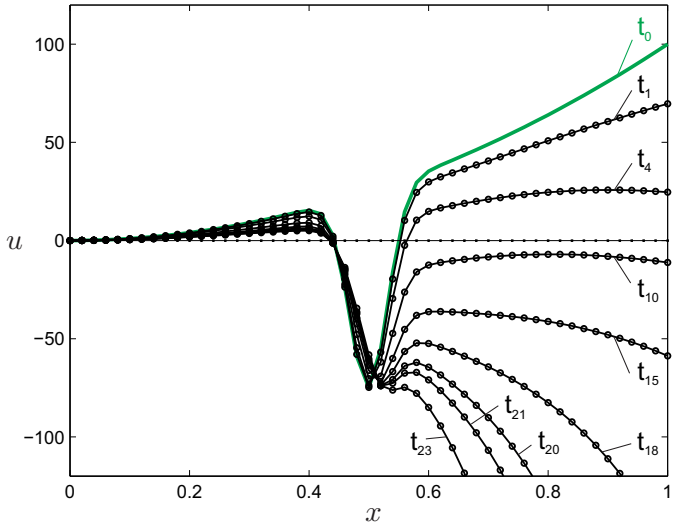

Рис. 2.2: Пример решения задачи (2.1) для набора параметров (2.4)–(2.5) по схеме (2.3). На рисунке отображено несколько наборов сеточных значений функций $u(x, t_m)$ для отдельных моментов времени t_m.

программы, а иногда и к нехватке компьютерной памяти. Существенное увеличение времени счёта в первую очередь связано с тем, что при решении СЛАУ (2.3) общим методом Гаусса необходимо совершить порядка $O(N^3)$ операций, где N — размерность решаемой системы. Поскольку решать СЛАУ приходится при каждом переходе на следующий временной слой, итоговое время работы программы пропорционально кубу размерности сетки по пространству и первой степени размерности сетки по времени. Однако матрица системы (2.3) имеет специальный вид — её внутренняя структура схематично изображена на рис. 2.3 (ненулевые элементы расположены только на диагонали и двух нижних кодиагоналях). Это позволяет разработать алгоритм реше-

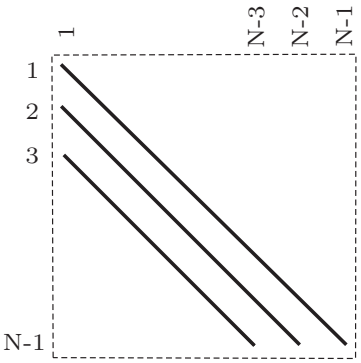

Рис. 2.3: Структура матрицы СЛАУ (2.3).

ния СЛАУ такого специального вида за $O(N^1)$ операций, который является гораздо более экономичной реализацией метода Гаусса для решения СЛАУ с матрицами специального вида как в плане времени выполнения вычислений (трудоёмкость равна $O(N^1)$ операций), так и в плане требуемой для работы алгоритма памяти (что критично при вычислениях на очень густых сетках в случае большого значения S). Главное, что итоговое время работы программы будет пропорционально первой степени размерности сетки по пространству и первой степени размерности сетки по времени. В дальнейшем будем условно называть разработанный алгоритм вырожденным методом Гаусса (это название не является общепринятым!).

Итак, пусть имеется СЛАУ вида $AX = B$ с матрицей A специального вида, изображённого на рис. 2.3. Для простоты будем считать, что размеры матрицы равны $N \times N$. Матрицу A будем хранить в памяти не в виде двумерного массива (что требует хранения $N \times N$ элементов), а в виде трёх массивов, которые содержат ненулевые элементы матрицы, расположенные на соответствующих диагоналях (что требует хранения $3 \times N$ элемен-

тов).

Замечание. Из рис. 2.3 видно, что диагонали имеют разные длины. Однако для удобства программной реализации алгоритма мы будем хранить их значения в массивах одинаковой длины N. При этом элементы первой кодиагонали будем записывать в соответствующий массив начиная со 2-го элемента массива, а элементы второй кодиагонали — в соответствующий массив начиная с 3-го элемента массива.

Ниже приводится пример MatLab-функции, которая реализует один из возможных алгоритмов решения такой СЛАУ $AX = B$ с матрицей специального вида (рис. 2.3), используя в качестве входных данных только 4 одномерных массива длины N: первые 3 содержат ненулевые элементы матрицы A, расположенные на диагонали и двух кодиагоналях, а четвёртый содержит вектор правой части.

```
function X = SpecialMatrixAlgorithm...
    (diag_m,diag_d_1,diag_d_2,B)

    % Функция реализует решение СЛАУ AX = B
    % с матрицей A специального вида, которая содержит
    % ненулевые элементы только на диагонали и
    % двух нижних кодиагоналях

    % Входные параметры:
    % diag_m, diag_d_1, diag_d_2 -
    % - массивы длины N, содержащие ненулевые элементы
    % матрицы A, расположенные на диагонали и
    % двух нижних кодиагонялях
    % (элементы массивов diag_d_1(1), diag_d_2(1)
    % и diag_d_2(2) не используются)
    % B - вектор правой части длины N

    N = length(B);
    X = zeros(N,1);

    % Система приводится к системе с
    % матрицей диагонального вида
    % за счёт обнуления элементов матрицы A,
```

```
    % лежащих ниже главной диагонали
    for n = 1:(N - 2)
        c = diag_d_1(n + 1)/diag_m(n);
        B(n + 1) = B(n + 1) - c*B(n);
        c = diag_d_2(n + 2)/diag_m(n);
        B(n + 2) = B(n + 2) - c*B(n);
    end
    c = diag_d_1(N)/diag_m(N - 1);
    B(N) = B(N) - c*B(N - 1);

    % Вычисляется решение X
    for n = 1:N
        X(n) = B(n)/diag_m(n);
    end

end
```

Таким образом, для применения разработанного вырожденного метода Гаусса для решения СЛАУ (2.3) необходимо сначала подготовить массивы, которые будут содержать ненулевые элементы, расположенные на соответствующих диагоналях матрицы

$$\left[\boldsymbol{D} - \frac{1+i}{2}\left(t_{m+1} - t_m\right)\boldsymbol{f_y}\Big(\boldsymbol{y}(t_m)\Big)\right].$$

Ниже приводится пример соответствующей MatLab-функции, которая по сути является небольшой модификацией функций D и f_y, выписанных ранее.

```
function [diag_m,diag_d_1,diag_d_2] = ...
    DiagonalsPreparation(y,tau,h,N)

    % Функция подготавливает массивы,
    % которые содержат ненулевые элементы матрицы
    % [D - (1+1i)/2*tau*f_y(y)],
    % расположенные на диагонали и
    % двух нижних кодиагоналях

    % Входные данные:
    % y - вектор решения системы ОДУ
    % на текущем временном слое
```

```
13      % tau - текущий шаг по времени
14      % h - шаг сетки по переменной x
15      % N - число интервалов сетки по переменной x
16
17      % Выходные параметры:
18      % diag_m, diag_d_1, diag_d_2 - искомые массивы
19
20      % Выделяем память под искомые массивы
21      diag_m = zeros(1,N-1);
22      diag_d_1 = zeros(1,N-1);
23      diag_d_2 = zeros(1,N-1);
24
25      diag_m(1) = 1/4*(2 - h^2) - ...
26          (1+1i)/2*tau*(-1/2 - h/4*y(1));
27      diag_m(2) = 1 - (1+1i)/2*tau*(-1 - h/2*y(1));
28      diag_d_1(2) = -(7/4 + h^2) - ...
29          (1+1i)/2*tau*(7/4 - h/2*(y(2) - 1/2*y(1)));
30      for n = 3:(N - 1)
31          diag_m(n) = 1 - (1+1i)/2*tau*(-1 - h/2*y(n - 1));
32          diag_d_1(n) = -(2 + h^2) - (1+1i)/2*tau*...
33              (2 - h/2*(y(n) - y(n - 2)));
34          diag_d_2(n) = 1 - (1+1i)/2*tau*...
35              (-1 + h/2*y(n - 1));
36      end
37
38  end
```

В результате для использования этих функций в MatLab-функции PDESolving необходимо заменить кусок кода, находящийся в строках 68–73, на набор следующих команд (при этом функции D и f_y больше не будут использоваться).

```
1       % Реализация схемы CROS1
2
3       % Подготовка массивов, которые содержат
4       % ненулевые элементы матрицы
5       % [D - (1+1i)/2*tau*f_y(y)],
6       % расположенные на диагонали и
7       % двух нижних кодиагоналях
8       [diag_m, diag_d_1, diag_d_2] = ...
9           DiagonalsPreparation(y, t(m + 1) - t(m), h, N);
10
```

```
11      % w_1 ищется с помощью вырожденного метода Гаусса
12      w_1 = SpecialMatrixAlgorithm...
13          (diag_m,diag_d_1,diag_d_2,f(y,h,N));
14
15      y = y + (t(m + 1) - t(m))*real(w_1)';
```

Сравним количество операций, которые необходимо произвести при переходе на следующий временной слой при поиске приближённого решения задачи (2.1) в двух случаях: при использовании общего метода Гаусса и при применении вышеописанного модифицированного алгоритма.

Как уже упоминалось ранее, в случае общего метода Гаусса время работы алгоритма пропорционально кубу размерности матрицы системы (2.3). Т. е. в нашем случае это время пропорционально

$$\sim \frac{2}{3}\big(N-1\big)^3 \sim \frac{2}{3}N^3 \sim O(N^3).$$

В случае предложенной реализации вырожденного метода Гаусса время работы программы будет пропорционально

$$\sim 7N - 9 \sim O(N^1).$$

Таким образом, выигрыш во времени счёта в случае реализации вырожденного метода Гаусса колоссален.

2.3. Контроль правильности реализации численного алгоритма в виде программного кода

При разработке численных алгоритмов и последующем написании программ любой человек может допустить ошибку (и, скорее всего, не одну). Соответственно, необходимо владеть методами проверки правильности как численного алгоритма, так и его программной реализации. Классический способ такого контроля заключается в проведении тестовых расчётов для такого набора входных данных, для которых известно точное решение.

Тогда в случае корректной реализации алгоритмов с увеличением размерности сеток, использующихся для поиска решения, приближённое численное решение должно сходиться к точному, что можно отследить визуально.

Одной из особенностей рассматриваемого нами класса задач является невозможность построения модельных примеров с известным точным решением для какого-либо набора входных данных задачи. В связи с этим мы предлагаем использовать следующий подход.

Изменим постановку задачи (2.1) таким образом, чтобы получить возможность построить пример с известным решением, но при этом внести минимальные изменения в уже разработанную схему численного решения задачи. Для этого введём в правую часть исходного уравнения неоднородность $f_{model}(x,t)$:

$$\begin{cases} \dfrac{\partial}{\partial t}\big(u_{xx} - u\big) + u_{xx} + uu_x = f_{model}(x,t), \quad x \in (a,b], \quad t \in (t_0,T], \\ u(a,t) = 0, \quad u_x(a,t) = 0, \quad t \in (t_0,T], \\ u(x,t_0) = u_{init}(x), \quad x \in [a,b]. \end{cases}$$

Это даст возможность задавать в качестве точного модельного решения любую функцию $u_{model}(x,t)$, которая удовлетворяет граничным условиям. Например, если мы хотим, чтобы точным решением видоизменённой задачи являлась функция

$$u_{model}(x,t) = e^{-t}\big(100x^2 - 100\sin(\pi x)^{100}\big),$$

определённая на отрезке $[a,b] \equiv [0,1]$, мы можем подставить её в систему и в результате определить, что

$$\begin{aligned} f_{model}(x,t) = {} & e^{-t}\big(100x^2 - 100\sin^{100}(\pi x)\big) + \\ & + e^{-2t}(200x - 10\,000\pi\cos(\pi x)\sin^{99}(\pi x)\big) \times \\ & \times \big(100x^2 - 100\sin^{100}(\pi x)\big), \\ u_{init}(x) = {} & e^{-t_0}\big(100x^2 - 100\sin^{100}(\pi x)\big). \end{aligned}$$

В результате такого видоизменения исходной задачи (2.1) система ОДУ (2.2) перестанет быть автономной, так как её правая часть — вектор-функция $\boldsymbol{f}$ — будет зависеть от временной переменной t (которая присутствует в неоднородности $f_{model}(x,t)$):

$$\begin{cases} \boldsymbol{D}\dfrac{d\boldsymbol{y}}{dt} = \boldsymbol{f}(\boldsymbol{y},t), \quad t \in (t_0, T], \\ \boldsymbol{y}(t_0) = \boldsymbol{y}_{init}, \end{cases}$$

А именно, вектор-функция $\boldsymbol{f}$ будет иметь следующую структуру:

$$f_n = \begin{cases} -\frac{1}{2}y_1 - \dfrac{h}{8}y_1^2 + f_{model}(x_1,t)\,h^2, & \text{если } n = 1, \\ -\Big(y_2 - \frac{7}{4}y_1\Big) - \dfrac{h}{2}y_1\Big(y_2 - \frac{1}{4}y_1\Big) + \\ \qquad\qquad + f_{model}(x_2,t)\,h^2, & \text{если } n = 2, \\ -\big(y_n - 2y_{n-1} + y_{n-2}\big) - \\ -\dfrac{h}{2}y_{n-1}\big(y_n - y_{n-2}\big) + f_{model}(x_n,t)\,h^2, & \text{если } n = \overline{3, N-1}. \end{cases}$$

Ниже приводится пример соответствующей модификации MatLab-функции f, основным отличием которой от введённой ранее является наличие дополнительного входного аргумента — времени t.

```
function f = f(y,x,h,N,t)

    % Функция вычисляет вектор правой части
    % решаемой системы ОДУ

    % Входные данные:
    % y - вектор решения системы ОДУ
    % на текущем временном слое
    % x - сетка по переменной x
    % h - шаг сетки по переменной x
    % N - число интервалов сетки по переменной x
    % t - текущий момент времени
```

```
13
14      % Выходные данные:
15      % f - искомый вектор f
16
17      f = zeros(N - 1,1);
18
19      f(1) = -1/2*y(1) - h/8*y(1)^2 +...
20          f_model(x(2),t)*h^2;
21      f(2) = -(y(2) - 7/4*y(1)) - ...
22          h/2*y(1)*(y(2) - 1/4*y(1))+ ...
23          f_model(x(3),t)*h^2;
24      for n = 3:(N - 1)
25          f(n) = -(y(n) - 2*y(n - 1) + y(n - 2)) + ...
26              -h/2*y(n - 1)*(y(n) - y(n - 2)) + ...
27              f_model(x(n + 1),t)*h^2;
28      end
29
30  end
```

Замечание. Вновь обращаем внимание на то, что при обращении к компонентам вектора x все индексы сдвинуты на $+1$ (по сравнению с аналитическими формулами выше), так как в MatLab'е нумерация элементов массивов начинается с 1 (поэтому $x_0 \equiv x(1)$, $x_1 \equiv x(2)$, $\ldots$, $x_N \equiv x(N+1)$).

Также здесь используется MatLab-функция f_model, которая задаёт модельную функцию $f_{model}(x,t)$. Она может быть реализована, например, в следующем виде:

```
1   function f_model = f_model(x,t)
2
3       % Функция задаёт модельную функуию  f_model(x,t)
4
5       f_model = exp(-t)*(100*x^2 - 100*sin(pi*x)^100) + ...
6           exp(-2*t)*(200*x -10000*pi*cos(pi*x)*...
7           sin(pi*x)^99)*(100*x^2 - 100*sin(pi*x)^100);
8   end
```

В связи с тем, что система ОДУ (2.2) теперь не является автономной (что означало отсутствие явной зависимости правой части $\boldsymbol{f}$ от t), одностадийная схема Розенброка с комплексным

коэффициентом CROS1 будет иметь следующий вид:

$$\boldsymbol{y}(t_{m+1}) = \boldsymbol{y}(t_m) + (t_{m+1} - t_m)\,\mathrm{Re}\,\boldsymbol{w}_1\,,$$

где $\boldsymbol{w}_1$ является решением СЛАУ

$$\left[\boldsymbol{D} - \frac{1+i}{2}(t_{m+1} - t_m)\,\boldsymbol{f}_{\boldsymbol{y}}\Big(\boldsymbol{y}(t_m), t_m\Big)\right]\boldsymbol{w}_1 = \boldsymbol{f}\Big(\boldsymbol{y}(t_m), \frac{t_m + t_{m+1}}{2}\Big).$$

Отметим, что порядок аппроксимации этой схемы по переменной t по-прежнему равен 2.

В результате видоизменённая MatLab-функция PDESolving будет отличаться от исходной только тем, что кусок кода, находящийся в строках 68–73 и отвечающий за реализацию схемы CROS1, необходимо будет заменить на следующий набор команд:

```
1   % Реализация схемы CROS1
2
3   w_1 = (D(h,N) - (1+1i)/2*(t(m + 1) - t(m))* ...
4   f_y(y,h,N))\ f(y,x,h,N,(t(m + 1) + t(m))/2);
5
6   y = y + (t(m + 1) - t(m))*real(w_1)';
```

Или, в случае использования вырожденного метода Гаусса:

```
1   % Реализация схемы CROS1
2
3   % Подготовка массивов, которые содержат
4   % ненулевые элементы матрицы
5   % [D - (1+1i)/2*tau*f_y(y)],
6   % расположенные на диагонали и
7   % двух нижних кодиагоналях
8   [diag_m,diag_d_1,diag_d_2] = ...
9       DiagonalsPreparation(y,t(m + 1) - t(m),h,N);
10
11  % w_1 ищется с помощью вырожденного метода Гаусса
12  w_1 = SpecialMatrixAlgorithm...
13      (diag_m,diag_d_1,diag_d_2,...
14      f(y,h,N,(t(m + 1) + t(m))/2));
15
16  y = y + (t(m + 1) - t(m))*real(w_1)';
```

И, наконец, в MatLab-функции draw необходимо после 16-й строчки вставить следующий кусок кода, который позволит отобразить на одном графике как найденное приближённое численное решение, так и точное модельное, что позволит визуально убедиться в сходимости приближённого решения к точному при измельчении сеток, на которых ищется приближённое решение.

```
% Рисуется график начального условия
t = t_0:(T - t_0)/M:T;  % Определение базовой сетки по t
u_model = exp(-t(m + 1))*...
(-100*sin(pi*x).^100 + 100*x.^2);
plot(x(1:k:N+1),u_model,'-*g',...
     'LineWidth',1); hold on;
```

Отметим, что при таком построении модельного примера с известным точным решением небольшому видоизменению подлежит только функция $\boldsymbol{f}$ и схема CROS1. После контроля правильности реализации соответствующих численных алгоритмов, положив $f_{model}(x,t) \equiv 0$, мы приводим систему к исходному виду (2.1).

2.4. Численная диагностика разрушения решения

Обобщим методику численной диагностики разрушения решения, которую мы описали в главе 1 для задачи Коши для обыкновенного дифференциального уравнения, на случай рассматриваемой в этом разделе начально-краевой задачи для уравнения в частных производных псевдопараболического типа.

Предположим, что мы нашли сеточное решение начально-краевой задачи (2.1) с использованием схемы с порядком точности p_x по пространственной переменной x и с порядком точности p_t по временной переменной t на равномерной базовой сетке $X_N \times T_M$: $X_N \times T_M = \{(x_n, t_m),\ 0 \leqslant n \leqslant N,\ 0 \leqslant m \leqslant M :\ x_n = a + n\,h,\ t_m = t_0 + m\tau,\ h = (b-a)/N,\ \tau = (T-t_0)/M\}$. Это

означает, что для всех узлов $(x,t) \in X_N \times T_M$ верно равенство

$$u(x,t) = u^{(N,M)}(x,t) + O(h^{p_x} + \tau^{p_t}). \tag{2.6}$$

Здесь верхний индекс (N,M) означает, что $u^{(N,M)}(x,t)$ найдено численно на сетке с размерности $N \times M$ интервалов соответственно по x и t.

Теперь перепишем соотношение (2.6) в виде

$$u(x,t) = u^{(N,M)}(x,t) + c_x(x,t)h^{p_x} + c_t(x,t)\tau^{p_t} + \\ + O(h^{p_x+1} + \tau^{p_t+1}), \tag{2.7}$$

явно выделяя главный член $R^{(N,M)}(t) \equiv c_x(x,t)h^{p_x} + c_t(x,t)\tau^{p_t}$ разложения ошибки $u(x,t) - u^{(N,M)}(x,t)$ приближённого вычисления $u(x,t)$ в ряд Тейлора. При этом будем исходить из предположения о существовании соответствующих непрерывных производных решения.

Предполагается, что вычисление $u^{(N,M)}(x,t)$, $(x,t) \in X_N \times T_M$, мы проводили на фиксированной сетке с N и M интервалами (через число которых однозначно определяются шаги сетки h и τ по каждой из переменных) по схеме с известными порядками точности p_x и p_t по пространственной и временной переменным соответственно. Таким образом, уравнение (2.7) содержит три неизвестных: $u(x,t)$, $c_x(x,t)$ и $c_t(x,t)$. Для того чтобы найти три неизвестных, нам необходимо три уравения. Однако, если мы будем рассматривать комбинацию слагаемых $c_x(x,t)h^{p_x} + c_t(x,t)\tau^{p_t}$ в качестве одной неизвестной, нам будет необходимо иметь только два уравнения. Получим второе уравнение, проведя вычисления на сгущённой в r_x раз по пространственной переменной x и в r_t по временной переменной t сетке (т.е. на сетке, содержащей $r_x N$ и $r_t M$ интервалов на отрезках $[a,b]$ и $[t_0,T]$ соответственно):

$$u(x,t) = u^{(r_x N, r_t M)}(x,t) + c_x(x,t)\Big(\frac{h}{r_x}\Big)^{p_x} + c_t(x,t)\Big(\frac{\tau}{r_t}\Big)^{p_t} + \\ + O\bigg(\Big(\frac{h}{r_x}\Big)^{p_x+1} + \Big(\frac{\tau}{r_t}\Big)^{p_t+1}\bigg). \tag{2.8}$$

Вычтем из уравнения (2.8) уравнение (2.7) и из полученного равенства попытаемся выразить $c_x(x,t)h^{p_x} + c_t(x,t)\tau^{p_t}$. Это удастся сделать, только если в правой части уравнения (2.8) слагаемые $c_x(x,t)h^{p_x}$ и $c_t(x,t)\tau^{p_t}$ будут иметь общий множитель. Это возможно только в том случае, если выполнено соотношение

$$r_x{}^{p_x} = r_t{}^{p_t}, \tag{2.9}$$

что определяет согласование коэффицентов сгущения сеток r_x и r_t по различным переменным с порядками точности схемы p_x и p_t (более подробное обсуждение особенностей выбора коэффициентов сгущения в случае многомерных задач см. в п. 4 § 2 главы II [6]).

В результате при выполнении соотношения (2.9) мы получим

$$\begin{aligned} c_x(x,t)h^{p_x} + c_t(x,t)\tau^{p_t} = \\ = \frac{u^{(r_xN,r_tM)}(x,t) - u^{(N,M)}(x,t)}{r_t{}^{p_t} - 1}\frac{r_t{}^{p_t}}{\tau^{p_t}} + O(h^1 + \tau^1). \end{aligned} \tag{2.10}$$

Таким образом,

$$\begin{aligned} R^{(r_xN,r_tM)}(x,t) \equiv c_x(x,t)\Big(\frac{h}{r_x}\Big)^{p_x} + c_t(x,t)\Big(\frac{\tau}{r_t}\Big)^{p_t} = \\ = \frac{u^{(r_xN,r_tM)}(x,t) - u^{(N,M)}(x,t)}{r_t{}^{p_t} - 1} + O(h^{p_x+1} + \tau^{p_t+1}). \end{aligned} \tag{2.11}$$

Дробь, стоящая в правой части равенства (2.11), является апостериорной асимптотически точной оценкой главного (т.е. наименее быстро убывающего) члена разложения в ряд Тейлора точного решения $u(x,t)$, или, что то же самое, апостериорной асимптотически точной оценкой главного члена разложения ошибки $u(x,t) - u^{(r_xN,r_tM)}(x,t)$ приближённого вычисления $u(x,t)$ в ряд Тейлора. Таким образом,

$$u(x,t) = u^{(r_xN,r_tM)}(x,t) + R^{(r_xN,r_tM)}(x,t) + O(h^{p_x+1} + \tau^{p_t+1}).$$

Исходя из того, что при $(h,\tau)\to 0$ слагаемое $R^{(r_xN,r_tM)}(x,t)$ превосходит все остальные члены разложения в ряд Тейлора ошибки $u(x,t)-u^{(r_xN,r_tM)}(x,t)$, его можно воспринимать как апостериорную асимптотически точную оценку погрешности вычисления $u^{(r_xN,r_tM)}(x,t)$. Далее будем под $R^{(r_xN,r_tM)}(x,t)$ подразумевать, опуская $O(h^{p_x+1}+\tau^{p_t+1})$ (но при этом не забывая об асимптотическом характере соответствующей формулы), величину

$$R^{(r_xN,r_tM)}(x,t)=\frac{u^{(r_xN,r_tM)}(x,t)-u^{(N,M)}(x,t)}{r_t{}^{p_t}-1}, \tag{2.12}$$

что является обобщением классической формулы Рунге–Ромберга (1.10) на случай вычислений функции двух переменных.

Проведя расчёты ещё на одной сетке — с числом интервалов r_x^2N и r_t^2M по соответствующим переменным, — мы сможем вычислить

$$\begin{aligned}R^{(r_x^2N,r_t^2M)}(x,t)&\equiv c_x(x,t)\Big(\frac{h}{r_x^2}\Big)^{p_x}+c_t(x,t)\Big(\frac{\tau}{r_t^2}\Big)^{p_t}=\\&=\frac{u^{(r_x^2N,r_t^2M)}(x,t)-u^{(r_xN,r_tM)}(x,t)}{r_t{}^{p_t}-1}+O(h^{p_x+1}+\tau^{p_t+1}).\end{aligned}$$

Заметим, что

$$\begin{aligned}R^{(r_xN,r_tM)}(x,t)&\equiv c_x(x,t)\Big(\frac{h}{r_x}\Big)^{p_x}+c_t(x,t)\Big(\frac{\tau}{r_t}\Big)^{p_t},\\R^{(r_x^2N,r_t^2M)}(x,t)&\equiv c_x(x,t)\Big(\frac{h}{r_x^2}\Big)^{p_x}+c_t(x,t)\Big(\frac{\tau}{r_t^2}\Big)^{p_t},\end{aligned} \tag{2.13}$$

и из отношения норм этих двух выражений при условии выполнения соотношения (2.9) найдём выражение для эффективного порядка точности

$$p^{eff}=\log_{r_t}\frac{\|R^{(r_xN,r_tM)}(x,t)\|_{X_N\times T_M}}{\|R^{(r_x^2N,r_t^2M)}(x,t)\|_{X_N\times T_M}}, \tag{2.14}$$

с которым вычисляется приближённое решение на промежутке времени $t \in [t_0, T]$ и которое обладает свойством: ${p_t}^{eff} \to {p_t}^{theor} \equiv p_t$ при $(h, \tau) \to 0$. Нормы в этих выражениях суть евклидовы нормы, взятые как корень из суммы квадратов значений на базовой сетке $X_N \times T_M$. Отметим, что мы говорим именно об *эффективном* порядке точности, потому что он вычисляется исходя из имеющихся приближённых решений в пренебрежении приближённым характером формулы (2.12) (сравните с (2.11)).

Эффективный порядок точности может быть вычислен и по отдельности для каждого слоя $t \in T_M$ сетки. Соответствующую формулу можно получить из отношения норм выражений в (2.13):

$$ {p_t}^{eff}(t) = \log_{r_t} \frac{\|R^{(r_x N, r_t M)}(x,t)\|_{X_N}}{\|R^{(r_x^2 N, r_t^2 M)}(x,t)\|_{X_N}}, \tag{2.15} $$

где нормы берутся уже только по сетке X_N, а $t \in T_M$ является параметром.

Также эффективный порядок точности может быть вычислен и поточечно для каждого отдельного $x \in X_N$ в фиксированный момент времени t. Сооветствующую формулу можно получить из отношения выражений в (2.13) в каждом узле базовой сетки по отдельности:

$$ {p_{xt}}^{eff}(x,t) = \log_{r_t} \frac{R^{(r_x N, r_t M)}(x,t)}{R^{(r_x^2 N, r_t^2 M)}(x,t)}. \tag{2.16} $$

Следует понимать, что аргумент логарифма в (2.16) может оказаться отрицательным, если члены более высокого порядка в разложении ошибки по Тейлору преобладают над формально главным членом порядка p. Такое может произойти, например, если коэффициент в главном члене случайно оказался равным нулю в конкретной задаче в конкретной точке либо если сетка ещё слишком грубая. Поэтому на практике, чтобы избежать аварийной остановки расчёта, имеет смысл в (2.16) взять модули числителя и знаменателя.

Теперь опишем практический алгоритм выполнения оценок эффективного порядка точности, что в дальнейшем позволит диагностировать факт разрушения решения в случае его наличия.

Замечание. Мы аппроксимировали все пространственные производные в (2.1) с точностью $O(h^2)$, а при численном интегрировании системы (2.2) используем схему CROS1 (2.3), которая имеет точность $O(\tau^2)$. Поэтому построенный метод решения системы (2.1) имеет точность $O(\tau^2 + h^2)$, т.е. $p_x = p_x^{theor} \equiv 2$ и $p_t = p_t^{theor} \equiv 2$. Таким образом из условия (2.9) следует, что коэффициенты сгущения r_x и r_t по разным переменным для выполнения условий применимости формулы Рунге–Ромберга (2.12) должны удовлетворять соотношению $r_x = r_t$. Наиболее удобно для счёта выбрать $r_x = r_t \equiv 2$.

Для начала введём базовую сетку $X_N \times T_M$: $\{x_n, t_m\}$, $0 \leqslant n \leqslant N$, $0 \leqslant m \leqslant M$. После этого произведём последовательное сгущение сетки, начиная с базовой, и вычислим решения

$$u_{(s)}(x,t) \equiv u^{(r_x^{s-1}N, r_t^{s-1}M)}(x,t)$$

на полученных сетках $X_{r_x^{s-1}N} \times T_{r_t^{s-1}M}$ (s — номер сетки из набора сеток $s = \overline{1,S}$) по схеме с порядком точности p_x по пространственной переменной x и порядком точности p_t по временной переменной t.

В этом случае, если, как и отмечено выше, r_x и r_t целые, каждая последующая сетка $X_{r_x^{s-1}N} \times T_{r_t^{s-1}M}$ имеет узлы, совпадающие с узлами базовой сетки $(x_n, t_m) \in X_N \times T_M$, $0 \leqslant n \leqslant N$, $0 \leqslant m \leqslant M$. В этих узлах (x,t) мы можем выполнить апостериорную асимптотически точную оценку погрешности на каждой сетке с номером s по формуле Рунге–Ромберга (2.12)

$$\Delta_{(s)}(x_n,t_m) \equiv R^{(r_x^{s-1}N, r_t^{s-1}M)}(x_n,t_m) = \\ = \frac{u_{(s)}(x_n,t_m) - u_{(s-1)}(x_n,t_m)}{r_t^{p_t} - 1},$$

и оценить эффективный порядок точности (2.14) на всём проме-

жутке времени $t \in [t_0, T]$

$$p_{t_{(s)}}^{\ eff} = \log_{r_t} \frac{\sqrt{\sum\limits_{n=0}^{N}\sum\limits_{m=0}^{M}\left(u_{(s-1)}(x_n,t_m) - u_{(s-2)}(x_n,t_m)\right)^2}}{\sqrt{\sum\limits_{n=0}^{N}\sum\limits_{m=0}^{M}\left(u_{(s)}(x_n,t_m) - u_{(s-1)}(x_n,t_m)\right)^2}} \quad (2.17)$$

(здесь выбрана наиболее удобная из возможных норм — евклидова). В случае, если на всём промежутке времени $t \in [t_0, T]$ решение задачи имеет непрерывные производные порядка p_x по x и p_t по t, имеет место сходимость

$$p_{t_{(s)}}^{\ eff} \xrightarrow[s\to\infty]{} p_t{}^{theor} \equiv p_t.$$

Нарушение этой сходимости говорит о потере гладкости точного решения на промежутке времени $t \in [t_0, T]$. Другими словами, теоретические порядки точности, равные соответственно p_x и p_t, характеризует способность соответствующей численной схемы передать все члены разложения в ряд Тейлора точного решения, младшие по сравнению с $c_x h^{p_x} + c_t \tau^{p_t}$. А эффективный порядок точности позволяет нам судить о том, были ли реально переданы все члены, младшие по сравнению с $c_x h^{p_x} + c_t \tau^{p_t}$. В частности, в случае $p_t{}^{eff} \leqslant 0$ можно сделать вывод об отсутвии возможности разложить точное решение в ряд Тейлора, что говорит об отсутствии решения (или, другими словами, о факте разрушения решения в какой-либо точке промежутка $t \in [t_0, T]$) или потере его гладкости.

С целью локализации конкретного момента времени, в который произошло разрушение решения, можно оценить эффективный порядок точности и поточечно (2.15) в каждом узле

$t_m \in T_M$, $0 \leqslant m \leqslant M$,

$$p_{t}{}_{(s)}^{eff}(t_m) = \log_{r_t} \frac{\sqrt{\sum\limits_{n=0}^{N} \left(u_{(s-1)}(x_n, t_m) - u_{(s-2)}(x_n, t_m)\right)^2}}{\sqrt{\sum\limits_{n=0}^{N} \left(u_{(s)}(x_n, t_m) - u_{(s-1)}(x_n, t_m)\right)^2}} \quad (2.18)$$

В точках t, в которых решение исходной задачи имеет непрерывные производные порядка p_t по времени и порядка p_t по пространству, имеет место сходимость

$$p_{t}{}_{(s)}^{eff}(t) \xrightarrow[s\to\infty]{} p_t{}^{theor} \equiv p_t$$

и соответствующая оценка погрешности является асимптотически точной при $s \to \infty$ (или, что то же самое, $N, M \to \infty$). Нарушение этой сходимости говорит о потере гладкости точного решения. В частности, в случае степенной «сингулярности» $u(x,t) \sim (T_{bl} - t)^{-\beta}$, где T_{bl} — время разрушения (индекс «bl» — сокращение от «blow-up», «разрушение»), для любого $t > T_{bl}$ эффективный порядок точности $p_{t}{}_{(s)}^{eff}(t) \xrightarrow[s\to\infty]{} -\beta$. Это позволяет нам найти соответствующую степень β. Если $p_{t}{}_{(s)}^{eff}(t) \xrightarrow[s\to\infty]{} -\infty$ для любого $t > T_{bl}$, мы можем утверждать, что решение экспоненциально возрастает, т.е. $u(x,t) = \infty$; если $p_{t}{}_{(s)}^{eff}(t) \xrightarrow[s\to\infty]{} 0$ для любого $t > T_{bl}$, то рост решения в окрестности «сингулярности» является логарифмическим: $u(x,t) \sim \ln(T_{bl} - t)$. Вывод соответствующих утверждений можно найти в [2, 9]. Момент разрушения решения T_{bl} может быть найден с точностью до величины интервала базовой сетки по времени T_M.

После локализации момента времени разрушения t можно локализовать и пространственные точки разрушения решения по переменной x для каждого конкретного момента времени t по формуле (2.16) в каждом узле $t_m \in T_M$, $0 \leqslant m \leqslant M$,

$$p_{xt}{}_{(s)}^{eff}(x_n, t_m) = \log_{r_t} \frac{|u_{(s-1)}(x_n, t_m) - u_{(s-2)}(x_n, t_m)|}{|u_{(s)}(x_n, t_m) - u_{(s-1)}(x_n, t_m)|}. \quad (2.19)$$

Далее приводится пример набора MatLab-команд, оформленных в виде отдельного файла test_2_1.m, которые позволяют посредством многократного запуска введённой ранее MatLab-функции PDESolving.m получить набор сеточных решений $u_{(s)}(x,t) \equiv u^{(r_x^{s-1}N, r_t^{s-1}M)}(x,t)$, $s = \overline{1,S}$, задачи (2.1) с набором параметров (2.4) на различных сетках, начиная с базовой сетки $X_N \times T_M$ с $N = 50$ и $M = 50$.

```
% Определение начального и конечного времени счёта
t_0 = 0; T = 1.667;

% Определение границ отрезка  x ∈ [a,b]
a = 0; b = 1;

% Определение числа интервалов базовой сетки
N = 50; M = 50;

% Определение начального условия
u_init = @(x) -100*sin(pi*x)^100+100*x^2;

S = 10;   % Число сеток, на которых ищется
          % приближённое решение
r_x = 2; % Коэффициент сгущения сетки по x
r_t = 2; % Коэффициент сгущения сетки по t

% Выделение памяти под массивы сеточных значений
% решений ОДУ на разных сетках с номерами  s = 1,S
% Первый индекс - номер сетки s из последовательности
% сгущающихся сеток, на которых ищется решение
% Второй и третий индексы определяют массив,
% в строке с номером (m + 1) которого  хранятся
% сеточные значения решения, соответствующего
% моменту времени  t_m , из узлов,
% совпадающих с узлами базовой сетки
array_of_u = zeros(S,M + 1,N + 1);

% "Большой цикл", который пересчитывает решение S раз
% на последовательности сгущающихся сеток
% Массив сеточных значений решения содержит только
% сеточные значения из узлов,
% совпадающих с узлами базовой сетки
```

```
for s = 1:S
    u = PDESolving(a,b,N,t_0,T,M,u_init,s,r_x,r_t);
    array_of_u(s,:,:) = u;
    s
end

% Сохраняем необходимые для дальнейшей диагностики
% разрушения решения данные Workspace'a в файл
save('data.mat','array_of_u','N','M',...
    'r_x','r_t','S','a','b','t_0','T');
```

В связи с тем, что время счёта на каждой очередной сетке возрастает в $r_x r_t$ раз, нахождение приближённых решений на разных сетках реализовано в виде вышеприведённого отдельного кода, который вычисляет все необходимые для дальнейшей численной диагностики разрушения решения данные и записывает их в файл data.mat. Его содержимое будет подгружаться вводимыми далее по ходу изложения материала функциями без повторного расчёта решений на последовательности сгущающихся сеток.

Далее приводится пример MatLab-кода, оформленного в виде отдельного файла BlowUpDiagnostics.m, который вычисляет эффективные порядки точности приближённого решения на промежутке времени $t \in [t_0, T]$ (2.17), используя данные из уже сформированного предыдущей программой файла data.mat.

```
% Загрузка результата вычисления приближённого решения
% на последовательности из S сгущающихся сеток
load('data.mat');

% Выделение памяти под массив значений эффективных
% порядков точности расчёта приближённого решения
% на разных сетках
p_eff = zeros(S,1);

% Вычисление эффективных порядков точности
p_eff(1) = NaN; % Вычисление p_(1)^eff невозможно
p_eff(2) = NaN; % Вычисление p_(2)^eff невозможно
```

```
for s = 3:S
    p_eff(s) = log(...
        sqrt(sum(sum((array_of_u(s - 1,:,:) - ...
        array_of_u(s - 2,:,:))).^2)))/...
        sqrt(sum(sum((array_of_u(s,:,:) - ...
        array_of_u(s - 1,:,:))).^2))))/...
        log(r_t);
end

% Вывод последовательности значений p_(s)^eff
for s = 1:S
    X = ['p^eff_(',int2str(s),')=',...
        num2str(p_eff(s),'%6.4f')];
    disp(X);
end
```

Отметим, что вычисление $p^{eff}_{(1)}$ и $p^{eff}_{(2)}$ невозможно, потому что для нахождения эффективных порядков точности необходимо знать приближённое решение на трёх последовательных сетках (текущая и две предыдущих) (см. формулу (2.17)).

В таблице 2.4 приведены примеры результатов вычислений последовательностей значений $p_{t^{eff}_{(s)}}$ для различных временных интервалов $t \in [t_0, T]$. Хорошо видно, что для тех временных промежутков, для которых решение задачи (2.1) с набором входных данных (2.4) существует, последовательность эффективных порядков точности $p_{t^{eff}_{(s)}}$ сходится к теоретическому порядку точности $p_t{}^{theor} \equiv p$. В случае же отсутствия решения на соответствующем временном промежутке последовательность эффективных порядков точности быстро сходится к неположительному числу.

Теперь от «интегральной» оценки эфективного порядка точности для диагностики факта разрушения «в целом» перейдём к детальной его локализации. Сначала мы найдём момент времени, когда наступает разрушение решение, с точностью до величины порядка шага сетки по времени. Затем перейдём к локализации по пространственной переменной.

Ниже приводится пример MatLab-кода, оформленного в ви-

s	$T = 0.400$	$T = 0.600$	$T = 1.000$	$T = 1.667$
3	1.7837	1.1368	-27.0854	-36.3775
4	1.9415	1.7175	-62.5598	-57.2937
5	1.9851	1.9229	-83.9025	-116.6789
6	1.9963	1.9802	-259.7775	$-\infty$
7	1.9991	1.9950	$-\infty$	$-\infty$
8	1.9998	1.9950	$-\infty$	$-\infty$
9	2.0001	1.9998	$-\infty$	$-\infty$
10	1.9987	1.9985	$-\infty$	$-\infty$

Таблица 2.1: Наборы значений $p_{t_{(s)}}^{eff}$ по итогам расчёта задачи (2.1) с набором входных данных (2.4) для разных временных промежутков $t \in [t_0, T] \equiv [0, T]$ с начальными сетками $X_N \times T_M$ с одинаковым числом интервалов $N = 50$, $M = 50$ и коэффициентами сгущения сеток $r_x = 2$, $r_t = 2$.

де отдельного файла BlowUpDiagnostics_for_each_t.m, который реализует вычисление эффективных порядков точности приближённого решения в узлах, совпадающих с узлами t_m базовой сетки T_M (2.18), используя данные из уже сформированного файла data.mat. Это позволит определить конкретный момент времени, в который происходит разрушение решения.

```
% Загрузка результата вычисления приближённого решения
% на последовательности из S сгущающихся сеток
load('data.mat');

% Выделение памяти под массив значений эффективных
% порядков точности расчёта приближённого решения
% в каждом узле t_m, 1 ⩽ m ⩽ M (второй индекс массива),
% кроме t_0, так как в нём решение задано точно,
% и на разных сетках (первый индекс массива)
p_eff_ForEveryTime = zeros(S,M + 1);

% Вычисление эффективных порядков точности
```

```
for m = 2:(M + 1)

    % Вычисление p_(1)^eff(t_m) и p_(2)^eff(t_m) невозможно
    p_eff_ForEveryTime(1,m) = NaN;
    p_eff_ForEveryTime(2,m) = NaN;

    for s = 3:S
        p_eff_ForEveryTime(s,1) = inf;
        p_eff_ForEveryTime(s,m) = log(...
            sqrt(sum((array_of_u(s - 1,m,:) - ...
            array_of_u(s - 2,m,:)).^2))/...
            sqrt(sum((array_of_u(s,m,:) - ...
            array_of_u(s - 1,m,:)).^2)))/...
            log(r_t);
    end
end

% Отрисовка результатов расчётов для сетки с номером s
S = 9;
figure;
t = t_0:(T - t_0)/M:T; % Определение базовой сетки
% Рисуется зависимость теоретического порядка точности
% от временного узла базовой сетки
plot(t,t*0 + 2,'-*k','MarkerSize',3); hold on;
% Рисуется зависимость эффективного порядка точности
% от временного узла базовой сетки
plot(t(2:M+1),p_eff_ForEveryTime(S,2:M + 1),...
    '-sk','MarkerSize',5,'LineWidth',1);
axis([t(1) t(M + 1) -2.0 3.0]);
xlabel('t'); ylabel('p^{eff}');
```

Отметим, что в узле t_0 базовой сетки T_M эффективный порядок точности $p^{eff}_{t_{(s)}}(t_0)$ не вычисляется, так как в этом узле на любой сетке решение задано точно начальным условием. Поскольку эффективный порядок точности определяет реальное число членов разложения в ряд Тейлора точного решения, которые численная схема передаёт точно, можно положить $p^{eff}_{t_{(s)}}(t_0) = +\infty$.

На рис. 2.4 представлен результат работы данного программного кода для решения задачи (2.1) с набором параметров (2.4) на различных сетках. В качестве базовой сетки $X_N \times T_M$ была

взята сетка с $N = 50$ и $M = 50$ интервалами, и было выполнено последовательное сгущение сеток с коэффициентами сгущения $r_x = r_t = 2$. Представлен результат расчётов после вычисления решения на 9-й сетке ($s = 9$), для которой получен явный выход поточечных значений эффективных порядков точности $p_{t_{(s)}}^{eff}(t_m)$, $0 \leqslant m \leqslant M$, на асимптотически точные значения. На рис. 2.5 для сравнения представлены графики функций $p_{t_{(s)}}^{eff}(t)$ для различных s, которые экспериментальным образом обосновывают выход на асимптотику значений $p_{t_{(9)}}^{eff}(t_m)$, представленных на рис. 2.4.

Таким образом, мы можем сделать следующий вывод о полученном численном решении задачи задачи (2.1) для набора параметров (2.4). После вычисления на $S = 9$ вложенных сетках поточечные значения эффективного порядка точности $p_{t_{(s)}}^{eff}(t)$ для каждого момента времени $t_m \in T_M$ до $m = 20$ включительно сходятся к $p_t{}^{theor} \equiv 2$, а для бóльших значений m, по-видимому, стремятся к $-\infty$. Это означает, что разрушение происходит в момент $T_{bl} \in (t_{20}, t_{21}] \equiv (0.667, 0.700]$, а видимое стремление эффективных порядков точности при $m \geqslant 21$ к $-\infty$ позволяет предположить, что в точке T_{bl} решение имеет особенность типа экспоненциального разрушения.

Если нарушение гладкости решения возникает во всей области по пространственной переменной одновременно, то отклонение сходимости $p_{t_{(s)}}^{eff}(x, t)$ (2.19) от 2 возникает во всех точках сетки X_N с первого временного слоя $t \geqslant T_{bl}$. Если разрушение решения возникает в одной-единственной точке x^*, то описанный метод позволяет проследить во времени процесс разрушения решения в остальных точках. Такая диагностика процесса разрушения решения возможна в связи с тем, что схема CROS1 не приводит к переполению даже в том случае, если решение задачи устремляется к бесконечности [2, 9].

Далее приводится пример MatLab-кода, оформленного в виде отдельного файла BlowUpDiagnostics_for_specified_t.m, который реализует вычисление эффективных порядков точности

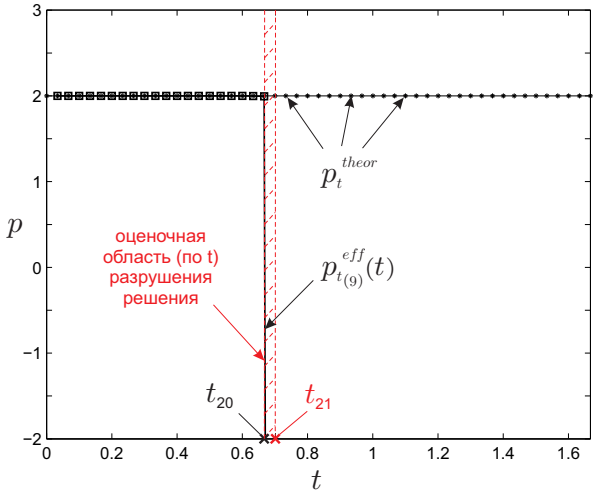

Рис. 2.4: Результат вычисленний эффективных порядков точности $p_{t_{(S)}}^{eff}(t_m)$, $0 \leqslant m \leqslant M$, задачи (2.1) с набором входных данных (2.4) для следующего набора параметров: $N = 50$, $M = 50$, $r_x = 2$, $r_t = 2$, $S = 9$.

приближённого решения в узлах, совпадающих с узлами базовой сетки X_N по пространству, в конкретный момент времени $t_m \in T_M$ (2.19), используя данные из уже сформированного файла data.mat, с целью локализации разрушения по пространственной переменной x.

```
% Загрузка результата вычисления приближённого решения
% на последовательности из S сгущающихся сеток
load('data.mat');

% Выделение памяти под массив значений эффективных
% порядков точности расчёта приближённого решения
% в каждом узле  x_n,  1 ≤ n ≤ N  (второй индекс массива),
% кроме  x_0, так как в нём решение задано точно,
```

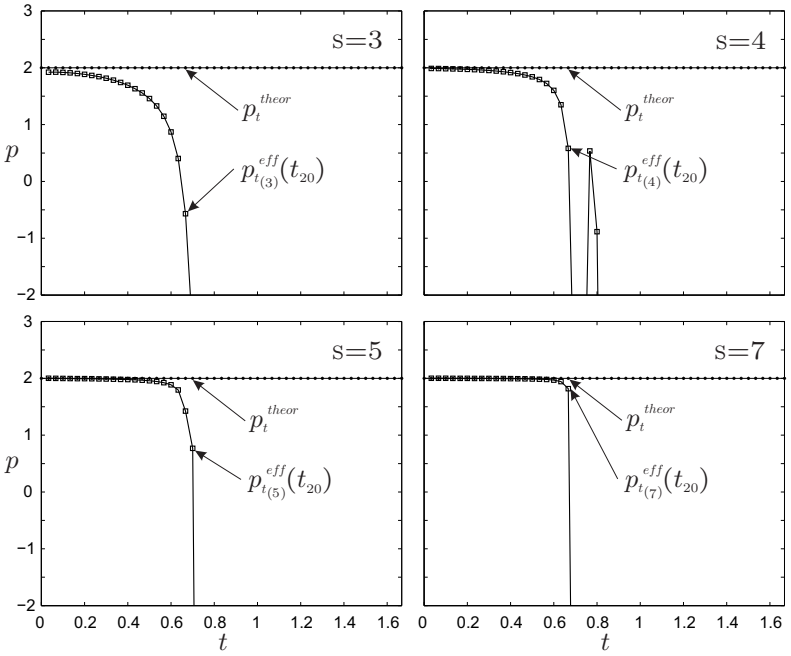

Рис. 2.5: Результат вычисленний эффективных порядков точности $p_{t_{(s)}}^{eff}(t_m)$, $0 \leqslant m \leqslant M$, задачи (2.1) с набором входных данных (2.4) для $N = 50$, $M = 50$, $r_x = 2$, $r_t = 2$ и различных s: $s = \{3, 4, 5, 7\}$.

```
% и на разных сетках (первый индекс массива)
p_eff_ForSpecifiedTime = zeros(S,N + 1);

% Вычисление эффективных порядков точности
% во всех пространственных точках, кроме левой граничной
% на определённом временном слое t_{m-1} с номером (m-1)
% (с учётом сдвига индекса в MatLab'е на +1)
m = 21;
for n = 2:(N + 1)
```

```
18
19      % Вычисление p_(1)^eff(x_n) и p_(2)^eff(x_n) невозможно
20      p_eff_ForSpecifiedTime(1,n) = NaN;
21      p_eff_ForSpecifiedTime(2,n) = NaN;
22
23      for s = 3:S
24          p_eff_ForSpecifiedTime(s,1) = inf;
25          p_eff_ForSpecifiedTime(s,n) = log(...
26            abs(array_of_u(s-1,m,n)-array_of_u(s-2,m,n))/...
27             abs(array_of_u(s,m,n)-array_of_u(s-1,m,n)))/...
28              log(r_t);
29      end
30  end
31
32  % Отрисовка результатов расчётов для сетки с номером s
33  % S = 9;
34  figure;
35  x = a:(b - a)/N:b; % Определение базовой сетки
36  % Рисуется зависимость теоретического порядка точности
37  % от пространственного узла базовой сетки
38  plot(x,x*0 + 2,'-*k','MarkerSize',3); hold on;
39  % Рисуется зависимость эффективного порядка точности
40  % от пространственного узла базовой сетки
41  plot(x(2:M+1),p_eff_ForSpecifiedTime(S,2:N + 1),...
42      '-sk','MarkerSize',5,'LineWidth',1);
43  axis([x(1) x(N + 1) -2.0 3.0]);
44  xlabel('x'); ylabel('p^{eff}');
```

Отметим, что в узле x_0 базовой сетки X_N эффективный порядок точности $p^{eff}_{t_{(s)}}(x_0, t_m)$ не вычисляется, так как в этом узле на любой сетке решение задано точно своим граничным условием. Таким образом, в этой точке эффективный порядок точности можно положить равным $+\infty$.

На рис. 2.6 представлен результат работы данного программного кода для решения задачи (2.1) с набором параметров (2.4) в различные моменты времени до и после разрушения решения. Хорошо видно, что изначально разрушение возникает на правом конце отрезка по пространственной координате, а область разрушения постепенно распространяется влево.

В итоге мы можем сделать вывод о том, какой части числен-

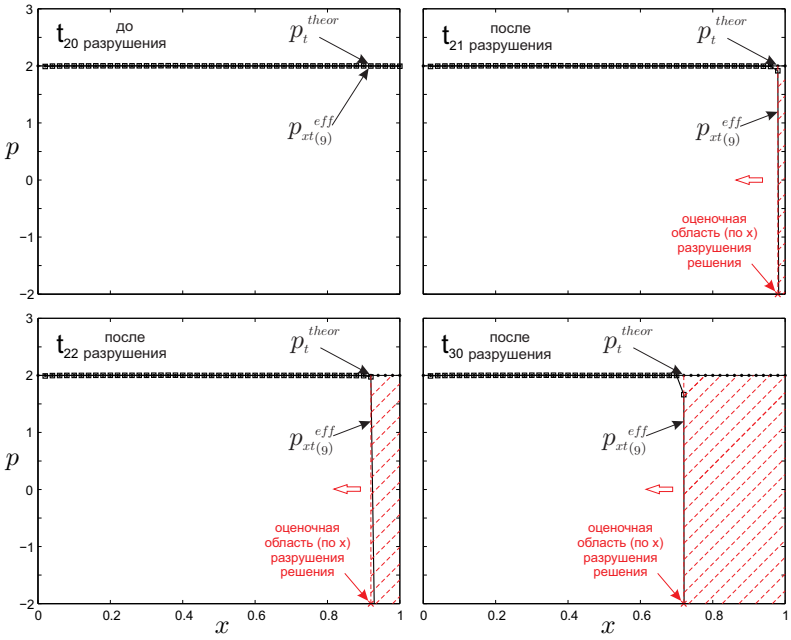

Рис. 2.6: Результат вычисленний эффективных порядков точности $p_{xt}{}^{eff}_{(S)}(x, t_m)$ задачи (2.1) с набором входных данных (2.4) для следующего набора параметров: $N = 50$, $M = 50$, $r_x = 2$, $r_t = 2$, $S = 9$. Представлены графики для $m = \{20, 21, 22, 30\}$.

ного решения (см. рис. 2.7), полученного на 9-й вложенной сетке, мы можем доверять, а какой нет.

Также обратите внимание на отличия решений, изображённых на рисунках 2.2 и 2.7, связанные с тем, что для одного и того же набора входных параметров представлены результаты расчётов для различных s.

Решение на 9-й сетке ($s = 9$) было получено посредством набора следующих команд, оформленных в виде отдельного

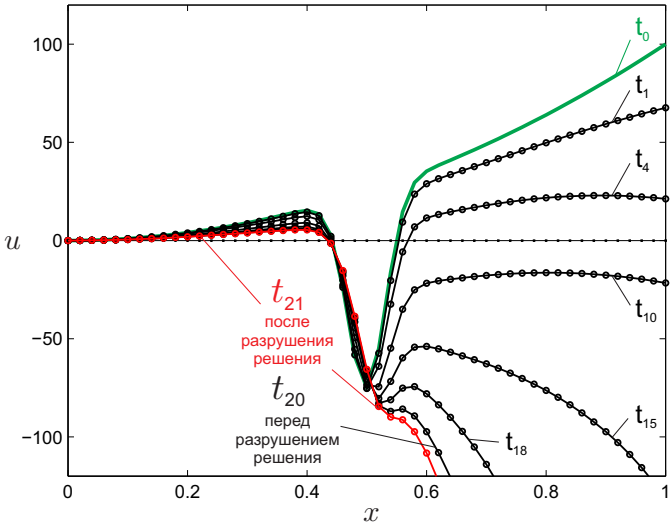

Рис. 2.7: Результат расчёта решения $u_{(s)}(x,t) \equiv u^{(r_x^{s-1}N, r_t^{s-1}M)}(t)$ задачи (2.1) с входными данными (2.4) для следующего набора параметров: $N = 50$, $M = 50$, $r_x = 2$, $r_t = 2$, $s = 9$. Отмечены только узлы, совпадающие с узлами базовой сетки.

MatLab-файла draw.m.

```
% Загрузка результата вычисления приближённого решения
% на последовательности из S сгущающихся в r раз сеток
load('data.mat');

% Из массива решений на разных сетках выбирается
% сеточное решение на 9-й сетке
s = 9;

u(:,:) = array_of_u(s,:,:);

% Отрисовка решения
figure;
```

```
x = a:(b - a)/N:b; % Определение базовой сетки по x
for m = 0:M
    % Рисуется график начального условия
    plot(x,u(1,:),'-g','LineWidth',1); hold on;
    % Рисуется график решения в момент времени t_m
    plot(x,u(m + 1,:),'-ok',...
        'MarkerSize',3,'LineWidth',1); hold on;
    axis([a b -120 120]); xlabel('x'); ylabel('u');
    hold off; drawnow; pause(0.1);
end
```

2.5. Практические задания

Замечание 1. Если в разностной схеме возникает трёхдиагональная матрица стандартной структуры (главная диагональ и по одной диагонали сверху и снизу от неё), можно решать СЛАУ с такой матрицей методом прогонки.

Замечание 2. Если численная оценка времени разрушения значительно меньше времени счёта (скажем, $T_{bl}/T < 1/2$), то разумно повторить расчёты на промежутке меньшей длины, но взять при этом то же количество узлов базовой сетки по переменной t.

Задача 1. Решите численно (по схеме CROS1) начально-краевую задачу

$$\begin{cases} \dfrac{\partial}{\partial t}\left(u_{xx} - u\right) + e^u = 0, \quad x \in (a,b], \quad t \in (t_0,T], \\ u(a,t) = 0, \quad u(b,t) = 0, \quad t \in (t_0,T], \\ u(x,t_0) = u_{init}(x), \quad x \in [a,b], \end{cases} \tag{2.20}$$

для следующего набора параметров:

$$t_0 = 0, \quad a = 0, \quad b = \pi, \quad T = 1, \quad u_{init}(x) = \sin x.$$

Какое значение эффективного порядка точности наблюдается после разрушения? (Примените формулу (2.18).) Проведите соответствующую классификацию особенности. После этого с по-

мощью вычисления эффективного порядка в каждом узле (формула (2.19)) на нескольких первых слоях, где предыдущая формула дала результат, существенно отличный от теоретического, изучите пространственную структуру разрушения.

Задача 2. Решите численно (по схеме CROS1) начально-краевую задачу

$$\begin{cases} \dfrac{\partial}{\partial t}\big(u_{xx} - u\big) + u_{xx} + uu_x + u^3 = 0, \quad x \in (a, b], \quad t \in (t_0, T], \\ u(a, t) = 0, \quad u(b, t) = 0, \quad t \in (t_0, T], \\ u(x, t_0) = u_{init}(x), \quad x \in [a, b], \end{cases} \tag{2.21}$$

для следующего набора параметров:

$$t_0 = 0, \quad a = 0, \quad b = \pi, \quad T = 4, \quad u_{init}(x) = \sin x.$$

Какое значение эффективного порядка точности наблюдается после разрушения? (Примените формулу (2.18).) Проведите соответствующую классификацию особенности. После этого с помощью вычисления эффективного порядка в каждом узле (формула (2.19)) на нескольких первых слоях, где предыдущая формула дала результат, существенно отличный от теоретического, изучите пространственную структуру разрушения.

3| Диагностика разрушения решения начально-краевой задачи для уравнения в частных производных псевдогиперболического типа

В данном разделе рассматриваются особенности численной диагностики разрушения решения начально-краевой задачи для уравнений в частных производных псевдогиперболического типа. В качестве примера будем рассматривать уравнение соболевского типа [12], которое возникает в теории ионно-звуковых волн в плазме. Требуется найти функцию $u(x,t)$, определённую в области $(x,t) \in [a,b] \times [t_0,T]$[1] и удовлетворяющую системе уравнений

$$\begin{cases} \dfrac{\partial^2}{\partial t^2}\left(u_{xx} - e^{\varepsilon u}\right) + u_{xx} = 0, \quad x \in (a,b), \quad t \in (t_0,T], \\ u_x(a,t) = 0, \quad u_x(b,t) = 0, \quad t \in (t_0,T], \\ u(x,t_0) = u_{init_0}(x), \ x \in [a,b], \\ u_t(x,t_0) = u_{init_1}(x), \ x \in [a,b], \end{cases} \tag{3.1}$$

[1] *Замечание.* Мы ставим задачу численного нахождения решения до момента времени T включительно, хотя знаем, что решение в этот момент времени может не существовать и, более того, даже может разрушиться ранее. Это допущение связано с тем, что мы хотим диагностировать разрушение решения численно, а значит, нам необходимо найти *численное* решение вплоть до этого момента времени включительно.

а также диагностировать факт разрушения решения (в случае его наличия) и уточнить его локализацию как во времени, так и в пространстве, по сравнению с априорной оценкой (если таковая имеется), полученной аналитическими методами.

3.1. Поиск численного решения

Прежде всего сведём исходную начально-краевую задачу (3.1) к системе первого порядка по времени, что необходимо для применения эффективных численных методов, описываемых ниже:

$$\begin{cases} \dfrac{\partial}{\partial t}\left(u_{xx} - e^{\varepsilon u}\right) = v, \quad x \in (a,b), \quad t \in (t_0, T], \\ \dfrac{\partial}{\partial t} v + u_{xx} = 0, \quad x \in (a,b), \quad t \in (t_0, T], \\ u_x(a,t) = 0, \quad u_x(b,t) = 0, \quad t \in (t_0, T], \\ u(x,t_0) = u_{init_0}(x), \quad x \in [a,b], \\ v(x,t_0) = u_{init_1}(x)_{xx} - \varepsilon u_{init_1}(x) e^{\varepsilon u_{init_0}(x)}, \quad x \in (a,b). \end{cases} \tag{3.2}$$

Для численного решения задачи (3.2) мы применим метод прямых (MOL) [1, 3, 10], аппроксимируя уравнения в частных производных системами обыкновенных дифференциальных уравнений, итоговая задача для которых может быть эффективно решена с помощью одностадийной схемы Розенброка с комплексным коэффициентом CROS1 [1, 11].

Введём равномерную сетку X_N с N интервалами только по пространственной переменной x с шагом $h = (b-a)/N$ (что соответствует $N+1$ узлу сетки): $X_N = \{x_n,\ 0 \leqslant n \leqslant N :\ x_n = a + nh\}$. Таким образом, после конечно-разностной аппроксимации пространственных производных со вторым порядком точности мы получим дифференциально-алгебраическую систему, из которой требуется определить $N+1$ неизвестную функцию $u_n \equiv u_n(t) \equiv u(x_n, t)$, $n = \overline{0, N}$, и $N-1$ вспомогательную функцию $v_n \equiv v_n(t) \equiv v(x_n, t)$, $n = \overline{1, N-1}$ (v_0 и v_N не входят в

систему):

$$\begin{cases} \dfrac{d}{dt}\Big(\dfrac{u_{n+1}-2u_n+u_{n-1}}{h^2}-e^{\varepsilon u_n}\Big)=v_n,\ n=\overline{1,N-1},\ t\in(t_0,T], \\ \dfrac{d}{dt}\,v_n+\dfrac{u_{n+1}-2u_n+u_{n-1}}{h^2}=0,\quad n=\overline{1,N-1},\quad t\in(t_0,T], \\ \dfrac{-\frac{3}{2}u_0+2u_1-\frac{1}{2}u_2}{h}=0,\ \dfrac{\frac{3}{2}u_N-2u_{N-1}+\frac{1}{2}u_{N-2}}{h}=0,\ t\in(t_0,T], \\ u_n(t_0)=u_{init_0}(x_n),\quad n=\overline{0,N}, \\ v_n(t_0)=\dfrac{u_{init_1}(x_{n+1})-2u_{init_1}(x_n)+u_{init_1}(x_{n-1})}{h^2}- \\ \qquad\qquad -\varepsilon u_{init_1}(x_n)e^{\varepsilon u_{init_0}(x_n)},\quad n=\overline{1,N-1}. \end{cases}$$

Для удобства последующих преобразований перепишем эту систему в следующем виде, уединяя дифференциальную часть в левой части каждого уравнения:

$$\begin{cases} \dfrac{du_{n-1}}{dt}-\big(2+\varepsilon h^2e^{\varepsilon u_n}\big)\dfrac{du_n}{dt}+\dfrac{du_{n+1}}{dt}=h^2v_n,\ n=\overline{1,N-1}, \\ \dfrac{dv_n}{dt}=-\dfrac{u_{n+1}-2u_n+u_{n-1}}{h^2},\quad n=\overline{1,N-1},\quad t\in(t_0,T], \\ u_0=\dfrac{4}{3}u_1-\dfrac{1}{3}u_2,\quad u_N=\dfrac{4}{3}u_{N-1}-\dfrac{1}{3}u_{N-2},\quad t\in(t_0,T], \\ u_n(t_0)=u_{init_0}(x_n),\quad n=\overline{0,N}, \\ v_n(t_0)=\dfrac{u_{init_1}(x_{n+1})-2u_{init_1}(x_n)+u_{init_1}(x_{n-1})}{h^2}- \\ \qquad\qquad -\varepsilon u_{init_1}(x_n)e^{\varepsilon u_{init_0}(x_n)},\quad n=\overline{1,N-1}. \end{cases}$$

Полученная система является дифференциально-алгебраической, поскольку включает в себя как дифференциальные уравнения, так и алгебраические (два уравнения, определяемые граничными условиями). Путём подстановки выражений для u_0 и u_N в дифференциальные уравнения эта

система может быть сведена к чисто дифференциальной:

$$
\begin{cases}
\dfrac{d}{dt}\Big(\dfrac{4}{3}u_1 - \dfrac{1}{3}u_2\Big) - \Big(2 + \varepsilon h^2 e^{\varepsilon u_1}\Big)\dfrac{du_1}{dt} + \dfrac{du_2}{dt} = h^2 v_1,\ t \in (t_0, T], \\
\dfrac{du_{n-1}}{dt} - \Big(2 + \varepsilon h^2 e^{\varepsilon u_n}\Big)\dfrac{du_n}{dt} + \dfrac{du_{n+1}}{dt} = h^2 v_n,\ n = \overline{2, N-2}, \\
\dfrac{du_{N-2}}{dt} - \Big(2 + \varepsilon h^2 e^{\varepsilon u_{N-1}}\Big)\dfrac{du_{N-1}}{dt} + \\
\qquad + \dfrac{d}{dt}\Big(\dfrac{4}{3}u_{N-1} - \dfrac{1}{3}u_{N-2}\Big) = h^2 v_{N-1},\ t \in (t_0, T], \\
\dfrac{dv_1}{dt} = -\dfrac{u_2 - 2u_1 + \big(\frac{4}{3}u_1 - \frac{1}{3}u_2\big)}{h^2}, \quad t \in (t_0, T], \\
\dfrac{dv_n}{dt} = -\dfrac{u_{n+1} - 2u_n + u_{n-1}}{h^2}, \quad n = \overline{2, N-2}, \quad t \in (t_0, T], \\
\dfrac{dv_{N-1}}{dt} = -\dfrac{\big(\frac{4}{3}u_{N-1} - \frac{1}{3}u_{N-2}\big) - 2u_{N-1} + u_{N-2}}{h^2}, \quad t \in (t_0, T], \\
u_n(t_0) = u_{init_0}(x_n), \quad n = \overline{0, N}, \\
v_n(t_0) = \dfrac{u_{init_1}(x_{n+1}) - 2u_{init_1}(x_n) + u_{init_1}(x_{n-1})}{h^2} - \\
\qquad - \varepsilon u_{init_1}(x_n) e^{\varepsilon u_{init_0}(x_n)}, \quad n = \overline{1, N-1},
\end{cases}
$$

Заметим, что теперь система содержит $2N-2$ уравнения и $2N-2$ неизвестных функций u_n и v_n, $n = \overline{1, N-1}$.

Эта система может быть переписана в следующем виде:

$$
\begin{cases}
\boldsymbol{D}(\boldsymbol{y})\dfrac{d\boldsymbol{y}}{dt} = \boldsymbol{f}(\boldsymbol{y}), \quad t \in (t_0, T], \\
\boldsymbol{y}(t_0) = \boldsymbol{y}_{init},
\end{cases}
\tag{3.3}
$$

где $\boldsymbol{y} = \big(u_1\ u_2\ \ldots\ u_{N-1}\ v_1\ v_2\ \ldots\ v_{N-1}\big)^T$, $\boldsymbol{f} = \big(f_1\ f_2\ \ldots\ f_{2N-2}\big)^T$ и $\boldsymbol{y}_{init} = \big(u_1(t_0)\ u_2(t_0)\ \ldots\ u_{N-1}(t_0)\ v_1(t_0)\ v_2(t_0)\ \ldots\ v_{N-1}(t_0)\big)^T$.

Здесь вектор-функция $\boldsymbol{f}$ имеет следующую структуру:

$$f_n = \begin{cases} h^2 y(N-1+n), & \text{если } n = \overline{1, N-1}, \\ -\dfrac{y_2 - 2y_1 + \left(\frac{4}{3}y_1 - \frac{1}{3}y_2\right)}{h^2}, & \text{если } n = N, \\ -\dfrac{y_{n-N+2} - 2y_{n-N+1} + y_{n-N}}{h^2}, & \text{если } n = \overline{N+1, 2N-3}, \\ -\dfrac{\left(\frac{4}{3}y_{N-1} - \frac{1}{3}y_{N-2}\right) - 2y_{N-1} + y_{N-2}}{h^2}, & \text{если } n = 2N-2. \end{cases}$$

Ниже приводится пример MatLab-функции, которая реализует вычисление компонент вектор-функции $\boldsymbol{f}$.

```
function f = f(y,h,N)

    % Функция вычисляет вектор правой части
    % решаемой системы ОДУ

    % Входные данные:
    % y - вектор решения системы ОДУ
    % на текущем временном слое
    % h - шаг сетки по переменной x
    % N - число интервалов сетки по переменной x

    % Выходные данные:
    % f - искомый вектор f

    f = zeros(2*N - 2,1);

    for n = 1:(N - 1)
        f(n) = h^2*y(N - 1 + n);
    end
    f(N) = -(y(2) - 2*y(1) + (4/3*y(1) - 1/3*y(2)))/h^2;
    for n = (N + 1):(2*N - 3)
        f(n) = -(y(n - N + 2) - 2*y(n - N + 1) + ...
            y(n - N))/h^2;
    end
    f(2*N - 2) =  -((4/3*y(N - 1) - 1/3*y(N - 2)) - ...
        2*y(N - 1) + y(N - 2))/h^2;

end
```

А матричная функция $\boldsymbol{D}$ имеет следующие ненулевые элементы:

$$
\begin{aligned}
D_{n,n-1} &= \begin{cases} 1, & \text{если } n = \overline{2, N-2}, \\ 1 - \frac{1}{3}, & \text{если } n = N-1, \end{cases} \\
D_{n,n} &= \begin{cases} \frac{4}{3} - \left(2 + \varepsilon h^2 e^{\varepsilon y_1}\right), & \text{если } n = 1, \\ -\left(2 + \varepsilon h^2 e^{\varepsilon y_n}\right), & \text{если } n = \overline{2, N-2}, \\ -\left(2 + \varepsilon h^2 e^{\varepsilon y_{N-1}}\right) + \frac{4}{3}, & \text{если } n = N-1, \\ 1, & \text{если } n = \overline{N, 2N-2}, \end{cases} \\
D_{n,n+1} &= \begin{cases} -\frac{1}{3} + 1, & \text{если } n = 1, \\ 1, & \text{если } n = \overline{2, N-2}. \end{cases}
\end{aligned}
$$

Ниже приводится пример MatLab-функции, которая реализует вычисление компонент матричной функции $\boldsymbol{D}$.

```
function D = D(eps,y,h,N)

    % Функция вычисляет матрицу дифференциального
    % оператора решаемой системы ОДУ

    % Входные данные:
    % eps - малый параметр
    % y - вектор решения системы ОДУ
    % на текущем временном слое
    % h - шаг сетки по переменной x
    % N - число интервалов сетки по переменной x

    % Выходные данные:
    % D - искомая матрица дифференциального оператора

    D = zeros(2*N - 2,2*N - 2);

    D(1,1) = 4/3 - (2 + eps*h^2*exp(eps*y(1)));
    D(1,2) = -1/3 + 1;
    for n = 2:(N - 2)
        D(n,n - 1) = 1;
        D(n,n) = - (2 + eps*h^2*exp(eps*y(n)));
```

```
23          D(n,n + 1) = 1;
24      end
25      D(N - 1,N - 2) = 1 - 1/3;
26      D(N - 1,N - 1) = -(2 + eps*h^2*exp(eps*y(N-1))) +4/3;
27      for n = N:(2*N - 2)
28          D(n,n) = 1;
29      end
30
31  end
```

Замечание. Важно отметить, что схема Розенброка с комплексным коэффициентом CROS1, которая будет применена ниже, будет иметь порядок точности $O(\tau^2)$ только в случае постоянной матричной функции $\boldsymbol{D}$ (подробные объяснения приведены в статьях [2, 9]). Неявная система ОДУ, подобная (3.3), с постоянной матрицей может быть получена, если мы введём в (3.2) ещё одну вспомогательную переменную $g = u_{xx} - e^{\varepsilon u}$. Данный вариант решения исходной задачи (3.1) мы рассмотрим позже (в конце данной главы). В нашем же случае (случай непостоянной матричной функции $\boldsymbol{D}(\boldsymbol{y})$) схема CROS1 будет иметь порядок точности $O(\tau^1)$.

Далее введём равномерную сетку T_M по времени t с шагом $\tau = (T - t_0)/M$, которая имеет M интервалов (то есть $M + 1$ узел): $T_M = \{t_m,\ 0 \leqslant m \leqslant M :\ t_m = t_0 + m\tau\}$.

В результате мы можем применить схему Розенброка CROS1 для решения системы (3.3):

$$\begin{gathered}\boldsymbol{y}(t_{m+1}) = \boldsymbol{y}(t_m) + (t_{m+1} - t_m)\,\mathrm{Re}\,\boldsymbol{w}_1\,,\\ \text{где } \boldsymbol{w}_1 \text{ является решением СЛАУ}\\ \left[\boldsymbol{D}\Big(\boldsymbol{y}(t_m)\Big) - \frac{1+i}{2}(t_{m+1} - t_m)\,\boldsymbol{f}_{\boldsymbol{y}}\Big(\boldsymbol{y}(t_m)\Big)\right]\boldsymbol{w}_1 = \boldsymbol{f}\Big(\boldsymbol{y}(t_m)\Big).\end{gathered} \tag{3.4}$$

Здесь $\boldsymbol{f}_{\boldsymbol{y}}$ — матрица с элементами $(f_y)_{n,m} \equiv \frac{\partial f_n}{\partial y_m}$ (матрица Якоби), которая для рассматриваемой системы имеет следующие ненулевые элементы:

$$(f_y)_{n,N-1+n} = h^2, \quad \text{если } n = \overline{1, N-1},$$

$$(f_y)_{n,n-N} = \begin{cases} -\frac{1}{h^2}, & \text{если } n = \overline{N+1, 2N-3}, \\ \frac{1/3-1}{h^2}, & \text{если } n = 2N-2. \end{cases}$$

$$(f_y)_{n,n-N+1} = \begin{cases} \frac{2-4/3}{h^2}, & \text{если } n = N, \\ \frac{2}{h^2}, & \text{если } n = \overline{N+1, 2N-3}, \\ \frac{-4/3+2}{h^2}, & \text{если } n = 2N-2, \end{cases}$$

$$(f_y)_{n,n-N+2} = \begin{cases} \frac{-1+1/3}{h^2}, & \text{если } n = N, \\ -\frac{1}{h^2}, & \text{если } n = \overline{N+1, 2N-3}. \end{cases}$$

Ниже приводится пример MatLab-функции, которая реализует вычисление компонент матрицы Якоби $\boldsymbol{f_y}$.

```
function f_y = f_y(y,h,N)

    % Функция вычисляет матрицу Якоби правой части
    % решаемой системы ОДУ

    % Входные данные:
    % y - вектор решения системы ОДУ
    % на текущем временном слое
    % h - шаг сетки по переменной x
    % N - число интервалов сетки по переменной x

    % Выходные данные:
    % f_y - искомая матрица Якоби

    f_y = zeros(2*N - 2,2*N - 2);

    for n = 1:(N-1)
        f_y(n,N - 1 + n) = h^2;
    end
    f_y(N,1) = (2 - 4/3)/h^2;
    f_y(N,2) = (-1 + 1/3)/h^2;
    for n = (N + 1):(2*N - 3)
        f_y(n,n - N) = -1/h^2;
        f_y(n,n - N + 1) = 2/h^2;
```

```
25          f_y(n,n - N + 2) = -1/h^2;
26      end
27      f_y(2*N - 2,N - 2) = (1/3 - 1)/h^2;
28      f_y(2*N - 2,N - 1) = (-4/3 + 2)/h^2;
29
30  end
```

Далее приводится пример MatLab-функции, которая реализует поиск численного решения задачи (3.1) в видоизменённой форме (3.2)–(3.3) по схеме (3.4), используя приведённые выше функции f, D и f_y.

```
1   function u = PDESolving(eps,a,b,N_0,t_0,T,M_0, ...
2       u_init_0,u_init_1,s,r_x,r_t)
3
4       % Функция находит приближённое численное решение
5       % уравнения в частных производных (УрЧП/PDE)
6
7       % Входные параметры:
8       % eps - малый параметр
9       % a,b - границы области [a,b] по переменной x
10      % N_0 - число интервалов базовой сетки по пространству
11      % t_0, T - начальный и конечный моменты счёта t_0 и T
12      % M_0 - число интервалов базовой сетки по времени
13      % u_init_0 и u_init_1 - функции,
14      % определяющие начальные условия
15      % s - номер сетки, на которой вычисляется решение
16      % (если s = 1, то решение ищется на базовой сетке)
17      % r_x и r_t - коэффициенты сгущения сетки по x и по t
18
19      % Выходной параметр:
20      % u- массив, содержащий сеточные значения решения УрЧП
21      % только в узлах, совпадающих с узлами базовой сетки
22
23      % Формирование сгущённой
24      % в r_x^(s - 1) раз по пространственной переменной x и
25      % в r_t^(s - 1) раз по временной переменной t
26      % сетки с номером s
27
28      N = N_0*r_x^(s - 1);     % Вычисление числа интервалов
29      M = M_0*r_t^(s - 1);     % на сетке с номером s
30
```

```
31      h = (b - a)/N;      % Определение шага сетки по x
32      x = a:h:b;          % Определение сгущённой сетки по x
33      tau = (T - t_0)/M; % Определение шага сетки по t
34      t = t_0:tau:T;      % Определение сгущённой сетки по t
35
36      % Выделение памяти под массив u
37      % В строке с номером (m + 1) этого массива хранятся
38      % сеточные значения решения, соответствующие
39      % моменту времени t_m базовой сетки по времени
40      u = zeros(M_0 + 1,N_0 + 1);
41
42      % Выделение памяти под массив сеточных значений
43      % решения системы ОДУ, соответствующего
44      % текущему моменту времени t_m
45      y = zeros(1,2*N - 2);
46
47      % Задание начального условия решаемой системы ОДУ
48      for n = 1:(N - 1)
49          y(1,n) = u_init_0(x(n + 1));
50          y(1,N - 1 + n) = (u_init_1(x(n + 2)) - ...
51              2*u_init_1(x(n +1)) + u_init_1(x(n)))/h^2 - ...
52              eps*u_init_1(x(n + 1))*...
53              exp(eps*u_init_0(x(n + 1)));
54      end
55
56      % Из первой строки массива u,
57      % соответствующего начальному условию,
58      % выбираются сеточные значения из узлов,
59      % совпадающих с узлами базовой сетки по пространству
60      for n = 1:(N_0 + 1)
61          u(1,n) = u_init_0(x((n - 1)*r_x^(s - 1) + 1));
62      end
63
64      % Введение индекса, отвечающего за выбор
65      % временного слоя на сетке с номером s,
66      % совпадающего с соответствующим
67      % временным слоем базовой сетки. На данный момент
68      % будем отслеживать совпадение t_m на сгущённой сетке
69      % с t_{m_basic} на базовой сетки
70      m_basic = 2;
71
72      for m = 1:M
73
```

```
74          % Реализация схемы CROS1
75
76          w_1 = (D(eps,y,h,N) - (1+1i)/2*(t(m + 1) - ...
77              t(m))*f_y(y,h,N))\f(y,h,N);
78
79          y = y + (t(m + 1) - t(m))*real(w_1)';
80
81          % Выполнение проверки совпадения  t_{m+1}
82          % на сгущённой сетке с  t_{m_basic}  базовой сетки
83          if (m + 1) == (m_basic - 1)*r_t^(s - 1) + 1
84
85              % Заполнение массива сеточных значений решения
86              % исходной задачи для УрЧП
87
88              % Учитываются левое и правое граничные условия
89              u(m_basic,1) = 4/3*y(1) - 1/3*y(2);
90              u(m_basic,N_0 + 1) = 4/3*y(N - 1) - ...
91                  1/3*y(N - 2);
92
93              % На текущем временном слое выбираются
94              % пространственные узлы, совпадающие
95              % с узлами базовой сетки
96              % (кроме граничных, которые уже учтены выше)
97              for n = 2:N_0
98                  u(m_basic,n) = y((n - 1)*r_x^(s - 1));
99              end
100
101             % Теперь будет отслеживаться совпадение  t_{m+1}
102             % на сгущённой сетке с очередным  t_{m_basic}
103             % базовой сетки
104             m_basic = m_basic + 1;
105
106         end
107
108     end
109
110 end
```

Замечание. Отметим некоторые особенности функции PDESolving.

1. В функции уже реализована возможность поиска приближённого численного решения на последовательности сгу-

щающихся сеток, включая выбор сеточных значений только из узлов, совпадающих с узлами базовой сетки. Эта особенность нам потребуется при реализации численной диагностики разрушения решения, которая рассматривается в следующем разделе. Сейчас же мы будем использовать эту функцию для вычисления решения только на одной (базовой) сетке. Этот случай соответствует значению входного параметра $s := 1$, поэтому значения параметров r_x и r_t не существенны и ни на что на данный момент не влияют.

2. С целью экономии памяти (что критично при больших значениях s) в памяти хранится только набор сеточных значений вектора $\boldsymbol{y}(t_m)$ в текущий расчётный момент времени, а в качестве сеточного решения на сетке с номером s функция возвращает не полный набор сеточных значений, а только набор значений в узлах, совпадающих с узлами базовой сетки.

3. Необходимо обратить внимание на то, что при обращении к компонентам вектора x все индексы сдвинуты на $+1$ (по сравнению с аналитическими формулами выше), так как в MatLab'е нумерация элементов массивов начинается с 1 (поэтому $x_0 \equiv x(1)$, $x_1 \equiv x(2)$, $\ldots$, $x_N \equiv x(N+1)$).

Запуск функции PDESolving может быть выполнен с помощью, например, следующего набора команд:

```
% Определение начального и конечного времени счёта
t_0 = 0; T = 5;

% Определение границ отрезка  x ∈ [a,b]
a = 0; b = pi;

% Определение числа интервалов базовой сетки
N = 50; M = 50;

% Определение параметра задачи
epsilon = 0.1;
```

```
12
13 % Определение начальных условий
14 u_init_0 = @(x) 0;
15 u_init_1 = @(x) -(x*(pi-x))^2*sin(x/3);
16
17 s = 1;   % Номер сетки (только базовая)
18 r_x = 2; % Коэффициент сгущения сетки по x
19 r_t = 4; % Коэффициент сгущения сетки по t
20
21 u = PDESolving(epsilon,a,b,N,t_0,T,M, ...
22     u_init_0,u_init_1,s,r_x,r_t);
23
24 % Отрисовка решения
25 figure;
26 x = a:(b - a)/N:b; % Определение базовой сетки по x
27 for m = 0:M
28     % Рисуется график начального условия
29     plot(x,u(1,:),'--k','LineWidth',1); hold on;
30     % Рисуется график решения в момент времени t_m
31     plot(x,u(m + 1,:),'-ok',...
32         'MarkerSize',3,'LineWidth',1); hold on;
33     axis([a b -20.5 0.01]); xlabel('x'); ylabel('u');
34     hold off; drawnow; pause(0.1);
35 end
```

Данный набор команд позволит получить решение для следующего набора параметров задачи (3.1):

$$\begin{aligned} &\varepsilon = 0.1, \quad a = 0, \quad b = \pi, \quad t_0 = 0, \quad T = 5, \\ &u_{init_0}(x) = 0, \quad u_{init_1}(x) = -\big(x(\pi - x)\big)^2 \sin\frac{x}{3}, \end{aligned} \tag{3.5}$$

с параметрами сеток по пространству и времени:

$$N = 50, \quad M = 50. \tag{3.6}$$

На рис. 3.1 приведено несколько наборов сеточных значений функций $u(x, t_m)$ для отдельных моментов времени t_m.

3.2. Оптимизация численного счёта

Для численной диагностики факта разрушения решения нам будет необходимо проводить вычисления на последовательности

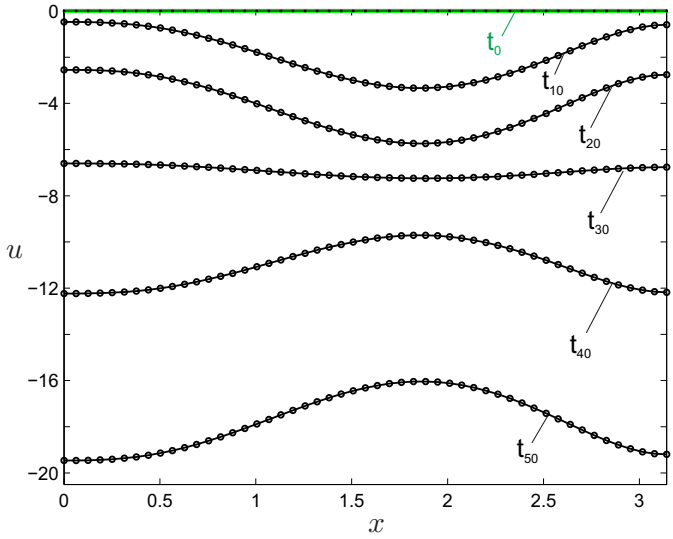

Рис. 3.1: Пример решения задачи (3.1) для набора параметров (3.5)–(3.6) по схеме (3.4). На рисунке отображено несколько наборов сеточных значений функций $u(x, t_m)$ для отдельных моментов времени t_m.

сгущающихся сеток (сгущение каждый раз производится в r_x раз по пространственной переменной и в r_t раз — по временной). С увеличением номера s сетки из последовательности ($s = \overline{1, S}$) размеры сетки $X_{r_x^{s-1}N} \times T_{r_t^{s-1}M}$ быстро растут, что приводит к значительному увеличению времени работы программы, а иногда и к нехватке компьютерной памяти. Существенное увеличение времени счёта в первую очередь связано с тем, что при решении СЛАУ (3.4) общим методом Гаусса необходимо совершить порядка $O(N^3)$ операций, где N — размерность решаемой системы. Поскольку решать СЛАУ приходится при каждом переходе на следующий временной слой, итоговое время работы програм-

мы пропорционально кубу размерности сетки по пространству и первой степени размерности сетки по времени. Однако матрица системы (3.4) имеет специальный вид и состоит из четырёх блоков размерности $(N-1)\times(N-1)$, внутренняя структура которых схематично изображена на рис. 3.2 (ненулевые элементы расположены только на отмеченных диагоналях). Это позволяет разработать алгоритм решения СЛАУ такого специального вида за $O(N^1)$ операций, который является гораздо более экономичной реализацией метода Гаусса для решения СЛАУ с матрицами специального вида как в плане времени выполнения вычислений (трудоёмкость равна $O(N^1)$ операций), так и в плане требуемой для работы алгоритма памяти (что критично при вычислениях на очень густых сетках в случае большого значения S). Главное, что итоговое время работы программы будет пропорционально первой степени размерности сетки по пространству и первой степени размерности сетки по времени.

3.2.1. Использование вырожденного метода Гаусса

Пусть имеется СЛАУ вида $AX = B$ с матрицей A специального вида, изображённая на рис. 3.2. Для простоты будем считать, что размеры каждого блока равны $N \times N$. Будем считать, что матрица системы A хранится в памяти не в виде двумерного массива (что требует хранения в памяти $4\times N\times N$ элементов), а в виде восьми массивов вида $\mathrm{diag}_{(i,j)_{m/d/u}}$ размеров $1\times N$, которые содержат ненулевые элементы матрицы, расположенные на соответствующих диагоналях (что требует хранения $8\times 1\times N$ элементов). Здесь пара индексов (i,j) отвечает за координаты блока, в котором находится соответствующая диагональ, а последний индекс принимает значения m≡«main»≡«основная», либо d≡«down»≡«нижняя», либо u≡«up»≡«верхняя» и обозначает позицию диагонали или кодиагонали в соответствующем блоке.

Замечание. Из рис. 3.2 видно, что диагонали имеют разные длины. Однако для удобства программной реализации алгоритма мы будем хранить их значения в массивах одинаковой длины

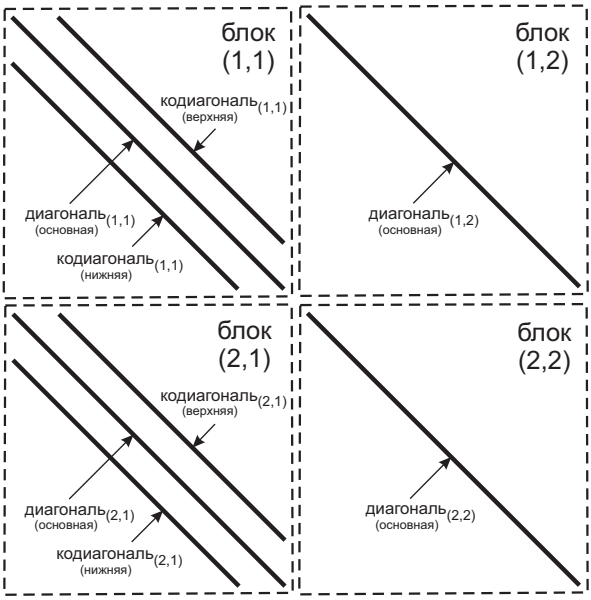

Рис. 3.2: Структура матрицы СЛАУ (3.4).

N. При этом элементы нижних диагоналей будем записывать в соответствующие массивы начиная со 2-го элемента массива (при этом элементы массивов $\mathrm{diag}_{(1,1)_d}(1)$ и $\mathrm{diag}_{(2,1)_d}(1)$ могут принимать любые значения, так как не будут использоваться в алгоритме), а элементы верхних диагоналей будут записываться в соответствующие массивы начиная с 1-го элемента массива (элементы массивов $\mathrm{diag}_{(1,1)_u}(N)$ и $\mathrm{diag}_{(2,1)_u}(N)$ могут принимать любые значения, так как не будут использоваться в алгоритме).

Один из возможных (но при этом эквивалентный многим другим вариантам реализации) алгоритмов решения такой системы можно сформулировать следующим образом:

1. Вычитанием строк расширенной матрицы $A|B$ с номерами $\overline{N+1, 2N}$ из соответствующих строк с номерами, на N меньшими, добиваемся обнуления диагонали, находящейся в блоке (1,2). В результате получаем систему $\tilde{A}X = \tilde{B}$, первые N уравнений которой образуют СЛАУ с трёхдиагональной матрицей только относительно первых N компонент вектора X.

2. Находим первые N компонент вектора X с помощью метода прогонки (подробное описание смотрите, например, в пункте 2.1.4 главы 2 учебника [4]), используя в качестве входной информации метода три вектора, содержащие диагональные элементы блока (1,1) матрицы $\tilde{A}$, и вектор, содержащий первые N компонент вектора $\tilde{B}$.

3. Строки с номерами $n = \overline{N+1, 2N}$ расширенной матрицы $\tilde{A}|\tilde{B}$ будут соответствовать линейным алгебраическим уравнениям, каждое из которых после подстановки найденных в предыдущем пункте первых N компонент вектора X будут содержать только одну неизвестную — n-ю компоненту вектора X, которая может быть найдена с помощью простых алгебраических преобразований.

Ниже приводится пример MatLab-функции, которая реализует описанный алгоритм решения СЛАУ $AX = B$ с матрицей специального вида (рис. 3.2), используя в качестве входных данных только 9 одномерных массивов длины N: первые 8 содержат ненулевые элементы матрицы A, а девятый содержит вектор правой части.

```
function X = SpecialMatrixAlgorithm...
    (diag_11_m,diag_11_d,diag_11_u,diag_12_m,...
    diag_21_m,diag_21_d,diag_21_u,diag_22_m,B)

    % Функция реализует решение СЛАУ AX = B
    % с матрицей специального вида, которая состоит
    % из четырёх квадратных блоков размерности (NxN)
```

```
8       % Квадраты (1,1) и (2,1) - трёхдиагональные матрицы
9       % Квадраты (1,2) и (2,2) - диагональные матрицы
10
11      % Входные параметры:
12      % diag_11_m,diag_11_d,diag_11_u,diag_12_m,
13      % diag_21_m,diag_21_d,diag_21_u,diag_22_m -
14      % - массивы длины N, содержащие ненулевые элементы
15      % матрицы A, расположенные на диагоналях
16      % (элементы массивов diag_11_d(1), diag_21_d(1),
17      % diag_11_u(N) и diag_21_u(N) не используются)
18      % B - вектор правой части длины 2N
19
20      N = length(B)/2;
21      X = zeros(2*N,1);
22
23      % Обнуляется диагональ в квадрате (1,2)
24      % При этом переопределяются диагональные элементы
25      % в блоке (1,1) и первые N элементов вектора B
26      for n = N:-1:1
27          c = diag_12_m(n)/diag_22_m(n);
28          diag_11_d(n) = diag_11_d(n) - c*diag_21_d(n);
29          diag_11_m(n) = diag_11_m(n) - c*diag_21_m(n);
30          diag_11_u(n) = diag_11_u(n) - c*diag_21_u(n);
31          B(n) = B(n) - c*B(n + N);
32      end
33
34      % С помощью метода прогонки ищутся
35      % первые N элементов вектора X
36      X(1:N) = TridiagonalMatrixAlgorithm...
37          (diag_11_m,diag_11_d,diag_11_u,B(1:N));
38
39      % Вычисляются оставшиеся элементы вектора X
40      % с индексами (N + 1):2*N
41      X(N + 1) = (B(N + 1) - diag_21_m(1)*X(1) - ...
42          diag_21_u(1)*X(2))/diag_22_m(1);
43      for n = 2:(N - 1)
44          X(N + n) = (B(N + n) - diag_21_d(n)*X(n - 1) ...
45              -diag_21_m(n)*X(n) -diag_21_u(n)*X(n+1))/...
46              diag_22_m(n);
47      end
48      X(2*N) = (B(2*N) - diag_21_d(N)*X(N - 1) - ...
49          diag_21_m(N)*X(N))/diag_22_m(N);
50
```

```
51 end
```

Здесь используется функция TridiagonalMatrixAlgorithm, которая реализует метод прогонки (подробное описание смотрите, например, в пункте 2.1.4 главы 2 учебника [4]) для решения СЛАУ $AX = B$ (размерности N) с трёхдиагональной матрицей, используя в качестве входных данных не матрицу системы A (что требует хранения в памяти $N \times N$ элементов), а только три вектора a, b и c (в этом случае в памяти будут хранится только $3 \times n \times 1$ элементов), которые содержат ненулевые элементы матрицы, расположенные на соответствующих диагоналях.

$$\begin{pmatrix} a(1) & c(1) & & & \\ b(2) & a(2) & c(2) & & \\ & b(3) & a(3) & c(3) & \\ & & \ddots & \ddots & \ddots \\ & & & b(n) & a(n) \end{pmatrix} \begin{pmatrix} X(1) \\ X(2) \\ X(3) \\ \vdots \\ X(n) \end{pmatrix} = \begin{pmatrix} B(1) \\ B(2) \\ B(3) \\ \vdots \\ B(n) \end{pmatrix}$$

```
1  function [X] = TridiagonalMatrixAlgorithm(a,b,c,B)
2
3  % Функция реализует метод прогонки (алгоритм Томаса)
4  % для решения СЛАУ A X = B с трёхдиагональной матрицей
5
6  % Входные параметры:
7  % B - вектор правой части длины n (столбец или строка)
8  % a, b, c - вектора длины n, содержащие элементы
9  % диагоналей (b(1) и c(n) не используются)
10
11 % [ a(1) c(1)                          ] [ X(1) ]   [ B(1) ]
12 % [ b(2) a(2) c(2)                     ] [ X(2) ]   [ B(2) ]
13 % [      b(3) a(3) c(3)               ] [      ]   [      ]
14 % [           ...  ...  ...           ] [ ...  ] = [ ...  ]
15 % [                ...  ...  c(n-1)]  [X(n-1)]   [B(n-1)]
16 % [                     b(n)  a(n) ]  [ X(n) ]   [ B(n) ]
17
18     n = length(B);
19     v = zeros(n,1);
20     X = zeros(n,1);
```

```
21
22      w = a(1);
23      X(1) = B(1)/w;
24      for i = 2:n
25          v(i - 1) = c(i - 1)/w;
26          w = a(i) - b(i)*v(i - 1);
27          X(i) = (B(i) - b(i)*X(i - 1))/w;
28      end
29      for j = n-1:-1:1
30          X(j) = X(j) - v(j)*X(j + 1);
31      end
32
33  end
```

Таким образом, для применения разработанного вырожденного метода Гаусса для решения СЛАУ (3.4) необходимо сначала подготовить массивы, которые будут содержать ненулевые элементы, расположенные на соответствующих диагоналях матрицы

$$\left[\boldsymbol{D}\Big(\boldsymbol{y}(t_m)\Big) - \frac{1+i}{2}\,(t_{m+1}-t_m)\,\boldsymbol{f}_{\boldsymbol{y}}\Big(\boldsymbol{y}(t_m)\Big)\right].$$

Ниже приводится пример соответствующей MatLab-функции, которая по сути является небольшой модификацией функций D и f_y, выписанных ранее.

```
1   function [diag_11_m,diag_11_d,diag_11_u,diag_12_m, ...
2       diag_21_m,diag_21_d,diag_21_u,diag_22_m] = ...
3       DiagonalsPreparation(eps,y,tau,h,N)
4
5       % Функция подготавливает массивы,
6       % которые содержат диагональные элементы
7       % блоков матрицы решаемой системы ОДУ
8
9       % Данная матрица имеет вид
10      % [D - (1+1i)/2*tau*f_u] и является блочной и состоит
11      % из четырёх квадратных блоков размерности (N-1)x(N-1)
12      % Квадраты (1,1) и (2,1) - трёхдиагональные матрицы
13      % Квадраты (1,2) и (2,2) - диагональные матрицы
14
15      % Входные данные:
```

```
16      % eps - малый параметр
17      % y - вектор решения системы ОДУ
18      % на текущем временном слое
19      % tau - текущий шаг по времени
20      % h - шаг сетки по переменной x
21      % N - число интервалов сетки по переменной x
22
23      % Выходные параметры:
24      % diag_11_m,diag_11_d,diag_11_u,diag_12_m,diag_21_m,
25      % diag_21_d,diag_21_u,diag_22_m - искомые массивы
26
27      % Выделяем память под искомые массивы
28      diag_11_m = zeros(1,N-1); diag_11_d = zeros(1,N-1);
29      diag_11_u = zeros(1,N-1); diag_12_m = zeros(1,N-1);
30      diag_21_m = zeros(1,N-1); diag_21_d = zeros(1,N-1);
31      diag_21_u = zeros(1,N-1); diag_22_m = zeros(1,N-1);
32
33      for n = 2:(N - 2)
34          diag_11_d(n) = 1;
35      end
36      diag_11_d(N - 1) = 1 - 1/3;
37
38      diag_11_m(1) = 4/3 -(2 + eps*h^2*exp(eps*y(1)));
39      for n = 2:(N - 2)
40          diag_11_m(n) = -(2 + eps*h^2*exp(eps*y(n)));
41      end
42      diag_11_m(N-1) = -(2 + eps*h^2*exp(eps*y(N-1))) +4/3;
43
44      diag_11_u(1) = -1/3 + 1;
45      for n = 2:(N - 2)
46          diag_11_u(n) = 1;
47      end
48
49      for n = 1:(N - 1)
50          diag_12_m(n) = -(1+1i)/2*tau*(h^2);
51      end
52
53      for n = (N + 1):(2*N - 3)
54          diag_21_d(n - N + 1) = -(1+1i)/2*tau*(-1/h^2);
55      end
56      diag_21_d(N - 1) = -(1+1i)/2*tau*((1/3 - 1)/h^2);
57
58      diag_21_m(1) = -(1+1i)/2*tau*((2 - 4/3)/h^2);
```

```
59      for n = (N + 1):(2*N - 3)
60          diag_21_m(n - N + 1) = -(1+1i)/2*tau*(2/h^2);
61      end
62      diag_21_m(N - 1) = -(1+1i)/2*tau*((-4/3 + 2)/h^2);
63
64      diag_21_u(1) = -(1+1i)/2*tau*((-1 + 1/3)/h^2);
65      for n = (N + 1):(2*N - 3)
66          diag_21_u(n - N + 1) = -(1+1i)/2*tau*(-1/h^2);
67      end
68
69      for n = N:(2*N - 2)
70          diag_22_m(n - N + 1) = 1;
71      end
72
73  end
```

В результате для использования этих функций в функции PDESolving необходимо заменить кусок кода, находящийся в строках 74–79, на набор следующих команд (при этом функции D и f_y больше не будут использоваться).

```
1           % Реализация схемы CROS1
2
3           % Подготовка массивов, которые содержат
4           % элементы диагоналей блочной матрицы
5           % [D(y) - (1+1i)/2*tau*f_u(y)]
6           [diag_11_m,diag_11_d,diag_11_u,diag_12_m,...
7               diag_21_m,diag_21_d,diag_21_u,diag_22_m] = ...
8               DiagonalsPreparation...
9               (eps,y,t(m + 1) - t(m),h,N);
10
11          % w_1 ищется с помощью вырожденного метода Гаусса
12          w_1 = SpecialMatrixAlgorithm(...
13              diag_11_m,diag_11_d,diag_11_u,diag_12_m,...
14              diag_21_m,diag_21_d,diag_21_u,diag_22_m,...
15              f(y,h,N));
16
17          y = y + (t(m + 1) - t(m))*real(w_1)';
```

Сравним количество операций, которые необходимо произвести при переходе на следующий временной слой при поиске приближённого решения задачи (3.2) в двух случаях: исполь-

зование общего метода Гаусса или применение вышеописанного модифицированного алгоритма.

Как уже упоминалось ранее, в случае общего метода Гаусса время работы алгоритма пропорционально кубу размерности матрицы системы (3.4). Т. е. в нашем случае это время пропорционально

$$\sim \frac{2}{3}\big(2(N-1)\big)^3 \sim \frac{2}{3}8N^3 = \frac{16}{3}N^3 \sim 5N^3 \sim O(N^3).$$

В случае предложенной реализации вырожденного метода Гаусса при выполнении 1-го пункта алгоритма совершается $9(N-1)$ операций, при реализации 2-го пункта $8(N-1)-7$ операций, при реализации 3-го пункта $5 \cdot 2 + 7(N-1)$ операций. В итоге время работы программы будет пропорционально

$$\sim 24N - 21 \sim O(N^1).$$

Таким образом, выигрыш во времени счёта в случае реализации вырожденного метода Гаусса колоссален.

3.2.2. Использование разреженных матриц

Второй вариант оптимизации процесса решения СЛАУ (3.4), как в плане затрачиваемого на вычисления времени, так и в плане требуемой компьютерной памяти, заключается в использовании методологии хранения и оперирования с матрицами $\boldsymbol{D}$ и $\boldsymbol{f_y}$ в разреженном формате записи, функционал которого предоставляется MatLab'ом.

MatLab позволяет хранить разряженные матрицы, т. е. матрицы с преобладанием нулевых элементов, в виде набора трёх векторов $\boldsymbol{i}$, $\boldsymbol{j}$ и $\boldsymbol{s}$, которые определяют положения и значения ненулевых элементов в матрице S: $S\big(\boldsymbol{i}(k), \boldsymbol{j}(k)\big) = \boldsymbol{s}(k)$. Для того, чтобы задать матрицу размерности $m \times n$, у которой в строке с номером $\boldsymbol{i}(k)$ и столбце с номером $\boldsymbol{j}(k)$ будет стоять элемент матрицы со значением $\boldsymbol{s}(k)$, необходимо сгенерировать соответствующие вектора, а затем воспользоваться командой sparse.

```
S = sparse(i,j,s,m,n)
```

Лёгкая модификация приведённых выше MatLab-функций $\boldsymbol{D}$ и $\boldsymbol{f}_y$ позволяет получить функции, результатом которых являются соответствующие матрицы, хранимые в разреженном формате.

```
function D = D(eps,y,h,N)

    % Функция вычисляет матрицу дифференциального
    % оператора решаемой системы ОДУ

    % Входные данные:
    % eps - малый параметр
    % y - вектор решения системы ОДУ
    % на текущем временном слое
    % h - шаг сетки по переменной x
    % N - число интервалов сетки по переменной x

    % Выходные данные:
    % D - искомая матрица дифференциального оператора
    % (в разреженной форме записи)

    i = zeros(1,4*N - 6);
    j = zeros(1,4*N - 6);
    s = zeros(1,4*N - 6);

    k = 1;
    i(k) = 1; j(k) = 1; s(k) = 4/3 - ...
        (2 + eps*h^2*exp(eps*y(1)));
    k = k + 1;
    i(k) = 1; j(k) = 2; s(k) = -1/3 + 1;
    for n = 2:(N - 2)
        k = k + 1;
        i(k) = n; j(k) = n - 1; s(k) = 1;
        k = k + 1;
        i(k) = n; j(k) = n; s(k) = ...
            - (2 + eps*h^2*exp(eps*y(n)));
        k = k + 1;
        i(k) = n; j(k) = n + 1; s(k) = 1;
    end
    k = k + 1;
```

```
36      i(k) = N - 1; j(k) = N - 2; s(k) = 1 - 1/3;
37      k = k + 1;
38      i(k) = N - 1; j(k) = N - 1; s(k) = ...
39          -(2 + eps*h^2*exp(eps*y(N-1))) +4/3;
40      for n = N:(2*N - 2)
41          k = k + 1;
42          i(k) = n; j(k) = n; s(k) = 1;
43      end
44
45      D = sparse(i,j,s,2*N - 2,2*N - 2);
46
47  end
```

```
1   function f_y = f_y(y,h,N)
2
3       % Функция вычисляет матрицу Якоби правой части
4       % решаемой системы ОДУ
5
6       % Входные данные:
7       % y - решение системы ОДУ в текущий момент времени
8       % h - шаг сетки по переменной x
9       % N - число интервалов сетки по переменной x
10
11      % Выходные данные:
12      % f_y - искомая матрица Якоби
13      % (в разреженной форме записи)
14
15      i = zeros(1,4*N - 6);
16      j = zeros(1,4*N - 6);
17      s = zeros(1,4*N - 6);
18
19      k = 0;
20      for n = 1:(N-1)
21          k = k + 1;
22          i(k) = n; j(k) = N - 1 + n; s(k) = h^2;
23      end
24      k = k + 1;
25      i(k) = N; j(k) = 1; s(k) = (2 - 4/3)/h^2;
26      k = k + 1;
27      i(k) = N; j(k) = 2; s(k) = (-1 + 1/3)/h^2;
28      for n = (N + 1):(2*N - 3)
29          k = k + 1;
```

```
30            i(k) = n; j(k) = n - N; s(k) = -1/h^2;
31            k = k + 1;
32            i(k) = n; j(k) = n - N + 1; s(k) = 2/h^2;
33            k = k + 1;
34            i(k) = n; j(k) = n - N + 2; s(k) = -1/h^2;
35        end
36        k = k + 1;
37        i(k) = 2*N - 2; j(k) = N - 2; s(k) = (1/3 - 1)/h^2;
38        k = k + 1;
39        i(k) = 2*N - 2; j(k) = N - 1; s(k) = (-4/3 + 2)/h^2;
40
41        f_y = sparse(i,j,s,2*N - 2,2*N - 2);
42
43    end
```

Изменения остальных MatLab-функций для работы с матрицами в разреженном формате данных не требуется. При этом необходимо отметить, что вариант использования разреженных матриц для решения СЛАУ (3.4) работает медленнее, чем реализация вырожденного метода Гаусса, но при этом значительно быстрее, чем общий подход реализующий полный метод Гаусса для поиска решения системы (3.4).

Выборочный тестовый расчёт для набора параметров, при котором полный метод Гаусса находил решение за 4172 секунды, показал, что применение вырожденного метода Гаусса позволяет найти решение за 30 секунд, а использование операций с матрицами, хранимыми в разреженной форме, находит решение за 52 секунды. Таким образом, зачастую хранение и оперирование с матрицами в разреженном формате может сильно сэкономить время даже по сравнению с тем выигрышем во времени, который может дать разработка и реализация вырожденного метода Гаусса (на разработку и реализацию которого надо потратить дополнительное время, в отличие от варианта использования методики работы с матрицами в разреженной форме записи).

3.3. Численная диагностика разрушения решения

Практический алгоритм диагностики факта разрушения решения в целом повторяет алгоритм, введённый в пункте 2.4 «Численная диагностика разрушения решения» главы 2. Обсудим здесь только принципиальные отличия и некоторые тонкости его применения.

В связи с тем, что мы аппроксимировали все пространственные производные в (3.2) с точностью $O(h^2)$, а при численном интегрировании системы (3.3) используем схему CROS1 (3.4), в которую входит непостоянная матрица $\boldsymbol{D}\big(\boldsymbol{y}(t)\big)$, с точностью $O(\tau^1)$, построенный метод решения системы (3.1) имеет точность $O(\tau^1 + h^2)$, т.е. $p_x = p_x^{theor} \equiv 2$ и $p_t = p_t^{theor} \equiv 1$. Таким образом из условия согласования (2.9) следует, что коэффициенты сгущения r_x и r_t по разным переменным для выполнения условий применимости формулы Рунге–Ромберга (2.12) должны удовлетворять соотношению $r_x^2 = r_t^1$. Наиболее удобно для счёта выбрать $r_x = 2$ и $r_t = 4$.

Таким образом, после вычисления набора решений $u_{(s)}(x,t) \equiv u^{(r_x^{s-1}N, r_t^{s-1}M)}(x,t)$ на последовательности сгущающихся сеток $s = \overline{1,S}$, начиная с базовой $X_N \times T_M$: $\{x_n, t_m\}$, $0 \leqslant n \leqslant N$, $0 \leqslant m \leqslant M$, мы можем выполнить следующие оценки.

1. Оценить оценить эффективный порядок точности на всём промежутке времени $t \in [t_0, T]$ решения системы (3.3) по формуле (2.17)

$$p_{t_{(s)}}^{eff} = \log_{r_t} \frac{\sqrt{\sum\limits_{n=0}^{N}\sum\limits_{m=0}^{M}\big(u_{(s-1)}(x_n,t_m) - u_{(s-2)}(x_n,t_m)\big)^2}}{\sqrt{\sum\limits_{n=0}^{N}\sum\limits_{m=0}^{M}\big(u_{(s)}(x_n,t_m) - u_{(s-1)}(x_n,t_m)\big)^2}}.$$

2. Оценить с точностью до шага базовой сетки T_M по времени локализацию конкретного момента времени, в который происходит разрушение решения, выполнив оценки эффективного порядка точности поточечно в каждом узле $t_m \in T_M$, $1 \leqslant m \leqslant M$, по формуле (2.18),

$$p_{t_{(s)}}^{eff}(t_m) = \log_{r_t} \frac{\sqrt{\sum\limits_{n=0}^{N} \left(u_{(s-1)}(x_n, t_m) - u_{(s-2)}(x_n, t_m)\right)^2}}{\sqrt{\sum\limits_{n=0}^{N} \left(u_{(s)}(x_n, t_m) - u_{(s-1)}(x_n, t_m)\right)^2}}.$$

Замечание. Отметим, что, как и в пункте 2.4 главы 2, в узле t_0 базовой сетки T_M эффективный порядок точности $p_{t_{(s)}}^{eff}(t_0)$ не вычисляется, так как в этом узле на любой сетке решение задано точно начальным условием.

3. Оценить с точностью до шага базовой сетки X_N по пространству локализацию пространственных точек разрушения решения для каждого конкретного момента времени $t_m in T_M$, $1 \leqslant m \leqslant M$, по формуле (2.19) в каждом узле $x_n \in X_N$, $0 \leqslant n \leqslant N$,

$$p_{xt_{(s)}}^{eff}(x_n, t_m) = \log_{r_t} \frac{|u_{(s-1)}(x_n, t_m) - u_{(s-2)}(x_n, t_m)|}{|u_{(s)}(x_n, t_m) - u_{(s-1)}(x_n, t_m)|}.$$

Замечание. Отметим, что, в отличие от пункта 2.4 главы 2, эффективные порядки точности $p_{xt_{(s)}}^{eff}(x, t_m)$ могут быть вычислины в обоих граничных узлах x_0 и x_N, так как на границах функция задана не условиями Дирихле, а условиями Неймана, т.е. точные значения функции в граничных узлах при численном счёте нам не известны, а лишь оцениваются приближённо численно.

Далее приводится пример набора MatLab-команд, оформленных в виде отдельного файла test_3_1.m, которые позволяют

посредством многократного запуска введённой в этом разделе MatLab-функции PDESolving получить набор сеточных решений $u_{(s)}(x,t) \equiv u^{(r_x^{s-1}N, r_t^{s-1}M)}(x,t)$, $s = \overline{1,S}$, задачи (3.1) с набором параметров (3.9) на различных сетках, начиная с базовой сетки $X_N \times T_M$.

```
% Определение начального времени счёта
t_0 = 0;

% Определение границ отрезка  x ∈ [a,b]
a = 0; b = pi;

% Определение числа интервалов базовой сетки
N = 50; M = 50;

% Определение параметра задачи
epsilon = 10^(10);

% Определение начальных условий
u_init_0 = @(x) 0;
u_init_1 = @(x) -(x*(pi-x))^2*sin(x/3);

% Определение аналитической оценки сверху
% времени разрушения решения
T = TimeOfBlowUpCalculation...
    (u_init_0,u_init_1,a,b,100,epsilon);

S = 7;    % Число сеток, на которых ищется
          % приближённое решение
r_x = 2; % Коэффициент сгущения сетки по x
r_t = 4; % Коэффициент сгущения сетки по t

% Выделение памяти под массивы сеточных значений
% решений ОДУ на разных сетках с номерами  s = 1,S
% Первый индекс - номер сетки s из последовательности
% сгущающихся сеток, на которых ищется решение
% Второй и третий индексы определяют массив,
% в строке с номером (m + 1) которого  хранятся
% сеточные значения решения, соответствующего
% моменту времени  t_m , из узлов,
% совпадающих с узлами базовой сетки
```

```
array_of_u = zeros(S,M + 1,N + 1);

% "Большой цикл", который пересчитывает решение S раз
% на последовательности сгущающихся сеток
% Массив сеточных значений решения содержит только
% сеточные значения из узлов,
% совпадающих с узлами базовой сетки
for s = 1:S
    u = PDESolving(epsilon,a,b,N,t_0,T,M, ...
        u_init_0,u_init_1,s,r_x,r_t);
    array_of_u(s,:,:) = u;
end

% Сохраняем необходимые для дальнейшей диагностики
% разрушения решения данные Workspace'a в файл
save('data.mat','array_of_u','N','M', ...
    'r_x','r_t','S','a','b','t_0','T');
```

Отметим, что в качестве финального времени счёта T используется результат вычисления априорной оценки сверху времени разрушения решения, полученной для задачи (3.1) в [12, формула (2.5)]

$$T_{bl} = -\frac{1}{\varepsilon}\frac{\int_a^b e^{\varepsilon u_{init_0}(x)}dx}{\int_a^b e^{\varepsilon u_{init_0}(x)}u_{init_1}(x)\,dx}.$$

Вычисления по этой формуле реализуются с помощью вспомогательной MatLab-функции TimeOfBlowUpCalculation.

```
function T = TimeOfBlowUpCalculation ...
    (u_init_0,u_init_1,a,b,N,eps)

    % Функция вычисляет аналитическую верхнюю оценку
    % для времени разрушения решения

    h = (b - a)/N; % Определение шага сетки по x

    Int1 = 0;
    Int2 = 0;
    for n = 1:N
        Int1 = Int1 + exp(eps*u_init_0(a + h*n - h/2))*h;
        Int2 = Int2 + exp(eps*u_init_0(a + h*n - h/2))* ...
```

```
14                u_init_1(a + h*n - h/2)*h;
15        end
16
17        T = -(1/eps)*(Int1/Int2);
18
19    end
```

Как и ранее, результатом работы этого MatLab-кода test_3_1.m будет являться файл data.mat, содержимое которого будет подгружаться функциями, диагностирующими факт разрушения решения, без повторного расчёта решений на последовательности сгущающихся сеток.

MatLab-файл BlowUpDiagnostics.m, введённый в пункте 2.4 главы 2, который вычисляет эффективные порядки точности приближённого решения на промежутке времени $t \in [t_0, T]$, используя данные из уже сформированного файла data.mat, никаких изменений содрежать не будет, поэтому мы его здесь не дублируем.

MatLab-файл BlowUpDiagnostics_for_each_t.m, который реализует вычисление эффективных порядков точности приближённого решения в узлах, совпадающих с узлами t_m, $0 \leqslant m \leqslant M$, базовой сетки T_M, используя данные из уже сформированного файла data.mat, для определения конкретного момента времени, в который происходит разрушение решения, будет содержать изменения только в коде, отвечающем за отрисовку результов расчёта. Необходимо заменить команду, отвечающую за отрисовку теоретических значений порядка точности, на следующую (которая соответствует $p_t^{theor} = 1$).

```
1 % Рисуется зависимость теоретического порядка точности
2 % от временного узла базовой сетки
3 plot(t,t*0 + 1,'-*k','MarkerSize',3); hold on;
```

MatLab-файл BlowUpDiagnostics_for_specified_t.m, который реализует вычисление эффективных порядков точности приближённого решения в узлах, совпадающих с узлами x_n, $0 \leqslant n \leqslant N$, базовой сетки X_N по пространству, в конкретный мо-

мент времени $t_m \in T_M$ (2.19), используя данные файла data.mat, с целью локализации разрушения по пространственной переменной x, будет содержать следующие изменения. Главным является то, что вычисление эффективных порядков точности проводится во всех узлах сетки X_N, включая оба граничные узла. Второстепенным изменением является изменение команд отрисовки полученных результатов.

```
% Загрузка результата вычисления приближённого решения
% на последовательности из S сгущающихся сеток
load('data.mat');

% Выделение памяти под массив значений эффективных
% порядков точности расчёта приближённого решения
% в каждом узле  x_n,  0 ≤ n ≤ N  (второй индекс массива),
% и на разных сетках (первый индекс массива)
p_eff_ForSpecifiedTime = zeros(S,N + 1);

% Вычисление эффективных порядков точности
% во всех пространственных точках
% на определённом временном слое  t_{m-1}  с номером (m-1)
% (с учётом сдвига индекса в MatLab'е на +1)
m = 31;
for n = 1:(N + 1)

    % Вычисление  p^{eff}_{(1)}(x_n)  и  p^{eff}_{(2)}(x_n)  невозможно
    p_eff_ForSpecifiedTime(1,n) = NaN;
    p_eff_ForSpecifiedTime(2,n) = NaN;

    for s = 3:S
        p_eff_ForSpecifiedTime(s,n) = log(...
          abs(array_of_u(s-1,m,n)-array_of_u(s-2,m,n))/...
           abs(array_of_u(s,m,n)-array_of_u(s-1,m,n)))/...
            log(r_t);
    end
end

% Отрисовка результатов расчётов для сетки с номером s
% S = 7;
figure;
```

```
x = a:(b - a)/N:b; % Определение базовой сетки
% Рисуется зависимость теоретического порядка точности
% от пространственного узла базовой сетки
plot(x,x*0 + 1,'-*k','MarkerSize',3); hold on;
% Рисуется зависимость эффективного порядка точности
% от пространственного узла базовой сетки
plot(x,p_eff_ForSpecifiedTime(S,:), ...
     '-sk','MarkerSize',5,'LineWidth',1);
axis([x(1) x(N + 1) -2.0 3.0]);
xlabel('x'); ylabel('p^{eff}');
```

В MatLab-файле draw.m, осуществляющем отрисовку решения, следует внести изменения только в команду axis, в которой необходимо изменить пределы отрисовки графика по оси ординат для каждого отдельного рассматриваемого далее примера.

Пример 1. Сначала рассмотрим тестовый пример с известным аналитическим результатом о точном моменте времени разрушения.

В случае набора параметров

$$\begin{aligned} &u_{init_0}(x) \equiv 0,\ u_{init_1}(x) \equiv -1, \\ &a = 0,\ b = \pi,\ t_0 = 0,\ \varepsilon = 1, \end{aligned} \tag{3.7}$$

решение задачи (3.1) будет иметь вид

$$u(x,t) = \ln(1-t). \tag{3.8}$$

Очевидно, что время разрушения решения может быть определено точно как $T_{bl} = 1$, в окрестности которого решение имеет логарифмический характер роста.

Для численных расчётов в качестве базовой сетки $X_N \times T_M$ была взята сетка с $N = 50$ и $M = 50$ интервалами, и было выполнено последовательное сгущение сеток с коэффициентами сгущения $r_x = 2$ и $r_t = 4$. На рис. 3.3 представлен результат расчётов (посредством последовательного запуска набора MatLab-команд из файлов test_3_1.m $\rightarrowtail$ draw.m) после вычисления решения на 7-й сетке ($s = 7$),

На рис. 3.4 представлен результат вычислений (посредством последовательного запуска набора MatLab-команд из файлов

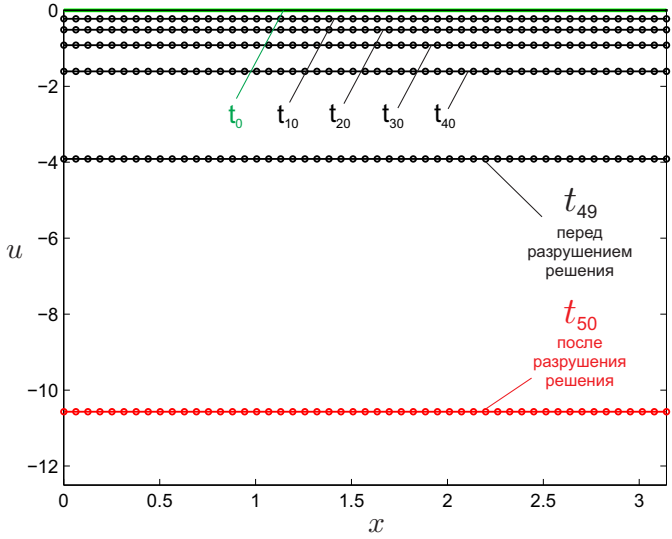

Рис. 3.3: Результат расчёта решения $u_{(s)}(x,t) \equiv u^{(r_x^{s-1}N, r_t^{s-1}M)}(t)$ задачи (3.1) с входными данными (3.7) для следующего набора параметров: $N = 50$, $M = 50$, $r_x = 2$, $r_t = 4$, $s = 7$. Отмечены только узлы, совпадающие с узлами базовой сетки.

test_3_1.m $\rightarrowtail$ BlowUpDiagnostics_for_each_t.m), демонстрирующий явный выход поточечных значений эффективных порядков точности $p_{t_{(s)}}^{eff}(t_m)$, $0 \leqslant m \leqslant M$, на асимптотически точные значения.

Таким образом, мы можем сделать следующий вывод о полученном численном решении задачи задачи (3.1) для набора параметров (3.7). После вычисления на $S = 7$ вложенных сетках поточечные значения эффективного порядка точности $p_{t_{(s)}}^{eff}(t)$ для каждого момента времени $t_m \in T_M$ до $m = 49$ включительно сходятся к $p_t{}^{theor} \equiv 1$, а для последнего 50-го узла сетки (соответствующего моменту времени $t_{50} = 1$) стремятся к 0. Это

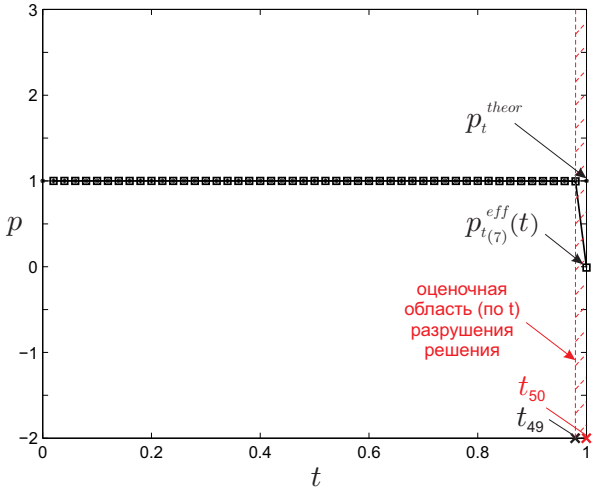

Рис. 3.4: Результат вычислений эффективных порядков точности $p_{t_{(S)}}^{eff}(t_m)$, $1 \leqslant m \leqslant M$, задачи (3.1) с набором входных данных (3.7) для следующего набора численных параметров: $N = 50$, $M = 50$, $r_x = 2$, $r_t = 4$, $S = 7$.

означает, что разрушение происходит в момент $T_{bl} \in (t_{49}, t_{50}] \equiv (0.980, 1.000]$, а видимое стремление эффективного порядка точности при $m = 50$ к 0 позволяет предположить, что в точке T_{bl} решение имеет особенность типа логарифмического роста $u(x,t) \sim \ln(T_{bl} - t)$, что полностью согласуется с известным аналитическим результатом (3.8).

Далее мы можем исследовать вопрос о характере разрушения решения по пространственной перемеменной. На рис. 3.5 представлен результат вычислений (посредством последовательного запуска набора MatLab-команд из файлов test_3_1.m $\rightarrowtail$ BlowUpDiagnostics_for_specidied_t.m) в различные моменты времени до и после (или, возможно, в момент) разрушения

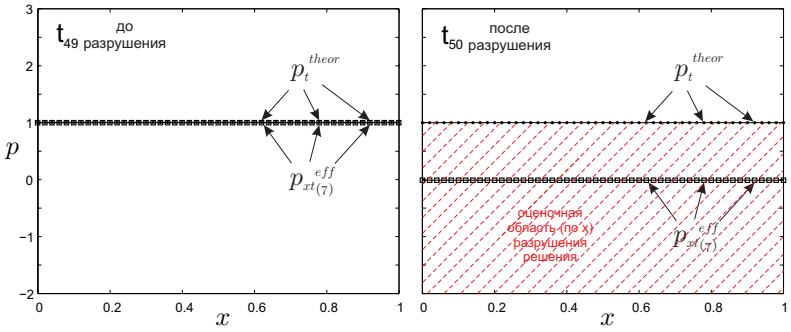

Рис. 3.5: Результат вычислений эффективных порядков точности $p_{xt(S)}^{eff}(x, t_m)$ задачи (3.1) с набором входных данных (3.7) для следующего набора параметров: $N = 50$, $M = 50$, $r_x = 2$, $r_t = 4$, $S = 7$. Представлены графики для $m \in \{49, 50\}$.

решения. Хорошо видно, что отклонение сходимости $p_{xt(s)}^{eff}(x, t)$ от теоретического значения $p_t^{theor} = 1$ возникает во всех точках сетки X_N в первом же узле после (или в момент) разрушения $t \geqslant T_{bl}$.

В итоге мы можем сделать вывод о том, какой части численного решения (см. рис. 3.4), полученного на 7-й вложенной сетке, мы можем доверять, а какой нет.

Пример 2. Теперь рассмотрим пример для набора параметров

$$u_{init_0}(x) \equiv 0,\ u_{init_1}(x) \equiv -\big(x(\pi - x)\big)^2 \sin\frac{x}{3},$$
$$a = 0,\ b = \pi,\ t_0 = 0,\ \varepsilon = 10^{10}. \tag{3.9}$$

В этом случае аналитическое решение не известно, поэтому применим численный алгоритм диагностики факта разрушения решения.

Для численных расчётов в качестве базовой сетки $X_N \times T_M$ была взята сетка с $N = 50$ и $M = 50$ интервалами, и было выполнено последовательное сгущение сеток с коэффициентами сгуще-

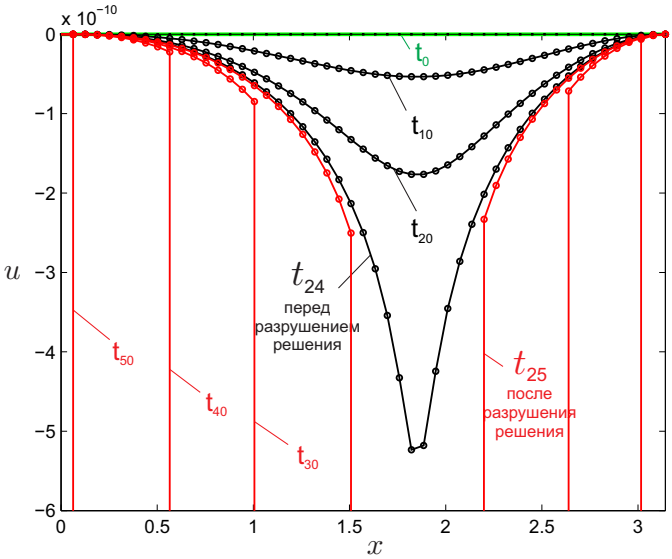

Рис. 3.6: Результат расчёта решения $u_{(s)}(x,t) \equiv u^{(r_x^{s-1}N, r_t^{s-1}M)}(t)$ задачи (3.1) с входными данными (3.9) для следующего набора параметров: $N = 50$, $M = 50$, $r_x = 2$, $r_t = 4$, $s = 7$. Отмечены только узлы, совпадающие с узлами базовой сетки.

ния $r_x = 2$ и $r_t = 4$. На рис. 3.6 представлен результат расчётов (посредством последовательного запуска набора MatLab-команд из файлов test_3_1.m $\rightarrowtail$ draw.m) после вычисления решения на 7-й сетке ($s = 7$),

На рис. 3.7 представлен результат вычислений (посредством последовательного запуска набора MatLab-команд из файлов test_3_1.m $\rightarrowtail$ BlowUpDiagnostics_for_each_t.m), демонстрирующий результат предположительного выхода поточечных значений эффективных порядков точности $p_{t_{(s)}}^{eff}(t_m)$, $0 \leqslant m \leqslant M$, на асимптотически точные значения.

Таким образом, мы можем сделать следующий вывод о по-

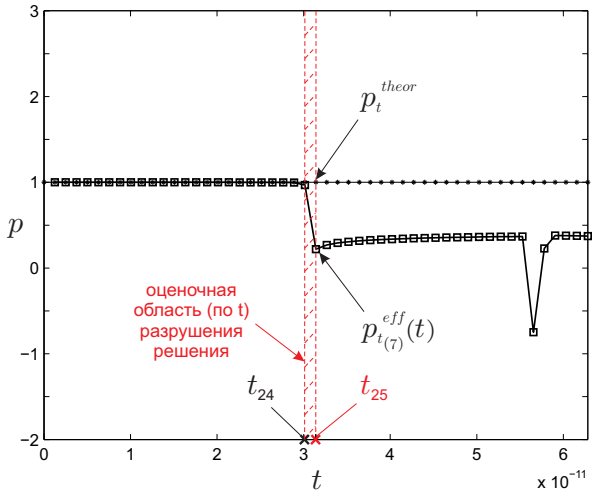

Рис. 3.7: Результат вычислений эффективных порядков точности $p_{t_{(S)}}^{eff}(t_m)$, $1 \leqslant m \leqslant M$, задачи (3.1) с набором входных данных (3.9) для следующего набора численных параметров: $N = 50$, $M = 50$, $r_x = 2$, $r_t = 4$, $S = 7$.

лученном численном решении задачи задачи (3.1) для набора параметров (3.9). После вычисления на $S = 7$ вложенных сетках поточечные значения эффективного порядка точности $p_{t_{(s)}}^{eff}(t)$ для каждого момента времени $t_m \in T_M$ до $m = 24$ включительно сходятся к ${p_t}^{theor} \equiv 1$, а для больших значений m, по-видимому, стремятся к 0. Это означает, что разрушение происходит в момент $T_{bl} \in (t_{24}, t_{25}] \equiv (3.015, 3.141] \cdot 10^{-11}$, а видимое стремление эффективного порядка точности при $m \geqslant 25$ к 0 позволяет предположить, что в точке T_{bl} решение имеет особенность типа логарифмического роста $u(x,t) \sim \ln(T_{bl} - t)$.

Далее мы можем исследовать вопрос о характере разрушения решения по пространственной перемеменной. На рис. 3.8

представлен результат вычислений (посредством последовательного запуска набора MatLab-команд из файлов test_3_1.m $\rightarrowtail$ BlowUpDiagnostics_for_specidied_t.m) в различные моменты времени до и после (или, возможно, в момент) разрушения решения. Хорошо видно, что отклонение сходимости $p_{xt_{(s)}}^{eff}(x,t)$ от теоретического значения $p_t^{theor} = 1$ возникает изначально во внутренних узлах сетки X_N, а затем распространяется на весь отрезок по пространственной переменной $x \in [a,b]$.

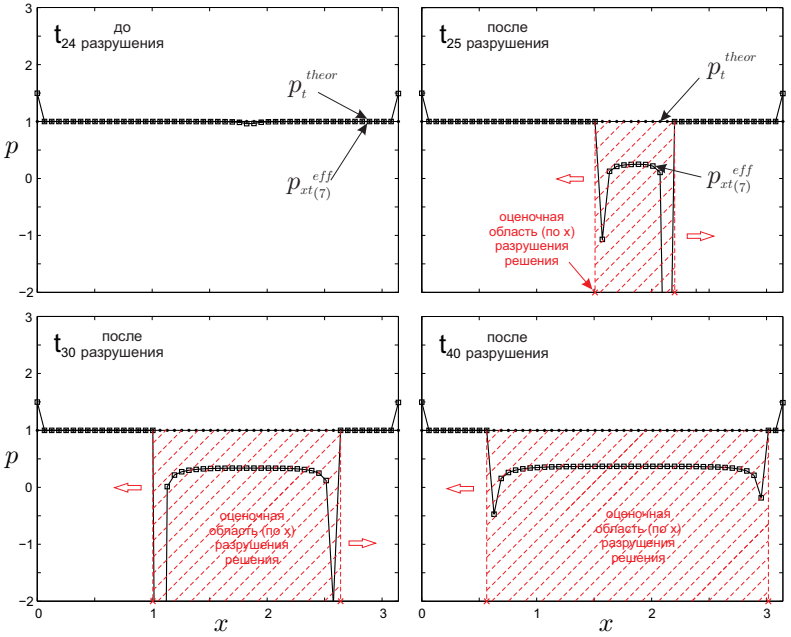

Рис. 3.8: Результат вычислений эффективных порядков точности $p_{xt_{(S)}}^{eff}(x,t_m)$ задачи (3.1) с набором входных данных (3.7) для следующего набора параметров: $N = 50$, $M = 50$, $r_x = 2$, $r_t = 4$, $S = 7$. Представлены графики для $m \in \{49, 50\}$.

В итоге мы можем сделать вывод о том, какой части численного решения (см. рис. 3.4), полученного на 7-й вложенной сетке, мы можем доверять, а какой нет.

3.4. Построение схемы со 2-м порядком точности по времени

Как уже было отмечено ранее, схема Розенброка с комплексным коэффициентом CROS1 (3.4) имеет порядок точности $O(\tau^1)$ в связи с тем, что матричная функция $\boldsymbol{D}(\boldsymbol{y})$ не является постоянной и зависит от переменной $\boldsymbol{y}$ (подробные объяснения приведены в статьях [2, 9]). Однако схема CROS1 будет иметь порядок точности $O(\tau^2)$ в случае постоянной матричной функции $\boldsymbol{D}$. Этого можно добиться, если мы введём в (3.2) ещё одну вспомогательную переменную $g = u_{xx} - e^{\varepsilon u}$, в результате чего под операторами взятия частных производных по времени не будут содержаться нелинейные функции аргументов u, v или g. Рассмотрим данный вариант решения исходной задачи (3.1) подробнее.

В результате указанной дополнительной замены $g = u_{xx} - e^{\varepsilon u}$ в (3.2) исходная начально-краевая задача (3.1) примет вид

$$\begin{cases} \dfrac{\partial}{\partial t} g = v, \quad x \in (a,b), \quad t \in (t_0, T], \\ \dfrac{\partial}{\partial t} v + u_{xx} = 0, \quad x \in (a,b), \quad t \in (t_0, T], \\ g = u_{xx} - e^{\varepsilon u}, \quad x \in (a,b), \quad t \in (t_0, T], \\ u_x(a,t) = 0, \quad u_x(b,t) = 0, \quad t \in (t_0, T], \\ u(x,t_0) = u_{init_0}(x), \quad x \in [a,b], \\ v(x,t_0) = u_{init_1}(x)_{xx} - \varepsilon u_{init_1}(x) e^{\varepsilon u_{init_0}(x)}, \quad x \in (a,b), \\ g(x,t_0) = u_{init_0}(x)_{xx} - e^{\varepsilon u_{init_0}(x)}, \quad x \in (a,b). \end{cases} \tag{3.10}$$

Далее выполним последовательность действий, аналогичную той, что описана в разделе 3.1. В результате после конечно-разностной аппроксимации пространственных производных со

вторым порядком точности на равномерной сетке X_N по пространственной переменной мы получим дифференциально-алгебраическую систему, из которой требуется определить $N+1$ неизвестную функцию $u_n \equiv u_n(t) \equiv u(x_n,t)$, $n = \overline{0,N}$, $N-1$ вспомогательную функцию $v_n \equiv v_n(t) \equiv v(x_n,t)$, $n = \overline{1,N-1}$, и $N-1$ вспомогательную функцию $g_n \equiv g_n(t) \equiv g(x_n,t)$, $n = \overline{1,N-1}$. Для удобства запишем эту систему в следующем виде:

$$
\begin{cases}
\dfrac{dg_n}{dt} = v_n, \quad n = \overline{1,N-1}, \quad t \in (t_0,T], \\
\dfrac{dv_n}{dt} = -\dfrac{u_{n+1} - 2u_n + u_{n-1}}{h^2}, \quad n = \overline{1,N-1}, \quad t \in (t_0,T], \\
0 = u_0 - \dfrac{4}{3}u_1 + \dfrac{1}{3}u_2, \quad t \in (t_0,T], \\
0 = g_n - \dfrac{u_{n+1} - 2u_n + u_{n-1}}{h^2} + e^{\varepsilon u_n}, \; n = \overline{1,N-1}, \; t \in (t_0,T], \\
0 = u_N - \dfrac{4}{3}u_{N-1} + \dfrac{1}{3}u_{N-2}, \quad t \in (t_0,T], \\
u_n(t_0) = u_{init_0}(x_n), \quad n = \overline{0,N}, \\
v_n(t_0) = \dfrac{u_{init_1}(x_{n+1}) - 2u_{init_1}(x_n) + u_{init_1}(x_{n-1})}{h^2} - \\
\qquad\qquad - \varepsilon u_{init_1}(x_n) e^{\varepsilon u_{init_0}(x_n)}, \quad n = \overline{1,N-1}, \\
g_n(t_0) = \dfrac{u_{init_0}(x_{n+1}) - 2u_{init_0}(x_n) + u_{init_0}(x_{n-1})}{h^2} - \\
\qquad\qquad - e^{\varepsilon u_{init_0}(x)}, \quad n = \overline{1,N-1}.
\end{cases}
$$

Полученная система является дифференциально-алгебраической, поскольку включает в себя как дифференциальные уравнения, так и алгебраические (два уравнения, определяемые граничными условиями, и уравнения, определяемые условием связи $g = u_{xx} - e^{\varepsilon u}$).

В отличие от подхода, применённого в разделе 3.1, выполнять подстановку выражений для u_0 и u_N через их соседние сеточные значения в остальные дифференциальные уравнения мы

не будем, так как свести эту дифференциально-алгебраическую систему к чисто дифференциальной у нас не получится за счёт наличия условия связи. Но даже в случае наличия алгебраических уравнений алгоритм решения этой системы не изменится. К тому же, в данном учебно-методическом пособии мы преследуем цель по возможности продемонстрировать как можно больше различных (но по сути эквивалентых) подходов к решению рассматриваемых задач.

Полученная дифференциально-алгебраическая система, содержащая $3N-1$ уравнение и $3N-1$ неизвестную функцию u_n, $n=\overline{0,N+1}$, и v_n, v_n, $n=\overline{1,N-1}$, может быть переписана в следующем виде:

$$\begin{cases} \boldsymbol{D}\dfrac{d\boldsymbol{y}}{dt} = \boldsymbol{f}(\boldsymbol{y}), \quad t \in (t_0, T], \\ \boldsymbol{y}(t_0) = \boldsymbol{y}_{init}, \end{cases} \tag{3.11}$$

где $\boldsymbol{y} = \begin{pmatrix} u_0 & u_1 & u_2 & \dots & u_{N-1} & u_N & v_1 & v_2 & \dots & v_{N-1} & g_1 & g_2 & \dots & g_{N-1} \end{pmatrix}^T$, $\boldsymbol{f} = \begin{pmatrix} f_1 & f_2 & \dots & f_{3N-1} \end{pmatrix}^T$ и $\boldsymbol{y}_{init} = \begin{pmatrix} u_0(t_0) & \dots & u_N(t_0) & v_1(t_0) & \dots & v_{N-1}(t_0) & g_1(t_0) & \dots & g_{N-1}(t_0) \end{pmatrix}^T$.

Здесь вектор-функция $\boldsymbol{f}$ имеет следующую структуру:

$$f_n = \begin{cases} y(n+N+1), & \text{если } n = \overline{1,N-1}, \\ -\dfrac{y_{n-N+3} - 2y_{n-N+2} + y_{n-N+1}}{h^2}, & \text{если } n = \overline{N,2N-2}, \\ y_1 - \frac{4}{3}y_2 + \frac{1}{3}y_3, & \text{если } n = 2N-1, \\ y_{n+1} - \frac{y_{n-2N+3} - 2y_{n-2N+2} + y_{n-2N+1}}{h^2} + & \\ \qquad + e^{\varepsilon y_{n-2N+2}}, & \text{если } n = \overline{2N,3N-2}, \\ y_{N+1} - \frac{4}{3}y_N + \frac{1}{3}y_{N-1}, & \text{если } n = 3N-1. \end{cases}$$

Ниже приводится пример MatLab-функции, которая реализует вычисление компонент вектор-функции $\boldsymbol{f}$.

```
function f = f(eps,y,h,N)

    % Функция вычисляет вектор правой части
    % решаемой системы ОДУ

    % Входные данные:
    % eps - малый параметр
    % y - вектор решения системы ОДУ
    % на текущем временном слое
    % h - шаг сетки по переменной x
    % N - число интервалов сетки по переменной x

    % Выходные данные:
    % f - искомый вектор f

    f = zeros(3*N - 1,1);

    for n = 1:(N - 1)
        f(n) = y(n + N + 1);
    end
    for n = N:(2*N - 2)
        f(n) = -(y(n - N + 3) - 2*y(n - N + 2) + ...
            y(n - N + 1))/h^2;
    end
    f(2*N - 1) = y(1) - 4/3*y(2) + 1/3*y(3);
    for n = 2*N:(3*N - 2)
        f(n) = y(n + 1) - (y(n - 2*N + 3) - ...
            2*y(n - 2*N + 2) + y(n - 2*N + 1))/h^2 +...
            exp(eps*y(n - 2*N + 2));
    end
    f(3*N - 1) =  y(N + 1) - 4/3*y(N) + 1/3*y(N - 1);

end
```

А матрица $\boldsymbol{D}$ имеет следующие ненулевые элементы:

$$\begin{aligned} D_{n,n+2N} &= 1, \quad \text{если } n = \overline{1, N-1}, \\ D_{n,n+2} &= 1, \quad \text{если } n = \overline{N, 2N-2}, \end{aligned}$$

Ниже приводится пример MatLab-функции, которая реализует вычисление компонент матрицы $\boldsymbol{D}$.

```
function D = D(N)

    % Функция вычисляет матрицу дифференциального
    % оператора решаемой системы ОДУ

    % Входные данные:
    % N - число интервалов сетки по переменной x

    % Выходные данные:
    % D - искомая матрица дифференциального оператора

    D = zeros(3*N - 1,3*N - 1);

    for n = 1:(N - 1)
        D(n,n + 2*N) = 1;
    end
    for n = N:(2*N - 2)
        D(n,n + 2) = 1;
    end

end
```

В результате после введения равномерной сетки T_M по временной переменной мы можем применить схему Розенброка CROS1 для решения системы (3.11):

$$\begin{aligned} &\boldsymbol{y}(t_{m+1}) = \boldsymbol{y}(t_m) + (t_{m+1} - t_m)\,\mathrm{Re}\,\boldsymbol{w}_1\,, \\ &\text{где } \boldsymbol{w}_1 \text{ является решением СЛАУ} \\ &\left[\boldsymbol{D} - \frac{1+i}{2}(t_{m+1} - t_m)\,\boldsymbol{f_y}\Big(\boldsymbol{y}(t_m)\Big)\right]\boldsymbol{w}_1 = \boldsymbol{f}\Big(\boldsymbol{y}(t_m)\Big). \end{aligned} \tag{3.12}$$

Здесь $\boldsymbol{f_y}$ — матрица с элементами $(f_y)_{n,m} \equiv \frac{\partial f_n}{\partial y_m}$ (матрица Якоби), которая для рассматриваемой системы имеет следующие ненулевые элементы:

$$(f_y)_{n,n+N+1} = 1, \quad \text{если } n = \overline{1, N-1},$$

$$(f_y)_{n,n-N+3} = -\frac{1}{h^2}, \quad \text{если } n = \overline{N, 2N-2},$$

$$(f_y)_{n,n-N+2} = \frac{2}{h^2}, \quad \text{если } n = \overline{N, 2N-2},$$

$$(f_y)_{n,n-N+1} = -\frac{1}{h^2}, \quad \text{если } n = \overline{N, 2N-2},$$

$$(f_y)_{2N-1,1} = 1, \quad (f_y)_{2N-1,2} = -\frac{4}{3}, \quad (f_y)_{2N-1,3} = \frac{1}{3},$$

$$(f_y)_{n,n+1} = 1, \quad \text{если } n = \overline{2N, 3N-2},$$

$$(f_y)_{n,n-2N+3} = -\frac{1}{h^2}, \quad \text{если } n = \overline{2N, 3N-2},$$

$$(f_y)_{n,n-2N+2} = \frac{2}{h^2} + \varepsilon e^{\varepsilon y_{n-2N+2}}, \quad \text{если } n = \overline{2N, 3N-2},$$

$$(f_y)_{n,n-2N+1} = -\frac{1}{h^2}, \quad \text{если } n = \overline{2N, 3N-2},$$

$$(f_y)_{3N-1,N-1} = \frac{1}{3}, \quad (f_y)_{3N-1,N} = -\frac{4}{3}, \quad (f_y)_{3N-1,N+1} = 1.$$

Ниже приводится пример MatLab-функции, которая реализует вычисление компонент матрицы Якоби $\boldsymbol{f_y}$.

```
function f_y = f_y(eps,y,h,N)

    % Функция вычисляет матрицу Якоби правой части
    % решаемой системы ОДУ

    % Входные данные:
    % eps - малый параметр
    % y - вектор решения системы ОДУ
```

```
9       % на текущем временном слое
10      % h - шаг сетки по переменной x
11      % N - число интервалов сетки по переменной x
12
13      % Выходные данные:
14      % f_y - искомая матрица Якоби
15
16      f_y = zeros(3*N - 1,3*N - 1);
17
18      for n = 1:(N - 1)
19          f_y(n,n + N + 1) = 1;
20      end
21      for n = N:(2*N - 2)
22          f_y(n,n - N + 3) = -1/h^2;
23          f_y(n,n - N + 2) = 2/h^2;
24          f_y(n,n - N + 1) = -1/h^2;
25      end
26      f_y(2*N - 1,1) = 1;
27      f_y(2*N - 1,2) = -4/3;
28      f_y(2*N - 1,3) = 1/3;
29      for n = 2*N:(3*N - 2)
30          f_y(n,n + 1) = 1;
31          f_y(n,n - 2*N + 3) = -1/h^2;
32          f_y(n,n - 2*N + 2) = 2/h^2 + ...
33              eps*exp(eps*y(n - 2*N + 2));
34          f_y(n,n - 2*N + 1) = -1/h^2;
35      end
36      f_y(3*N - 1,N - 1) = 1/3;
37      f_y(3*N - 1,N) = -4/3;
38      f_y(3*N - 1,N + 1) = 1;
39
40  end
```

В результате для поиска численного решения задачи (3.1) в видоизменённой форме (3.10)–(3.11) по схеме (3.12) с использованием приведённых выше функции f, D и f_y необходимо выполнить в MatLab-функции PDESolving следующие изменения.

1) Необходимо заменить кусок кода, находящийся в строках 42–54 и отвечающий за определение размерности вектора $\boldsymbol{y}$ и задание его в начальный момент времени, его следующей модификацией.

```
1       % Выделение памяти под массив сеточных значений
2       % решения системы ОДУ, соответствующего
3       % текущему моменту времени  t_m
4       y = zeros(1,3*N - 1);
5
6       % Задание начального условия решаемой системы ОДУ
7       for n = 1:(N + 1)
8           y(1,n) = u_init_0(x(n));
9       end
10      for n = 1:(N - 1)
11          y(1,n + N + 1) = (u_init_1(x(n + 2)) - ...
12              2*u_init_1(x(n +1)) + u_init_1(x(n)))/h^2 - ...
13              eps*u_init_1(x(n + 1))*...
14              exp(eps*u_init_0(x(n + 1)));
15          y(1,n + 2*N) = (u_init_0(x(n + 2)) - ...
16              2*u_init_0(x(n +1)) + u_init_0(x(n)))/h^2 - ...
17              exp(eps*u_init_0(x(n + 1)));
18      end
```

2) Необходимо заменить кусок кода, находящийся в строках 74–106 и отвечающий за реализацию схемы CROS1 и выбор совпадающих узлов с узлами базовой сетки, его следующей модификацией.

```
1           % Реализация схемы CROS1
2
3           w_1 = (D(N) - (1+1i)/2*(t(m + 1) - ...
4               t(m))*f_y(eps,y,h,N))\f(eps,y,h,N);
5
6           y = y + (t(m + 1) - t(m))*real(w_1)';
7
8           % Выполнение проверки совпадения  t_{m+1}
9           % на сгущённой сетке с  t_{m_basic}  базовой сетки
10          if (m + 1) == (m_basic - 1)*r_t^(s - 1) + 1
11
12              % Заполнение массива сеточных значений решения
13              % исходной задачи для УрЧП
14
15              % На текущем временном слое выбираются
16              % пространственные узлы, совпадающие
17              % с узлами базовой сетки
18              for n = 1:(N_0 + 1)
```

```
19                  u(m_basic,n) = y((n - 1)*r_x^(s-1) + 1);
20              end
21
22              % Теперь будет отслеживаться совпадение  t_{m+1}
23              % на сгущённой сетке с очередным  t_{m_basic}
24              % базовой сетки
25              m_basic = m_basic + 1;
26
27          end
```

Отметим, что матрица системы (3.12) имеет специальный вид (см. рис. 3.9), в связи с чем численный счёт можно оптимизировать как с помощью разработки вырожденного метода Гаусса, учитывающего специальный вид матрицы системы, так и с помощью использования матриц, хранимых в разреженной форме записи.

3.5. Практические задания

Задача 1. Решите численно начально-краевую задачу

$$\begin{cases} \dfrac{\partial^2}{\partial t^2}\left(u_{xx} - e^{\varepsilon u}\right) + u_{xx} = 0, \quad x \in (a,b), \quad t \in (t_0, T], \\ u(a,t) = 0, \quad u_x(a,t) = 0, \quad t \in (t_0, T], \\ u(x,t_0) = u_{init_0}(x), \ x \in [a,b], \\ u_t(x,t_0) = u_{init_1}(x), \ x \in [a,b], \end{cases} \tag{3.13}$$

для следующего набора параметров:

$$t_0 = 0, \quad a = 0, \quad b = \pi, \quad T = 1,$$
$$u_{init_0}(x) = 0, \quad u_t(x,0) = u_{init_1}(x) \equiv -r(-\varphi_{1xx} + \varepsilon\varphi_1), \quad x \in (0,\pi),$$

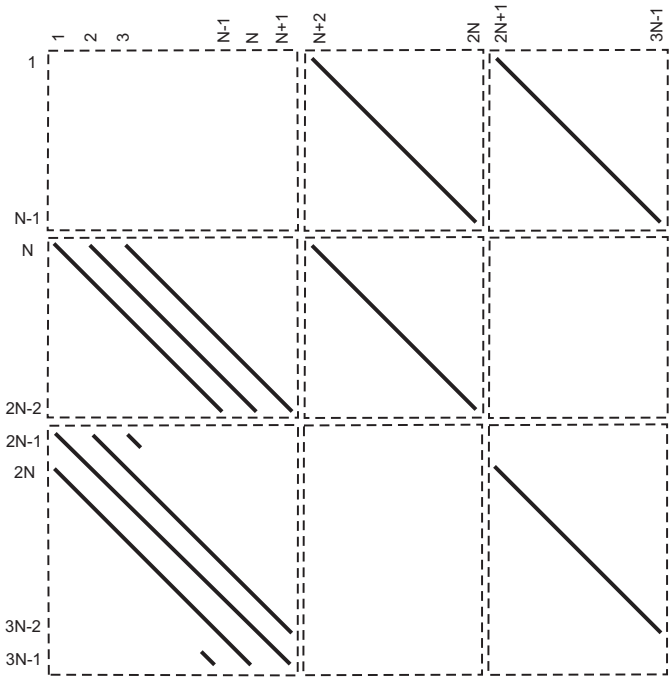

Рис. 3.9: Структура матрицы СЛАУ (3.12).

где

$$r = \frac{4\int\limits_0^\pi \varphi_1 dx + \frac{37}{3}\int\limits_a^b \frac{(\varphi_{1x})^2}{\varphi_1}dx}{\int\limits_0^\pi (-\varphi_{1xx} + \varepsilon\varphi_1)^2 dx},$$

$$\varphi_1 = \begin{cases} 0, & \text{если } x \in [0, a), \\ \sin^5 \pi \dfrac{x - A}{B - A}, & \text{если } x \in [a, b], \\ 0, & \text{если } x \in (b, \pi], \end{cases}$$

причём $\varepsilon = 0.2$, $A = 0.1\pi$, $B = 0.9\pi$.

Какое значение эффективного порядка точности наблюдается после разрушения? (Примените формулу (2.18).) Проведите соответствующую классификацию особенности. После этого с помощью вычисления эффективного порядка в каждом узле (формула (2.19)) на нескольких первых слоях, где предыдущая формула дала результат, существенно отличный от теоретического, изучите пространственную структуру разрушения.

Задача 2. Решите численно начально-краевую задачу

$$\begin{cases} \dfrac{\partial^2}{\partial t^2}\left(u_{xx} - e^{\varepsilon u}\right) + u_{xx} = 0, \quad x \in (a,b), \quad t \in (t_0, T], \\ u(a,t) = 0, \quad u_x(a,t) = 0, \quad t \in (t_0, T], \\ u(x,t_0) = u_{init_0}(x), \ x \in [a,b], \\ u_t(x,t_0) = u_{init_1}(x), \ x \in [a,b], \end{cases} \tag{3.14}$$

для следующего набора параметров:

$$t_0 = 0, \quad a = 0, \quad b = \pi, \quad T = 1,$$
$$u_{init_0}(x) = 0, \quad u_t(x,0) = u_{init_1}(x) \equiv -r(-\varphi_{1xx} + \varepsilon\varphi_1), \quad x \in (0,\pi),$$

где

$$r = \frac{4\int\limits_0^\pi \varphi_1 dx + \frac{37}{3}\int\limits_a^b \frac{(\varphi_{1x})^2}{\varphi_1}dx}{\int\limits_0^\pi (-\varphi_{1xx} + \varepsilon\varphi_1)^2 dx},$$

$$\varphi_1 = \begin{cases} 0, & \text{если } x \in [0,a), \\ \sin^3 \pi \dfrac{x-A}{B-A}, & \text{если } x \in [a,b], \\ 0, & \text{если } x \in (b,\pi], \end{cases}$$

причём $\varepsilon = 0.2$, $A = 0.1\pi$, $B = 0.9\pi$.

Какое значение эффективного порядка точности наблюдается до и после разрушения? (Примените формулу (2.18).) С чем может быть связана малость величины эффективного порядка?

Теперь с помощью вычисления эффективного порядка в каждом узле (формула (2.19)) на нескольких первых слоях, где предыдущая формула дала отрицательный эффективный порядок, изучите пространственную структуру разрушения.

*Задача 3**. Решите численно начально-краевую задачу

$$\begin{cases} \dfrac{\partial}{\partial t}\Big(u_{xx} + \big(|u_x|^{p-2}u_x\big)_x\Big) + |u|^q u = 0, \quad x \in (a,b), \quad t \in (t_0, T], \\ u(a,t) = 0, \quad u(b,t) = 0, \quad t \in (t_0, T], \\ u(x,t_0) = u_{init_0}(x), \ x \in [a,b], \end{cases} \tag{3.15}$$

для следующего набора параметров:

$$t_0 = 0, \quad a = 0, \quad b = \pi, \quad T = 2, \quad p = 3, \quad q = 2,$$
$$u_{init_0}(x) = \sin x.$$

Какое значение эффективного порядка точности наблюдается после разрушения? (Примените формулу (2.18).) С помощью вычисления эффективного порядка в каждом узле (формула (2.19)) на нескольких первых слоях, где предыдущая формула дала результат, существенно отличный от теоретического, изучите пространственную структуру разрушения.

Литература

[1] *Альшин А. Б., Альшина Е. А., Калиткин Н. Н., Корягина А. Б.* Схемы Розенброка с комплексными коэффициентами для жёстких и дифференциально-алгебраических систем // Журнал вычислительной математики и математической физики. 2006. 46, № 8. С. 1392–1414.

[2] *Альшина Е. А., Калиткин Н. Н., Корякин П. В.* Диагностика особенностей точного решения при расчетах с контролем точности // Журнал вычислительной математики и математической физики. 2005. 45, № 10. С. 1837–1847.

[3] *Калиткин Н. Н.* Численные методы решения жестких систем // Математическое моделирование. 1995. 7. № 5. Р. 8-11.

[4] *Калиткин Н. Н., Альшина Е. А.* Численные методы: в 2 кн. Кн. 1. Численный анализ: учебник для студ. учреждений высш. проф. образования // М.: Издательский центр «Академия», 2013.

[5] *Калиткин Н. Н., Корякин П. В.* Численные методы: в 2 кн. Кн. 2. Методы математической физики: учебник для студ. учреждений высш. проф. образования // М.: Издательский центр «Академия», 2013.

[6] *Калиткин Н. Н., Альшин А. Б., Альшина Е. А., Рогов Б.В.* Вычисления на квазиравномерных сетках // М.: Физматлит, 2005.

[7] *Корпусов М. О., Панин А. А.* Локальная разрешимость и разрушение решения для уравнения Бенджамена-Бона-Махони-Бюргерса с нелокальным граничным условием // Теоретическая и математическая физика. 2013. 175, № 2. С. 159–172.

[8] А. Г. Свешников, А. Б. Альшин, М. О. Корпусов, Ю. Д. Плетнер, *Линейные и нелинейные уравнения соболевского типа*, ФИЗМАТЛИТ, М., 2007.

[9] *Alshin A. B., Alshina E. A.* Numerical diagnosis of blow-up of solutions of pseudoparabolic equations // Journal of Mathematical Sciences. 2008. 148, № 1. C. 143–162.

[10] *Hairer E.,Wanner G.* Solving of Ordinary Differential Equations. Stiff and Differential-Algebraic Problems. Springer, 2002.

[11] *Rosenbrock H. H.* Some general implicit processes for the numerical solution of differential equations // Computer Journal. 1963. 5. № 4. P. 329-330.

[12] *Korpusov M. O., Lukyanenko D. V., Panin A. A., Yushkov E. V.* Blow-up for one Sobolev problem: theoretical approach and numerical analysis // Journal of Mathematical Analysis and Applications. 2016. 442. № 2. P. 451-468.