В. М. Волков, О. Л. Зубко, И. Н. Катковская,
И. Л. Ковалева, В. Г. Кротов, П. Лима

ЧИСЛЕННЫЙ АНАЛИЗ И ОПТИМИЗАЦИЯ

Минск
2017

УДК 517.5, 519.6

Рецензенты:

кафедра математических проблем управления и информатики УО Гомельского государственного университета им. Франциска Скорины (заведующий кафедрой доктор технических наук, профессор В.С. Смородин),

кандидат технических наук, доцент кафедры автоматики и компьютерных систем ФГАОУ ВО Национального исследовательского томского политехнического университета Е.А. Кочегурова.

Учебное пособие подготовлено в рамках Международного образовательного проекта TEMPUS-ACES «Прикладная компьютерная обработка данных в науке и инженерии» для магистрантов, обучающихся по специальности 1-31 81 12 «Прикладной компьютерный анализ данных».

Содержит краткое изложение методов численного анализа задач для обыкновенных дифференциальных уравнений и уравнений с частными производными, а также методов условной и безусловной оптимизации. Пособие ориентировано на освещение практических вопросов решения дифференциальных и оптимизационных задач, содержит большое число примеров.

В. М. Волков

К83 Численный анализ и оптимизация / В. М. Волков, О. Л. Зубко, И. Н. Катковская, И. Л. Ковалева, В. Г. Кротов, П. Лима. — Минск : РУП «Белгослес», 2017. — 207 с.

Оглавление

Введение

Строгость математических понятий и утверждений представляется наиболее привлекательным атрибутом математического аппарата с точки зрения его использования в качестве универсального языка науки. Описания закономерностей реального мира на языке математических абстракций и формулировка задачи в виде математической модели, учитывающей основные закономерности поведения объекта, позволяет получать новые знания об этом объекте исключительно математическими методами без необходимости непосредственного контакта с ним. Например, широкий круг явлений электродинамики может быть всесторонне изучен на основе анализа уравнений Максвелла. Аналогичным образом, глубокое теоретическое осмысление закономерностей гидродинамических течений доступно посредством анализа уравнений Навье–Стокса. Данный метод теоретических исследований, в основе которого математическая формулировка задачи и использование адекватного математического аппарата для ее решения, получил название математическое моделирование.

В последние полвека успехи прикладных математических методов в различных областях науки и техники во многом обязаны интенсивному использованию техники численного анализа. Благодаря численным методам математический язык стал еще более универсальным и обстоятельным. С другой стороны, развитие компьютерных технологий сделало этот язык более простым и доступным широкому кругу исследователей в различных областях знаний.

В настоящее время компьютерные технологии приобрели статус третьей силы в арсенале научных методов исследования, дополняя и расширяя возможности традиционных экспериментальных и теоретических подходов. Эффективность компьютерного моделирования достигается благодаря численным методам, открывающим широчайшие возможности приближенного анализа математических моделей. Техника численного эксперимента ассоциируются с решением масштабных проблем, таких как прогноз погоды, расчет орбит космических аппаратов, аэродинамика сверхзвуковых скоростей, безопасность ядерных реакторов, оптимизация сложных технических систем и др., для которых другие подходы оказываются мало эффективными или вовсе неприменимы.

Предлагаемое учебное пособие содержит краткое изложение методов численного анализа начальных и краевых задач для обыкновенных дифференциальных уравнений и уравнений в частных производных, а также методов условной и безусловной оптимизации в евклидовых пространствах. Пособие ориентировано на освещение практических вопросов, содержит большое число примеров и предназначено для студентов инженерных специальностей второй ступени обучения.

Глава 1

Численные методы решения задачи Коши

1.1. Постановка задачи

В общем случае дифференциальная задача для обыкновенных дифференциальных уравнений может быть сформулирована для системы уравнений первого порядка

$$\frac{du_k}{dt} = f_k(t, u_1, u_2, \ldots, u_N), \quad k = 1, \ldots, N, \tag{1.1}$$

где $u_k = u_k(t)$ — искомые функции одной переменной, а f_k — заданные функции $N+1$ переменных.

Примером дифференциальной задачи является система уравнений движения пружинного маятника :

$$\frac{dz}{dt} = v_z, \quad \frac{dv_z}{dt} = -\omega^2 z, \tag{1.2}$$

$$z(t_0) = z^0, \quad v_z(t_0) = v_z^0. \tag{1.3}$$

Здесь z и v_z — вертикальная координата и скорость груза, точка $z = 0$ соответствует положению равновесия, в которой сила упругости взаимно компенсируется с силой тяжести, постоянная ω — циклическая частота колебаний маятника,

$$\omega = \sqrt{\frac{k}{m}}$$

k — коэффициент упругости пружины, m — масса груза, подвешенного на пружине.

Решение дифференциальной задачи не может быть однозначно определено уравнениями (1.2). Интуитивно понятно, что положение маятника в момент времени $t = t_1$ не может быть установлено с полной определенностью при отсутствии информации о его положении и скорости в некоторый предыдущий момент времени $t_0 < t_1$. В рассматриваемом примере такие дополнительные **начальные условия** заданы уравнениями (1.3).

Таким образом, для однозначного определения решения дифференциальной задачи система дифференциальных уравнений (1.2) должна быть дополнена условиями, например, в виде алгебраических уравнений, определяющих значения искомых функций в одной и той же или различных точках на оси независимой переменной. Такие дополнительные условия принято называть начальными условиями, когда значения искомых функций задается в одной точке, или краевыми условиями в противном случае.

Задачи с начальными условиями (**задачи Коши**) широко используются для моделирования динамических систем. Существование и единственность решения задачи Коши ассоциируется с принципом детерминизма. Как правило, если уравнения и начальные условия сформулированы корректно на физическом уровне строгости, то и с математической точки зрения задача также является корректной в смысле существования и единственности решения. Число дополнительных начальных условий для однозначного определения решения обычно совпадает с количеством уравнений системы.

В дальнейшем мы будем использовать постановку задачи Коши для системы ОДУ первого порядка:

$$\frac{d\mathbf{u}}{dt} = \mathbf{f}(t, \mathbf{u}), \quad t \in [0, T], \tag{1.4}$$

$$\mathbf{u}(0) = \mathbf{u_0}, \tag{1.5}$$

где

$$\mathbf{u} = \mathbf{u}(t) = [u_1(t), u_2(t), \dots u_N(t)]^T,$$

$$\mathbf{f}(t, \mathbf{u}) = [f_1, f_2, \dots, f_N]^T,$$

$$f_k = f_k(t, u_1(t), u_2(t), \dots u_N(t)), \quad k = 1, \dots, N,$$

$$\mathbf{u}_0 = [u_1^0, u_2^0, \dots, u_N^0]^T.$$

В простейшем случае задача Коши включает одно уравнение и соответствующее начальное значение искомой функции при $t = 0$.

Упражнение 1.1.

1. Сформулируйте задачу Коши для уравнений (1.2), (1.3) в эквивалентном виде для одного дифференциального уравнения второго порядка.

2. Сформулируйте следующую задачу Коши

$$-\frac{du^3}{dt^3} + \frac{d^2u}{dt^2} + \frac{du}{dt} = f(t, u), \quad t \in [t_0, T], \tag{1.6}$$

$$\frac{d^2u}{dt^2}\,|_{t=t_0} = g_2, \quad \frac{du}{dt}\,|_{t=t_0} = g_1, \quad u(t)\,|_{t=t_0} = g_0, \tag{1.7}$$

в виде эквивалентной системы дифференциальных уравнений первого порядка.

1.2. Метод Эйлера. Явные и неявные схемы. Понятие устойчивости

Метод Эйлера — самый элементарный метод численного интегрирования задачи Коши для обыкновенных дифференциальных уравнений. Схема данного метода, например, может быть получена как следствие приближенного представления решения дифференциальной задачи ((1.4)), ((1.5)) отрезком степенного ряда :

$$u(t_0+\tau) = u(t_0) + \tau u'(t_0) + \frac{\tau^2}{2!}u''(t_0) + O(\tau^3). \tag{1.8}$$

При подстановке $u'(t_0) = f(t_0, u(t_0))$ в уравнение (1.8), пренебрегая членами высшего порядка малости, мы приходим к следующему уравнению для приближенного решения задачи (1.4),(1.5):

$$u(t_0+\tau) \simeq u(t_0) + \tau f(t_0, u(t_0)). \tag{1.9}$$

Таким образом, нам известно значение искомого решения задачи (1.4)–(1.5) при $t = t_0$, и мы можем вычислить приближенное значение искомого решения при $t = t_0 + \tau$ согласно (1.9). Применяя формулу (1.9), шаг за шагом, мы приходим к рекурсивному алгоритму численного интегрирования задачи Коши, (1.4)–(1.5), который позволяет вычислить приближенное решение для произвольного $t \in [0, T]$:

$$U(t_{k+1}) = U(t_k) + \tau f(t_k, U(t_k)), \quad t_k = k\tau, k = 0, 1, 2, \ldots. \tag{1.10}$$

Формула (1.10) известна как **метод Эйлера**,где τ — **шаг численного интегрирования**.

Заметим, что решение в методе Эйлера вычисляется непосредственно по явной формуле. Данный класс алгоритмов принято называть **явными**.

Мы будем также называть метод Эйлера **одношаговым**, подчеркивая тем самым то, что для вычисления нового значения U_{k+1} при $t = t_{k+1}$ мы используем значение решения только в одной предыдущей точке при $t = t_k$ (на дистанции одного шага τ от искомого решения в новой точке).

Вычислительная сложность алгоритмов решения дифференциальных задач принято характеризовать количеством вычислений функций $f(t, u)$ на одном шаге численного интегрирования. Для метода Эйлера на каждом шаге требуется однократное вычисление $f(t, u)$.

Отличие точного решения $u(t_{k+1})$ и его приближенного значения (1.10) (при условии, что $u(t_k)$ определено точно) называется **локальной погрешностью** , т.е. погрешностью на одном шаге численного интегрирования:

$$\delta(t_k+\tau) = U(t_{k+1}) - u(t_{k+1}) = \frac{\tau^2}{2!}u''(t_k) + O(\tau^3). \tag{1.11}$$

Для определения глобальной погрешности при произвольном значении $t = T$ мы должны учитывать эффект накопления ошибки, обусловленный рекурсивной структурой алгоритма. Кроме того, необходимо убедиться, что алгоритм устойчив и локальная ошибка, полученная на одном шаге, не испытывает неограниченного роста на последующих шагах.

Устойчивость численных методов для решения линейной задачи Коши (1.4), (1.5) можно определить требованием выполнения оценки

$$|U_k| \leqslant C_1|U_0\| \ + C_2 \max_{0<m<k-1} |f_m|, \tag{1.12}$$

где C_1 и C_2 — положительные постоянные, не зависящие от τ, $U_k = U(t_k)$.

Неравенство (1.12) означает, что приближенное решение непрерывно зависит от входных данных: малые возмущения начальных условий и правой приводят к ограниченным отклонениям траектории решения на произвольном отрезке $t \in [0, T]$.

Для исследования устойчивости методов численного решения задачи Коши, как правило, рассматривается однородная тестовая задача:

$$\frac{du}{dt} = \lambda u, \quad \lambda < 0. \tag{1.13}$$

Заметим, что решение уравнения (1.13) при отрицательном значении λ является ограниченным и устойчивым:

$$u(t) = u(0)\exp(-|\lambda|t). \tag{1.14}$$

Приближенное решение данной задачи при использовании метода Эйлера имеет вид:

$$U_{k+1} = (1 - \tau|\lambda|)U_k = (1 - \tau|\lambda|)^k U_0. \tag{1.15}$$

Легко видеть, что решение (1.15) будет ограниченным и устойчивым при выполнении условия

$$|1 - \tau|\lambda|| \leqslant 1. \tag{1.16}$$

Последнее неравенство можно рассматривать как **условие устойчивости** метода Эйлера. Как следует из (1.16), метод Эйлера является устойчивым для тестовой задачи (1.13) при условии

$$\tau < 2/|\lambda|. \tag{1.17}$$

Численный метод, который устойчив при выполнении некоторых ограничений на шаги дискретизации, называется **условно устойчивым**. Если для устойчивости метода никаких ограничений на шаги сетки не требуется, то такой метод называют **безусловно устойчивым**.

В общем случае системы линейных дифференциальных уравнений вида

$$\frac{d\mathbf{u}}{dt} = A\mathbf{u}, \quad \mathbf{u} \in R^N, \quad A \in R^{N\times N}, \tag{1.18}$$

условие устойчивости имеет вид, аналогичный (1.17), где вместо $|\lambda|$ следует использовать максимальное по модулю собственное значение или норму матрицы A.

В качестве примера безусловно устойчивого метода решения задачи Коши отметим так называемый **неявный метод Эйлера**:

$$U(t_{k+1}) = U(t_k) + \tau f(t_{k+1}, U(t_{k+1})), \quad t_k = k\tau, k = 0, 1, 2, \ldots. \tag{1.19}$$

где в отличие от (1.10) решение выражается неявно и может быть вычислено посредством решения соответствующего, вообще говоря нелинейного, уравнения или системы уравнений. Применяя метод (1.19) к решению тестовой задачи (1.13), имеем

$$U_{k+1} = (1+\tau\lambda)^{-1}U_k = (1+\tau\lambda)^{-k}U_0. \tag{1.20}$$

Очевидно, что для любого $\tau > 0$ и $\lambda > 0$:

$$|U_{k+1}| \leqslant |U_0|, \tag{1.21}$$

и, согласно (1.12), неявный метод Эйлера является безусловно устойчивым.

Будем говорить, что численный метод сходится и имеет n-й **порядок точности** (**скорость сходимости** $O(\tau^n)$), если глобальная погрешность данного метода убывает пропорционально τ^n: $\delta_k = O(\tau^n)$.

Несложно показать, что глобальная погрешность метода Эйлера стремится к нулю при $\tau \to 0$:

$$|\delta_k| \leqslant C\tau, \quad k = 1, 2, \ldots. \tag{1.22}$$

Здесь C — постоянная, не зависящая от τ.

Как следует из оценки (1.22), метод Эйлера (1.4)–(1.5) сходится, имеет первый порядок точности и его погрешность убывает пропорционально τ. Эффект накопления ошибки приводит к тому, что глобальная ошибка на порядок превосходит локальную погрешность. Так, если локальная погрешность имеет порядок $O(\tau^p)$, то глобальная погрешность при условии устойчивости алгоритма, как правило, ограничена величиной $O(\tau^{p-1})$. В силу низкой скорости сходимости и условной устойчивости явный метод Эйлера имеет слабую эффективность и практически не используются для решения реальных задач.

Пример 1.1. Рассмотрим пример, демонстрирующий типичные проявления эффекта потери устойчивости, когда условия устойчивости оказываются нарушенными. Для этого рассмотрим явный метод Эйлера применительно к модельному уравнению вида (1.13), $\lambda = 1.5$ с начальными условиями $u(0) = 2$. Программная реализация и результаты численных экспериментов представлены ниже. Несложно видеть, что при $\tau < \tau_0/2$, где τ_0 опрелено условием устойчивости (1.17), приближенное решение мало отличается от точного. В случае $\tau_0/2 < \tau < \tau_0$ приближенное решение теряет свойство монотонности, но остается асимптотически ограниченным и устойчивым. Наконец, когда $\tau > \tau_0$ приближенное решение перестает сходится к стационарному значению из-за потери устойчивости. Продолжая вычисления, легко убедиться, что неустойчивость приводит к неограниченному росту решения, в то время как точное решение стремится к нулю при $t \to 0$.

```
%%% Устойчивость & Сходимость
%%% метод Эйлера для уравнения u'=-lambda*u
lambda = 1.5;
T = 10.
U0 = 2;
tau_0 = 2/lambda;
NN = round(T/tau_0)*10;
u = zeros(3,NN);
t = u;
n = 0;
t(:,1) = 0;
```

```
u(:,1) = U0;
for tau = [0.2, 0.8 1.1]*tau_0
n = n+1;
N = T/tau;
U = U0;
for m = 1:N-1
U = (1-tau*lambda)*U;
u(n,m+1) = U;
t(n,m+1) = tau*m;
end
NN(n) = N;
end
plot(t(1,1:NN(1)),u(1,1:NN(1)),'.-',...
t(2,1:NN(2)),u(2,1:NN(2)),'o-',...
t(3,1:NN(3)),u(3,1:NN(3)),'o-')
legend('\tau=0.2*\tau_0',...
'\tau=0.8*\tau_0','\tau=1.1*\tau_0')
xlabel('t')
ylabel('U(t)')
grid
```

Типичная картина развития неустойчивости представлена на Рис. 1.1. Аналогичное поведение решения характерно для большинства явных численных методов при использовании шага дискретизации, не удовлетворяющего условиям устойчивости. Отметим, что существуют также абсолютно неустойчивые методы, сохраняющие подобное поведение при произвольном, сколь угодно малом значении шага численного интегрирования.

Упражнение 1.2.

1. Исследовать устойчивость двухстадийного численного метода решения задачи Коши для уравнения (1.13), состоящего в чередовании явной и неявной формул Эйлера (1.10) и (1.19) на нечетных и четных шагах соответственно.

2. Оцените локальную погрешность неявного метода Эйлера (1.19) и метода, рассмотренного в предыдущем упражнении.

3.Пусть задачи Коши была решена численно трижды с различными значениями шага численного интегрирования: $\tau = \tau_0$; $\tau = \tau_0/2$; и $\tau = \tau_0/4$, где τ_0 достаточно мало. Оцените погрешность метода и его порядок точности, используя упомянутые выше три приближенные решения, полагая асимптотическое поведение погрешности $|\delta| \simeq C\tau^p$ и считая точное решение неизвестным.

4. Воспроизведите численный эксперимент предыдущего пункта 3, используя численный алгоритм, рассмотренный в примере выше, используя $\tau_0 = 0.01$. Оцените фактическую погрешность численного метода, используя точное решение задачи и сравните результат с оценкой (1.22), а также с оценкой, полученной в предыдущем упражнении 3.

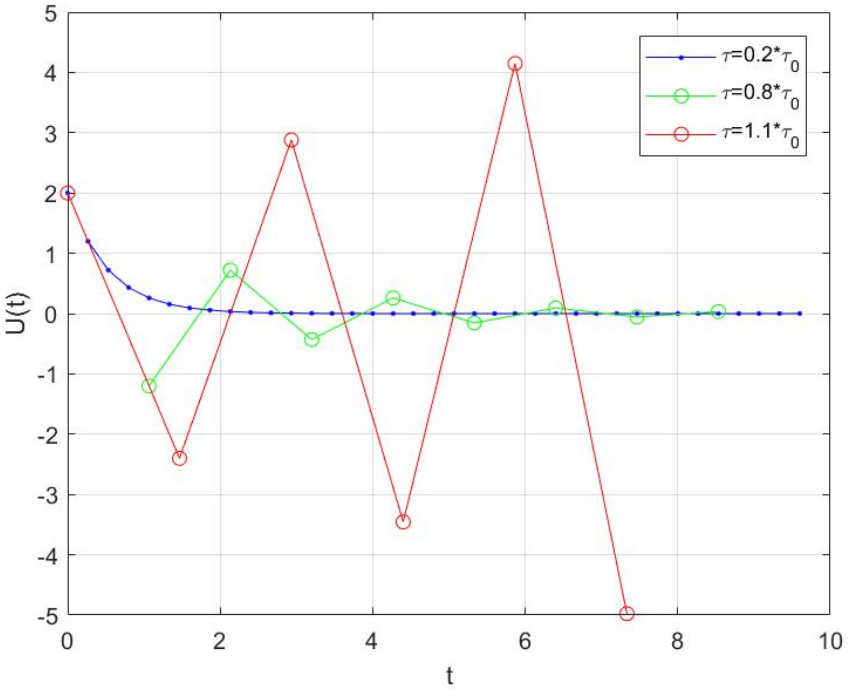


Рис. 1.1. Потеря устойчивости явного метода Эйлера при нарушении условия устойчивости (1.17)

1.3. Методы Рунге – Кутты

Методы Рунге – Кутты представляют собой семейство методов численного анализа задач Коши для систем обыкновенных дифференциальных уравнений вида (1.4)–(1.5). Наибольшей распространение в практике вычислений получили явные одношаговые **методы Рунге – Кутты** следующего вида:

$$U_{k+1} = U_k + \sum_{m=1}^{s} b_m K_m, \tag{1.23}$$

где s — целая постоянная, определяющая количество стадий вычислений, связанных с пересчетом функции правой части, в пределах одного шага численного интегрирования,

$$\begin{aligned} &K_1 = \tau f(t_k,\, U_k),\\ &K_2 = \tau f(t_k + c_2\tau,\, U_k + a_{21}K_1),\\ &\ldots\ldots\ldots\ldots\ldots\\ &K_s = \tau f(t_k + c_s\tau,\, U_k + \sum_{i=1}^{s-1} a_{si}K_i). \end{aligned} \tag{1.24}$$

Коэффициенты b_m, c_m и a_{km} определяются из условия максимального поряд-

ка малости локальной погрешности для заданного числа стадий. Для наглядности коэффициенты методов Рунге – Кутты удобно представлять в виде **таблицы Бутчера** (в честь Дж. Бутчера (John C. Butcher)):

$$\begin{array}{c|c} c & A \\ \hline & b^T \end{array} = \begin{array}{c|cccccc} 0 & 0 & & & & & \\ c_2 & a_{21} & 0 & & & & \\ \vdots & \vdots & \ddots & & & & \\ c_s & a_{s1} & a_{s2} & \dots & a_{ss-1} & 0 \\ \hline & b_1 & b_2 & \dots & \dots & b_s \end{array} \tag{1.25}$$

Примечательно, что условие максимального порядка малости локальной погрешности не обеспечивает однозначности определения коэффициентов метода Рунге – Кутты. Пример коэффициентов наиболее популярного метода Рунге – Кутты четвертого порядка точности представлен ниже в таблице:

$$\begin{array}{c|c} c & A \\ \hline & b^T \end{array} = \begin{array}{c|cccc} 0 & 0 & 0 & 0 & 0 \\ \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 \\ \frac{1}{2} & 0 & \frac{1}{2} & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ \hline & \frac{1}{6} & \frac{1}{3} & \frac{1}{3} & \frac{1}{6} \end{array} \tag{1.26}$$

Несложно заметить, что для $s = 1$ одностадийный метод Рунге – Кутты полностью совпадает с методом Эйлера. Следует отметить, что порядок точности методов Рунге – Кутты (1.23)–(1.24) при $1 \leqslant s \leqslant 4$ совпадает с количеством стадий. Для более сложных схем, $s > 4$, порядок точности p ($|\delta_k| = |U_k - u(t_k)| = O(\tau^p)$) не превосходит s и увеличивается с увеличением количества стадий как показано в таблице ниже.

s	1	2	3	4	5	6	7	8	9	10
p	1	2	3	4	4	5	6	6	7	7

(1.27)

Вычислительная сложность метода Рунге – Кутты зависит от количества вычислений функции $f(t, u)$ и растет пропорционально числу стадий s, поскольку на

каждой стадии значение функции вычисляется один раз. Учитывая не всегда пропорциональную зависимость порядка точности от числа стадий, можно сделать вывод, что наиболее интенсивный рост эффективности метода Рунге – Кутты происходит до четвертого порядка включительно, после чего вычислительная сложность растет быстрее, нежели порядок точности (см.таблицу 1.27) . Этим, в частности, можно объяснить наибольшую популярность среди методов Рунге – Кутты именно четырех-этапной версии, обеспечивающей соответственно четвертый порядок точности.

Информация относительно порядка точности численного метода может быть использована для апостериорной оценки фактической погрешности приближенного решения. В частности, скорость сходимости определяет асимптотическое поведение погрешности решения при $\tau \to 0$:

$$|\delta(t_k)| = O(\tau^p) \;\Rightarrow \delta(t_k) \simeq C\tau^p, \tag{1.28}$$

где C — постоянная не зависящая от τ.

Имея два приближенных решения $U_{(1)}(t_k)$ и $U_{(2)}(t_{2k})$, полученные при использовании различных шагов интегрирования, например, $\tau_1 = \tau_0$ и $\tau_2 = \tau_0/2$, мы можем приближенно выразить погрешность следующим образом:

$$U_{(1)}(t_k) \simeq u(t_k) + C\tau_0^p, \quad U_{(2)}(t_{2k}) \simeq u(t_{2k}) + 2^{-p}C\tau_0^{-p},$$

$$\delta_{(2)}(t_{2k}) \simeq \frac{U_{(1)}(t_k) - U_{(2)}(t_{2k})}{2^p - 1}. \tag{1.29}$$

Уравнение (1.29) при достаточно малом значении τ_0 дает реалистичную оценку погрешности приближенного решения при $\tau = \tau_0/2$.

Упражнение 1.3.

1. Докажите, что приближенное решение метода Рунге–Кутты четвертого порядка точности (1.25), (1.26) для модельной задачи

$$\frac{du}{dt} = \lambda u, \quad u(0) = u_0, \tag{1.30}$$

может быть представлено в следующем виде:

$$U(t_k) = u_0 \left(1 + \tau\lambda + \frac{(\tau\lambda)^2}{2!} + \frac{(\tau\lambda)^3}{3!} + \frac{(\tau\lambda)^4}{4!}\right)^k, \tag{1.31}$$

где выражение в скобках есть отрезок степенного ряда разложения функции $\exp(k\tau\lambda)$.

2. Оцените величину погрешности метода Рунге – Кутты четвертого порядка точности используя формулы (1.31) и (1.29) при $\tau_0 = 0.02$, $\tau_0 = 0.01$, $\tau_0 = 0.005$ применительно к тестовой задаче (1.30) при $\lambda = 1$, $t = 1$. Сравните полученную оценку погрешности с фактической погрешностью $\delta(1) = U(1) - u(1)$, используя точное решение тестовой задачи $u(t) = u_0 \exp(\lambda t)$.

3. Построить график функции

$$W_{RK}(s) = \left|1 + s + \frac{s^2}{2!} + \frac{s^3}{3!} + \frac{s^4}{4!}\right|, \tag{1.32}$$

которая может быть использована в качестве модуля функции передачи для расчета приближенного решения методом Рунге - Кутты четвертого порядка точности на одном шаге, применительно к решению уравнения (1.30), $s = \tau\lambda$ (см. выражение (1.31)). Используя утверждение упражнения 1., определите область устойчивости метода Рунге – Кутты четвертого порядка точности, т.е. область, где $|W_{RK}(s)| \leqslant 1$ при $\lambda < 0$.

4 Сравните функцию передачи (1.32), и функцию передачи метода Эйлера (1.15), модуль которой

$$|W_E(s)| = |1 + s| . \tag{1.33}$$

Какой из методов обеспечивают большую область устойчивости и допускает использование больших шагов τ.

1.4. Многошаговые методы

Мы рассмотрели семейство методов Рунге – Кутты, относящихся к классу одношаговых явных методов. Гибкость в использовании и достаточно высокая эффективность применительно к широкому кругу дифференциальных задач является бесспорным достоинством данного семейства методов. Тем не менее, увеличение скорости сходимости выше четвертого порядка в методах Рунге – Кутты сопряжено с некоторым дополнительным ростом вычислительных затрат (см. зависимость порядка точности от числа стадий в методе Рунге – Кутты (1.27)). Эффективность одношаговых методов Рунге – Кутты может существенно ухудшится, если вычисление функции правой части системы дифференциальных уравнений требуе больших вычислительных затрат. Такого рода недостатки можно преодолеть, используя многошаговый подход к построению дискретной модели дифференциальной задачи.

Линейные **многошаговые методы** для решения задачи Коши (1.4)–(1.5) имеет следующий общий вид:

$$U_k + a_1 U_{k-1} + \ldots + a_m U_{k-m} = \tau \left[b_0 f_k + b_1 f_{k-1} + \ldots + b_m f_{k-m} \right], \tag{1.34}$$

где $f_k = f(t_k, U_k)$, $U_k = U(t_k)$, а значения коэффициентов a_n и b_n определяются из условий минимума локальной погрешности метода. Максимальный порядок точности, который формально может быть получен в рамках m-шаговой схемы (1.34), достигает $O(\tau^{2m})$. Тем не менее, многошаговые схемы максимального порядка точности оказываются абсолютно неустойчивыми и непригодными для практического использования. Приемлемые условия устойчивости могут быть достигнуты лишь для многошаговых методов с порядком точности $O(\tau^m)$ при использовании явной схемы метода, или на один (два) порядка выше для неявных методов с нечетным (четным) значением числа шагов m.

В качестве многошаговых методов, способных обеспечивать устойчивость, можно отметить семейство **методов Адамса**,

$$U_k - U_{k-1} = \tau \left[b_0 f_k + b_1 f_{k-1} + \ldots + b_m f_{k-m} \right]. \tag{1.35}$$

среди которых явные методы **Адамса–Башфорта** , для которых $b_0 = 0$, и неявные **методы Адамса – Моултона** ($b_0 \neq 0$). Максимальный порядок точности m-шагового метода Адамса – Башфорта и Адамса– Моултона достигает $O(\tau^m)$ и $O(\tau^{m+1})$ соответственно. Простейшим примером метода Адамса–Башфорта может служить одношаговый явный метод Эйлера ($b_0 = 0,\ b_1 = 1$).

Методы Адамса – Башфорта более высокого порядка точности с оценками главного члена локальной ошибки δ_u имеют вид:

$$U_k = U_{k-1} + \tau V_k, \tag{1.36}$$

где

$$V_k = \left[\frac{3}{2}f_{k-1} - \frac{1}{2}f_{k-2}\right], \quad \delta_u = \frac{5\tau^3}{12}u''',$$

$$V_k = \left[\frac{23}{12}f_{k-1} - \frac{16}{12}f_{k-2} + \frac{5}{12}f_{k-3}\right], \quad \delta_u = \frac{9\tau^4}{24}u^{(4)},$$

$$V_k = \left[\frac{55}{24}f_{k-1} - \frac{59}{24}f_{k-2} + \frac{37}{24}f_{k-3} - \frac{9}{24}f_{k-4}\right], \quad \delta_u = \frac{251\tau^5}{720}u^{(5)},$$

$$V_k = \left[\frac{1901}{720}f_{k-1} - \frac{2774}{720}f_{k-2} + \frac{2616}{720}f_{k-3} - \frac{1274}{720}f_{k-4} + \frac{251}{720}f_{k-5}\right], \quad \delta_u = \frac{95\tau^6}{2888}u^{(6)}. \tag{1.37}$$

Глобальная ошибка этих методов на порядок больше и варьируется от второго ($m = 2$) до пятого ($m = 5$).

Методы Адамса–Моултона различного порядка точности также определяются общей формулой (1.36), где:

$$V_k = \tau\left[\frac{1}{2}f_k + \frac{1}{2}f_{k-1}\right], \qquad \delta_u = -\frac{\tau^3}{12}u''',$$

$$V_k = \tau\left[\frac{5}{12}f_k + \frac{8}{12}f_{k-1} - \frac{1}{12}f_{k-2}\right], \qquad \delta_u = -\frac{\tau^4}{24}u^{(4)},$$

$$V_k = \tau\left[\frac{9}{24}f_k + \frac{19}{24}f_{k-1} - \frac{5}{24}f_{k-2} + \frac{1}{24}f_{k-3}\right], \qquad \delta_u = -\frac{19\tau^5}{720}u^{(5)},$$

$$V_k = \tau\left[\frac{251}{720}f_k + \frac{646}{720}f_{k-1} - \frac{264}{720}f_{k-2} + \frac{106}{720}f_{k-3} - \frac{19}{720}f_{k-4}\right], \quad \delta_u = -\frac{3\tau^6}{160}u^{(6)}. \tag{1.38}$$

Несмотря на существенное сходство методов Адамса – Моултона и Адамса–Башфорта, принципиальное их отличие состоит в том, что методы Адамса – Моултона являются неявными и их решение не может быть выражено в явном виде, как в методе Адамса–Башфорта. В общем случае реализация неявных методов (1.38) приводит к решению на каждом шаге системы нелинейных алгебраических уравнений. Для этих целей обычно используются итерационные методы Ньютона или Пикара. Эффективным также представляется комбинация явных и неявных методов

Адамса в схеме получившей название **предиктор – корректор**. Схема предиктор - корректор может рассматриваться как одна итерация в неявном методе, начальное приближение для которой вычисляется по явной схеме соответствующего порядка точности.

В качестве примера рассмотрим схему предиктор – корректор, построенную на трехшаговых методах Адамса. Пусть нам известно решение задачи в точках сетки t_{k-1}, t_{k-2}, t_{k-3}. Вычисляем неизвестное значение U_k в два этапа. На первом этапе, **предиктор**, мы используем трехшаговый явный метод Адамса–Башфорта, рассматривая полученное значение U_k как промежуточный результат:

$$\tilde{U}_k = U_{k-1} + h\left[\frac{23}{12}f_{k-1} - \frac{16}{12}f_{k-2} + \frac{5}{12}f_{k-3}\right]. \tag{1.39}$$

Затем мы корректируем полученное значение, используя для этого неявную формулу Адамса – Моултона:

$$U_k = U_{k-1} + h\left[\frac{5}{12}\tilde{f}_k + \frac{8}{12}f_{k-1} - \frac{1}{12}f_{k-2}\right], \tag{1.40}$$

где для вычисления $\tilde{f}_k = f(t_k, \tilde{U}_k)$ используем значение $\tilde{U}_k$, вычисленное на стадии предиктора. Стадия корректора (1.40) позволяет не только повысить точность решения, но и улучшить устойчивость метода. Примечательная деталь: главные члены локальной погрешности одинакового порядка у явных и неявных методов Адамса имеют противоположные знаки (см. (1.37) и (1.38)). Кроме того, абсолютная величина локальной ошибки m-шаговых неявных методов (1.38)) во много раз меньше по сравнению с явными $(m+1)$-шаговыми методами (1.37).

Основное преимущество методов Адамса состоит в том, что, в отличие от методов Рунге – Кутты, функция правой части задачи на каждом шаге вычисляется всего лишь один раз, независимо от порядка точности метода. Как следствие, вычислительная сложность многошаговых методов практически не зависит от порядка точности, что выгодно отличает их от одношаговых аналогов. Так, например, в методе Рунге – Кутты четвертого порядка точности функция правой части вычисляется четыре раза на каждом шаге, в то время как в явном методе Адамса можно ограничиться однократным вычислением практически для произвольного порядка точности. Таким образом, преимущества многошаговых методов становится более существенным при использовании формул более высокого порядка точности и особенно в случаях, когда вычисление функции правой части задачи (1.4)–(1.5) сопряжено со значительными вычислительными затратами.

Тем не менее, следует отметить также то, что для вычисления решения U_k в точке $t = t_k$ при использовании m-шагового метода нам необходимо знать решение в точках, $U_{k-1}, \ldots, U_{k-m}$. В начале вычислений значение U_{k-m} при $t = t_{k-m} = t_0$ определено начальными условиями. Для расчета остальных недостающих значений требуется использовать другие, например одношаговые методы типа методов Рунге – Кутты соответствующего порядка точности. Аналогичная ситуация может возникнуть при изменении шага τ в многошаговой схеме.

Для сравнения устойчивости явных и неявных многошаговых методов рассмотрим тестовую задачу

$$\frac{du}{dt} = \lambda u, \tag{1.41}$$

где λ — произвольное комплексное число. Решение задачи (1.41) экспоненциально устойчиво при $Re(\lambda) < 0$, т.е. область устойчивости дифференциальной задачи занимает левую полуплоскость комплексной плоскости. Для исследования устойчивости представим решение методов Адамса в степенном виде:

$$U_k = q^k. \tag{1.42}$$

Подстановка данного решения в формулы метода Адамса приводит к характеристическим уравнениям для величины q. Например, применение метода Адамса – Моултона второго порядка точности к решению задачи (1.42) приводит к следующему характеристическому уравнению:

$$q^k - q^{k-1} = \frac{\lambda\tau}{2}(q^k + q^{k-1}), \tag{1.43}$$

Уравнение (1.43) имеет корень

$$q = \frac{1 + \tau\lambda/2}{1 - \tau\lambda/2}. \tag{1.44}$$

Очевидно, что в области устойчивости дифференциальной задачи ($Re(\lambda) < 0$) мы имеем $|q| \leqslant 1$, и приближенное решение (1.42) также устойчиво, подобно решению дифференциальной задачи. В общем случае для оценки устойчивости используются так называемое **условие корней**: если все корни характеристического уравнения (полинома) локализованы на комплексной плоскости внутри единичного круга, $|q| \leqslant 1$, и отсутствуют кратные корни $|q| = 1$, то соответствующий многошаговый метод является устойчивым. Если условие корней не выполнено для $\lambda = 0$, то метод считается абсолютно неустойчивым. Таким образом, исследование устойчивости многошаговых методов сводится к проверке условия корней. Очевидно, что для рассмотренной тестовой задачи корни уравнения зависят от величины $\mu = \tau\lambda$. Множество точек комплексной плоскости $\mu \in C$, в которых выполняется условие корней называется **областью устойчивости метода**.

Если область устойчивости метода совпадает с областью устойчивости дифференциальной задачи, то такой метод называется **A-устойчивым**, или **абсолютно устойчивым**. К сожалению, не существует явных A - устойчивых методов. Более того, среди неявных линейных методов не существует методов выше второго порядка точности.

Если многошаговый метод не является A — устойчивым, то его область устойчивости ограничена и граница области устойчивости определяется соотношением между шагом τ и λ, вытекающим из характеристического уравнения при $|q| = 1$. Простейший способ определения границы области устойчивости состоит в том, чтобы нарисовать на комплексной плоскости множество точек $\mu = \tau\lambda$ для которых $|q| \equiv 1$. Например, в случае метода Адамса – Моултона третьего порядка точности характеристическое уравнение имеет вид

$$1 - q = \tau\lambda\left[\frac{5}{12} + \frac{8}{12}q - \frac{1}{12}q^2\right], \tag{1.45}$$

и область устойчивости определяется уравнением:

$$\mu = \tau\lambda = (1-q)\left[\frac{5}{12} + \frac{8}{12}q - \frac{1}{12}q^2\right]^{-1}, \quad q = \exp(-i\varphi), \quad \varphi \in [1, 2\pi]. \tag{1.46}$$

Области устойчивости для некоторых явных и неявных методов Адамса представлены на рис. 1.2. Несложно заметить, что область устойчивости неявных методов существенно превосходит область устойчивости явных аналогов, имеющих тот же порядок точности. С ростом порядка точности область устойчивости уменьшается.

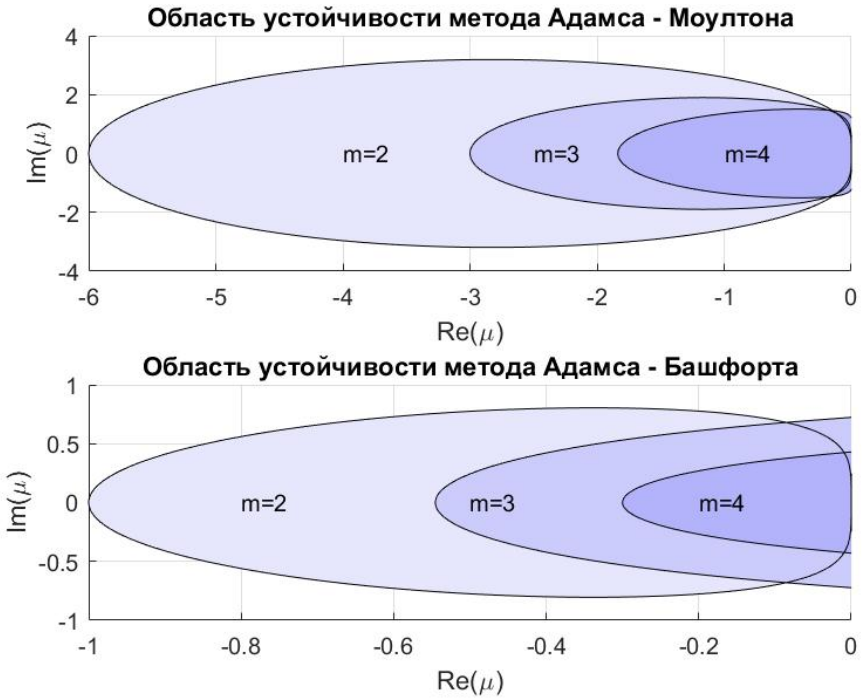


Рис. 1.2. Области устойчивости многошаговых явных и неявных методов Адамса.

Пример 1.2. Проведем сравнительный анализ эффективности методов Рунге – Кутты и многошаговых методов Адамса, оценивая вычислительные затраты для получения заданной точности. В качестве тестовой задачи мы рассмотрим уравнение (1.2), для которого известно точное решение. Сравним эффективность методов четвертого порядка точности. Программная реализация алгоритмов и результаты численных экспериментов представлены ниже (см. рис. 1.2 и 1.3).

```
%Эффекктивность методов решения ОДУ
% Метод Рунге -- Кутты 4-го порядка (РК)
% Метод Адамса -- Башфорта 4-го порядка (АБ)
```

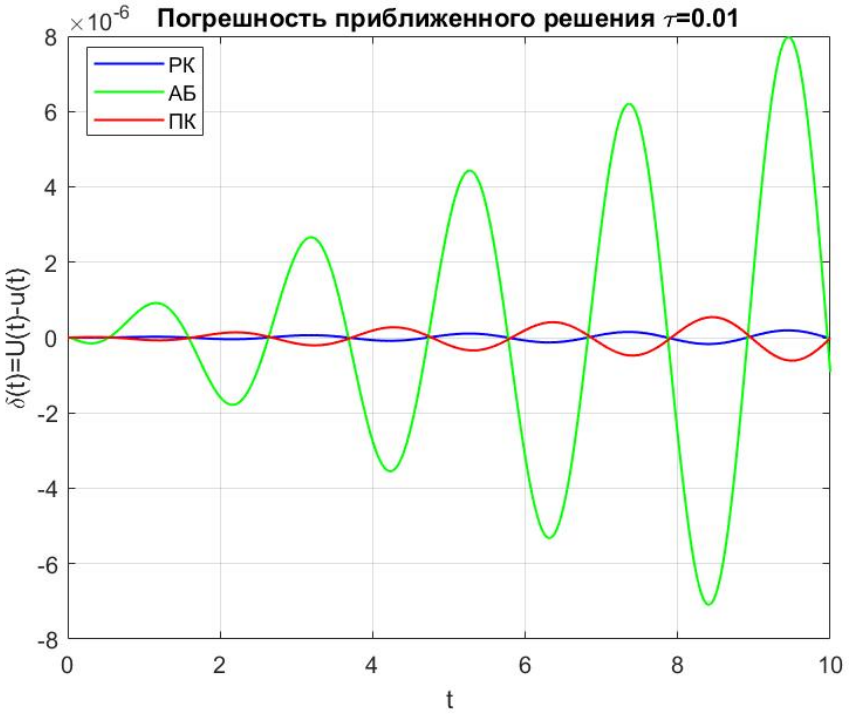


Рис. 1.3. Динамика погрешности методов четвертого порядка точности Рунге – Кутты (RK), Адамса – Башфорта (AB) предиктор – корректор (PC)

```
% Метод предиктор -- корректор 4-го порядка (ПК)
% Задача:   u''=-3u, t in [1,10];
% начальные условия: u(0)=0, u'(0)=3;
% точное решение: u(t)=sin(3t);
f = @(t,u)[u(2); -9*u(1)];
T = 10;
tau = 0.01;
u = [0;3];
N = T/tau;
ts = 0:tau:T;
U = zeros(2,N+1);
%Метод Рунге -- Кутты 4 порядка
tic
U(:,1) = u;
for m = 1:N
t = (m-1)*tau;
K1 = tau*f(t,u);
K2 = tau*f(t+tau/2,u+K1/2);
K3 = tau*f(t+tau/2,u+K2/2);
```

```
K4 = tau*f(t+tau,u+K3);
u = u+(K1+2*K2+2*K3+K4)/6;
U(:,m+1) = u;
end
RK_time = toc
B = tau*[-9/24;37/24;-59/24;55/24];
F = zeros(2,N+1);
u = [0;3];
U1 = zeros(2,N+1);
U1(:,1) = u;
F(:,1) = f(0,u);
% Метод Адамса -- Башфорта 4 порядка
tic
for m = 1:3
t = (m-1)*tau;
K1 = tau*f(t,u);
K2 = tau*f(t+tau/2,u+K1/2);
K3 = tau*f(t+tau/2,u+K2/2);
K4 = tau*f(t+tau,u+K3);
u=u+(K1+2*K2+2*K3+K4)/6;
U1(:,m+1) = u;
F(:,m+1) = f(t+tau,u);
end
for m = 4:N
u = u+F(:,m-3:m)*B;
t = m*tau;
F(:,m+1) = f(t,u);
U1(:,m+1) = u;
end
AB_time = toc
% Метод предиктор -- корректор 4-го порядка
A = tau*[1/24;-5/24;19/24;9/24;];
u2 = [0;3];
U2 = zeros(2,N+1);
U2(:,1) = u2;
F(:,1) = f(0,u2);
tic
for m=1:3
t = (m-1)*tau;
K1 = tau*f(t,u2);
K2 = tau*f(t+tau/2,u2+K1/2);
K3 = tau*f(t+tau/2,u2+K2/2);
K4 = tau*f(t+tau,u2+K3);
u2 = u2+(K1+2*K2+2*K3+K4)/6;
U2(:,m+1) = u2;
F(:,m+1) = f(t+tau,u2);
```

```
end
for m = 4:N
u22 = u2+F(:,m-3:m)*B;%предиктор
t = m*tau;
F(:,m+1) = f(t,u22);
u2 = u2+F(:,m-2:m+1)*A;%корректор
F(:,m+1) = f(t,u2);
U2(:,m+1) = u2;
end
PC_time = toc
y = sin(3*ts); % точное решение;
plot(ts,(U(1,:)-y),'b',ts,U1(1,:)-y,'g',ts,U2(1,:)-y,'r','LineWidth',1)
title(['Погрешность приближенного решения \tau=',num2str(tau,3)])
xlabel('t')
ylabel('\delta(t)=U(t)-u(t)')
legend('РК','АБ','ПК');
grid on
```

Когда вычисления заканчиваются, в командном окне можно видеть время решения задачи для каждого из рассмотренных методов, Рунге – Кутты (RK), Адамса – Башфорта (AB) предиктор – корректор (PC):

```
RK_time =  0.1174
AB_time =  0.0518
PC_time =  0.0776
```

Несложно заметить, что ни один из рассмотренных методов не имеет бесспорного преимущества. Явный метод Адамса оказывается несколько быстрее своих конкурентов, но он существенно проигрывает им в точности. Погрешности метода Адамса – Башфорта примерно в десять раз превосходит погрешность метода Адамса – Моултона (использованного на стадии корректора в PC), как это предсказывают оценки их локальных ошибок (см. (1.37), (1.38)).

Упражнение 1.4.

1. Внося необходимые коррективы в рассмотренном выше примере, замените программную реализацию методов Рунге – Кутты и Адамса – Башфорта на соответствующие стандартные функции MATLAB **ode45** и **ode15s**, в которых реализованы указанные методы. Сравните эффективность стандартной реализации рассмотренных методов, сопоставляя вычислительные затраты для достижения заданной точности.

2. Используя тестовую задачу, рассмотренную в примере выше и упражнении 1., проверьте, как значения входного параметра **RelTol** в функциях **ode45** и **ode15s** влияют на фактическую погрешностью численного решения.

3. Сравните эффективность решения задачи Коши с помощью MATLAB функций **ode45** и **ode15s** применительно к модифицированной тестовой задаче:

$$\begin{cases} \dfrac{du_1}{dt} = u_2, \\ \dfrac{du_2}{dt} = -\omega^2 \sin(u_1) \end{cases} \tag{1.47}$$

$$u_1(0) = 3, \quad u_2(0) = 0, \quad \omega = 3. \tag{1.48}$$

4. Исследовать устойчивость и локальную погрешность следующих многошаговых методов:

$$U_{k+1} = U_{k-1} + 2\tau f_k, \tag{1.49}$$

$$U_{k+1} = U_{k-1} + \frac{\tau}{2}\left(f_{k+1} + f_k + f_{k-1}\right), \tag{1.50}$$

$$U_{k+1} = U_k + \frac{\tau}{6}\left(6f_k - 3f_{k-1} + 3f_{k-2}\right), \tag{1.51}$$

1.5. Жесткие системы. Метод Гира и неявные методы Рунге – Кутты

Большинство методов, используемых для решения задачи Коши являются условно устойчивыми. Условия устойчивости налагают определенные ограничения на размер шага численного интегрирования. Например, чтобы обеспечить устойчивость метода Адамса – Моултона третьего порядка точности при решении тестовой задачи (1.41) в случае $\lambda \leqslant 0$ необходимо выполнение условия $\tau_0 \leqslant 6/|\lambda|$, что непосредственно определяется границей области устойчивости, представленной на рис. 1.2. В то же время для устойчивости явного метода Адамса – Башфорта третьего порядка точности требуется размер шага в двенадцать раз меньше, чем для соответствующего неявного аналога.

Как правило, при решения задачи Коши для одного уравнения ограничения на размер шага, вытекающие из условий устойчивости, не являются более жесткими, нежели ограничения, продиктованные требованиями точности. Как следствие, при численном интегрировании одного уравнения явные методы представляются более предпочтительными. Однако, в случае систем ОДУ может возникать иная ситуация. Например, рассмотрим следующую систему дифференциальных уравнений:

$$\frac{du}{dt} = -Au, \tag{1.52}$$

где, для большей наглядности, A — симметричная, положительно определенная матрица 2×2, собственные значения которой $\lambda_1 = 1e3$, $\lambda_2 = 1e-3$. С помощью преобразований подобия система уравнений (1.52) может быть приведена к диагональному виду

$$\frac{d\psi}{dt} = -D\psi, \quad D = P^{-1}AP = \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix}, \quad u = P\psi, \tag{1.53}$$

где P — матрица преобразования подобия. Сделанные преобразования показывают, что решение задачи представлено двумя затухающими компонентами с существенно различающимися скоростями затухания:

$$\psi_1(t) = \psi_1(0)\exp(-\lambda_1 t), \quad \psi_2(t) = \psi_2(0)\exp(-\lambda_2 t).$$

Первая компонента является быстрой, однако при $t > 1$ ее значение близко к нулю и по абсолютному значению практически не меняется. Малость быстрой компоненты вместе с ее производными высших порядков указывает на малость локальной погрешности и возможность использования относительно больших значений шага. Кроме того, вторая компонента остается медленной и также допускает использование крупных шагов численного интегрирования.

Тем не менее, условия устойчивости для явных методов приводят к ограничению на размер шага:

$$\tau \leqslant 2\|A\|^{-1} = 2\lambda_1^{-1}. \tag{1.54}$$

Если условия устойчивости не выполнено, то стационарное состояние первой компоненты становится неустойчивым, как это показано на рис. 1.1.

Описанная выше ситуация является типичной для линейных систем дифференциальных уравнений вида (1.52), когда для собственных значений матрицы A, $\lambda(A) = \lambda_k$, $k = 1, 2, \ldots, n$, имеет место следующее:

$$\mathrm{Re}(\lambda(A)) > 0, \tag{1.55}$$

$$s = \frac{\max Re(\lambda(A))}{\min Re(\lambda(A))} >> 1. \tag{1.56}$$

Системы дифференциальных уравнений вида (1.52) с плохо обусловленной матрицей A, собственные значения которой характеризуются неравенствами (1.55)–(1.56), принято называть **жесткими**, а постоянную s — **коэффициент жесткости**. Коэффициент жесткости является количественной характеристикой жестких систем. Чем больше коэффициент жесткости, тем более сложным может оказаться численное интегрирование такой задачи с использованием условно устойчивых методов.

Понятие жесткости имеет естественное обобщение на случай нелинейных систем

$$\frac{du}{dt} = -f(t, u), \tag{1.57}$$

для которых, вместо матрицы A, неравенства (1.55)–(1.55), следует рассматривать применительно к собственным значениям матрицы Якоби, порождаемой вектор–функцией правой части системы — $f(t, u)$.

Для того, чтобы избежать дополнительных ограничений на размер шага, не связанных с требованиями точности, следует использовать безусловно устойчивые (A-устойчивые) неявные методы. Каждый шаг неявных методов требует больших вычислительных затрат, однако, благодаря возможности использовать более крупный размер шага без потери устойчивости, такие методы в случае жестких систем оказываются более предпочтительными по сравнению с явными. В рамках неявной

многошаговой схемы Адамса – Моултона мы можем получить абсолютно устойчивый метод не превосходящий второго порядка точности.

Для улучшения устойчивости многошагового метода более перспективным представляется использование чисто неявной схемы, основанной на **формулах дифференцирования назад** (**backward differentiation formulae**). Данный класс многошаговых методов известен также как **методы Гира**:

$$\sum_{m=0}^{M} a_m U_{k-m} = \tau f(t_k, U_k). \tag{1.58}$$

Для получения минимальной локальной ошибки коэффициенты a_m находятся как решение следующей системы алгебраических уравнений:

$$\begin{array}{lllll} a_1 & + & 2a_2 & +\ldots+ & Ma_m = -1, \\ a_1 & + & 2^2 a_2 & +\ldots+ & M^2 a_m = 0, \\ \ldots & & & & \\ a_1 & + & 2a_M & +\ldots+ & M^M a_m = 0, \end{array} \tag{1.59}$$

$$a_0 = -\sum_{m=1}^{M} a_m.$$

Методы Гира имеют глобальную погрешность $O(\tau^M)$ и сохраняют безусловную устойчивость для действительных λ вплоть до $M = 6$. Однако при $M > 6$ этот метод является абсолютно неустойчивым, т.е. устойчивость не может быть обеспечена сколь угодно малым размером шага τ.

Одношаговым аналогом метода Гира является неявный метод Эйлера (1.19). Наиболее значимые с практической точки зрения формулы дифференцирования назад имеют следующий вид и локальную погрешность:

$$\begin{aligned} &\frac{3}{2}U_k - 2U_{k-1} + \frac{1}{2}U_{k-2} = \tau f_k, && \delta_u = -\frac{2\tau^3}{9}u''', \\ &\frac{11}{6}U_k - 3U_{k-1} + \frac{3}{2}U_{k-2} - \frac{1}{3}U_{k-3} = \tau f_k, && \delta_u = -\frac{3\tau^4}{22}u^{(4)}, \\ &\frac{25}{12}U_k - 4U_{k-1} + 3U_{k-2} - \frac{4}{3}U_{k-3} + \frac{1}{4}U_{k-4} = \tau f_k, && \delta_u = -\frac{12\tau^5}{125}u^{(5)}, \\ &\frac{137}{60}U_k - 5U_{k-1} + 5U_{k-2} - \frac{10}{3}U_{k-3} + \frac{15}{12}U_{k-4} - \frac{1}{5}U_{k-5} = \tau f_k, && \delta_u = -\frac{10\tau^6}{137}u^{(6)}. \end{aligned} \tag{1.60}$$

Область устойчивости многошаговых методов Гира является неограниченной, как можно заметить из иллюстрации, представленной на рис. 1.4.

Неявные формулы дифференцирования назад приводят к системам нелинейных алгебраических уравнений вида

$$F(U_k) = 0, \quad F(U_k) = U_k - a_0^{-1}\tau f(t_k, U_k) + a_0^{-1}\sum_{m=1}^{M} a_m U_{k-m}, \tag{1.61}$$

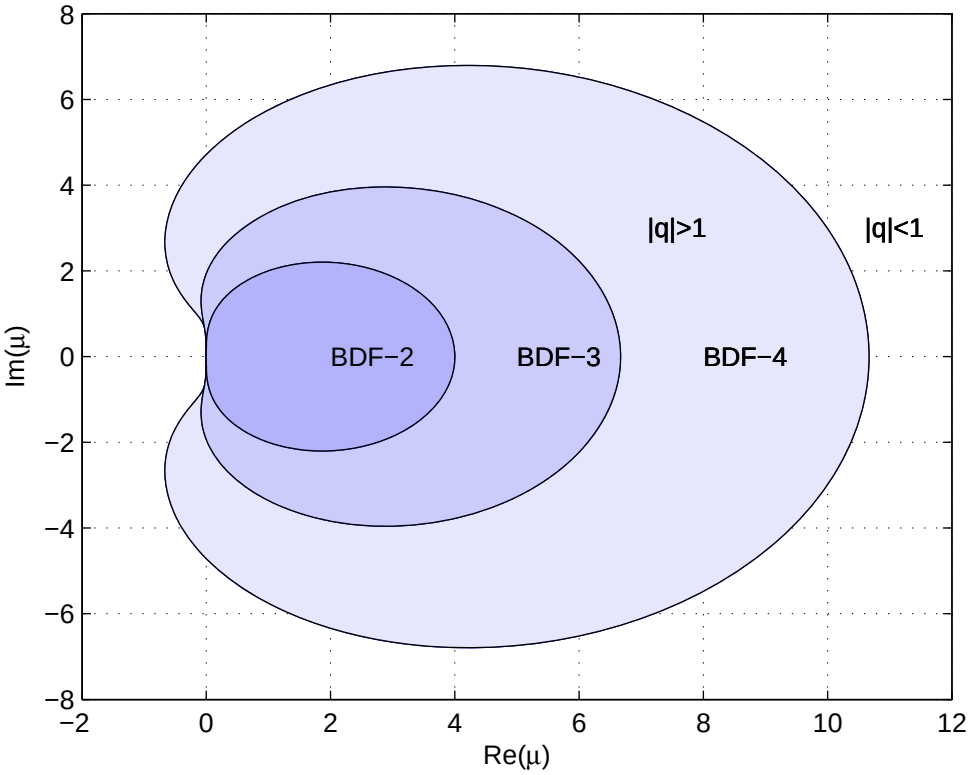


Рис. 1.4. Границы областей устойчивости чисто неявных формул дифференцирования назад для методов Гира второго – четвертого порядка. Отметим, что области устойчивости лежат снаружи закрашенных областей.

для решения которых может быть использован итерационный метод Ньютона:

$$U_k^{(s+1)} = U_k^{(s)} - \left[I - \tau a_0^{-1} \frac{\partial f_k^s}{\partial U} \right]^{-1} \left[U_k^{(s)} - \tau a_0^{-1} f_k^s + a_0^{-1} \sum_{m=1}^{M} a_m U_{k-m} \right]. \qquad (1.62)$$

Здесь $\dfrac{\partial f_k^s}{\partial U}$ — матрица Якоби, соответствующая вектор–функции правой части системы $f_k^s = f(U_k^s, t_k)$. Сходимость метода Ньютона (1.62) может быть обеспечена без существенных ограничений размера шага τ при достаточно близком начальном приближении $U_k^{(0)}$, например, используя какой-либо явный метод в качестве предиктора.

Другой подход к решению жестких систем основан на использовании **неявных методов Рунге – Кутты**. Напомним, что традиционно методы Рунге – Кутты явного типа (1.23) могут быть определены набором коэффициентов, которые удобно представить в виде таблицы Бутчера (1.23). В случае явных методов таблица Бут-

чера имеет треугольный вид. Естественным обобщением явных формул являются их неявные аналоги с полной матрицей коэффициентов A, $a_{km} \neq 0$, $m \geqslant k$:

$$\begin{aligned} K_1 &= \tau f(t_k + c_1\tau,\, U_k + \sum_{i=1}^{s} a_{1i}K_i), \\ K_2 &= \tau f(t_k + c_2\tau,\, U_k + \sum_{i=1}^{s} a_{2i}K_i), \\ &\cdots\cdots\cdots\cdots \\ K_s &= \tau f(t_k + c_s\tau,\, U_k + \sum_{i=1}^{s-1} a_{si}K_i), \end{aligned} \tag{1.63}$$

$$U_{k+1} = U_k + \sum_{m=1}^{s} b_m K_m, \tag{1.64}$$

где

$$\begin{array}{c|c} c & A \\ \hline & b^T \end{array} = \begin{array}{c|cccc} c_1 & a_{11} & a_{12} & \dots & a_{1s} \\ c_2 & a_{21} & a_{22} & \dots & a_{2s} \\ \vdots & \vdots & \vdots & \dots & \vdots \\ c_s & a_{s1} & a_{s2} & \dots & a_{ss} \\ \hline & b_1 & b_2 & \dots & b_s \end{array} \tag{1.65}$$

Значения коэффициентов c, b и A определяются требованием минимизации локальной погрешности метода. Например, двухстадийный неявный метод Рунге – Кутты максимальной точности, основанный на квадратурных формулах Гаусса – Лежандра, определяется следующим набором коэффициентов

$$\begin{array}{c|c} c & A \\ \hline & b^T \end{array} = \begin{array}{c|cc} \dfrac{3-\sqrt{3}}{6} & \dfrac{1}{4} & \dfrac{3-2\sqrt{3}}{12} \\ \dfrac{3+\sqrt{3}}{6} & \dfrac{3+2\sqrt{3}}{12} & \dfrac{1}{4} \\ \hline & \dfrac{1}{2} & \dfrac{1}{2} \end{array} \tag{1.66}$$

В отличие от явного метода Рунге – Кутты, где вспомогательные значения K_m, вычисляются последовательно, используя уже вычисленные значения $K_1, \dots K_{m-1}$, для определения последующих, $m = 1, 2, \dots s$, в неявном методе для вычисления значений K_m необходимо решить соответствующую систему алгебраических уравнений (1.63), вообще говоря, нелинейную. Размерность систем алгебраических уравнений, требующих решения на каждом шаге, пропорционально количеству уравнений

дифференциальной задачи, и количеству стадий неявного метода. Таким образом, неявные методы Рунге – Кутты с вычислительной точки зрения более затратны. Ситуация несколько упрощается в случае неявного диагонального метода Рунге – Кутты, когда матрица коэффициентов имеет нижний треугольный вид с ненулевыми диагональными значениями: $a_{km} \neq 0$, если $m \leqslant k$, и $a_{km} = 0$ в противном случае. При таких условиях каждое значение K_m находится независимо от остальных.

Среди достоинств неявных методов Рунге – Кутты, прежде всего следует отметить его преимущества в устойчивости, что делает весьма привлекательным использование этого класса методов для численного анализа жестких систем. В случае тестовой задачи (1.41) метод Рунге – Кутты (1.63) – (1.65) приводит к рекурсивному равенству

$$U_k = R(\tau\lambda)U_n, \tag{1.67}$$

где $R(z)$ имеет вид

$$R(z) = 1 + zb^T(I - zA)^{-1}e. \tag{1.68}$$

Здесь $e = (1, 1, \dots 1)^T$, I — единичная матрица. Условие устойчивости определяется неравенством:

$$|R(z)| \leqslant 1. \tag{1.69}$$

Как будет показано ниже, в рамках неявного метода Рунге – Кутты возможно построение A-устойчивых методов выше второго порядка точности.

Следует также отметить, что порядок точности неявного метода Рунге – Кутты при одинаковом числе стадий превосходит точность явной схемы. Так, например, двухстадийный неявный метод (1.69) имеет четвертый порядок точности, в то время как явная схема достигает только второго порядка. В общем случае, s-стадийный неявный метод Рунге –Кутты имеет порядок точности вплоть до $O(\tau^{2s})$ в то время как для явных аналогов порядок точности не превосходит $O(\tau^s)$. Для диагональной неявной схемы Рунге – Кутты максимальный порядок точности не превышает $O(\tau^{s+1})$. Пример такой двухстадийной диагональной неявной схемы имеет следующий набор коэффициентов:

$$\begin{array}{c|c} c & A \\ \hline & b^T \end{array} = \begin{array}{c|cc} \gamma & \gamma & 0 \\ 1-\gamma & 1-2\gamma & \gamma \\ \hline & \frac{1}{2} & \frac{1}{2} \end{array} \tag{1.70}$$

где $\gamma = \dfrac{3+\sqrt{3}}{6}$

Пример 1.3. Сравним область устойчивости явного и неявного метода Рунге – Кутты четвертого порядка точности, представленных коэффициентами (1.26) и (1.66) соответственно. Пример расчета области устойчивости методов Рунге – Кутты на основе условия (1.69) вместе с программной реализацией и результатами численных расчетов представлены ниже.

```
%% область устойчивости явного и неявного методов РК
%% A,b,e коэффициенты явной четырехстадийной схемы РК %%%%%
%% A1,b1,e1 коэффициенты неявной двухстадийной схемы РК  %%%
x = -4:0.01:1; y = -3.5:0.01:3.5; [X,Y] = ndgrid(x,y);
A = [0 0 0 0; 1/2 0 0 0;0 1/2 0 0;0 0 1 0];
A1 = [1/4 (3-2*sqrt(3))/12; (3+2*sqrt(3))/12 1/4];
b = [1 2 2 1]/6; b1 = [1/2 1/2];
e = [1 1 1 1]; e1 = [1 1];
E = eye(4); E1 = eye(2);
R = zeros(size(X)); R1 = R;
for k = 1:length(x);
for m = 1:length(y);
z = X(k,m)+1i*Y(k,m);
R(k,m) = abs(1+z*b*(inv(E-z*A)*e'));
R1(k,m) = abs(1+100*z*b1*(inv(E1-100*z*A1)*e1'));
end
end
%% граница области устойчивости задана
%% уравнением: 1-|RE(z)|=0 и отображается изолинией
subplot(2,1,1), [C, H]= contourf(X,Y,1-R,[0 3])
colormap([0.9 0.9 0.99])
title('Обрасть устойчивости явного метода РК 4-го порядка')
text(-0.7,0.9,'|R(z)|<1')
text(0.3,0.9,'|R(z)|> 1')
xlabel('Re(z)')
ylabel('Im(z)')
grid on
subplot(2,1,2), [C, H]= contourf(100*X,100*Y,1-R1,[0 3]);
colormap([0.9 0.9 0.99]);
text(-90,90,'|R(z)|< 1');
text(30,90,'|R(z)|>1')
xlabel('Re(z)')
ylabel('Im(z)')
title('Обрасть устойчивости неявного метода РК 4-го порядка')
grid on
```

Рассмотренный пример показывает, что неявный метод Рунге – Кутты четвертого порядка имеет неограниченную область устойчивости, покрывающую всю левую полуплоскость комплексной плоскости, т.е. данный метод является A-устойчивым.

Несмотря на превосходные свойства устойчивости и высокую точность, неявные методы Рунге – Кутты имеют серьезный недостаток, связанный с высокой вычислительной сложностью реализации. По этой причине практический интерес привлекают преимущественно варианты неявной схемы невысокого порядка точности для решения жестких систем при умеренных требованиях к точности результатов. В

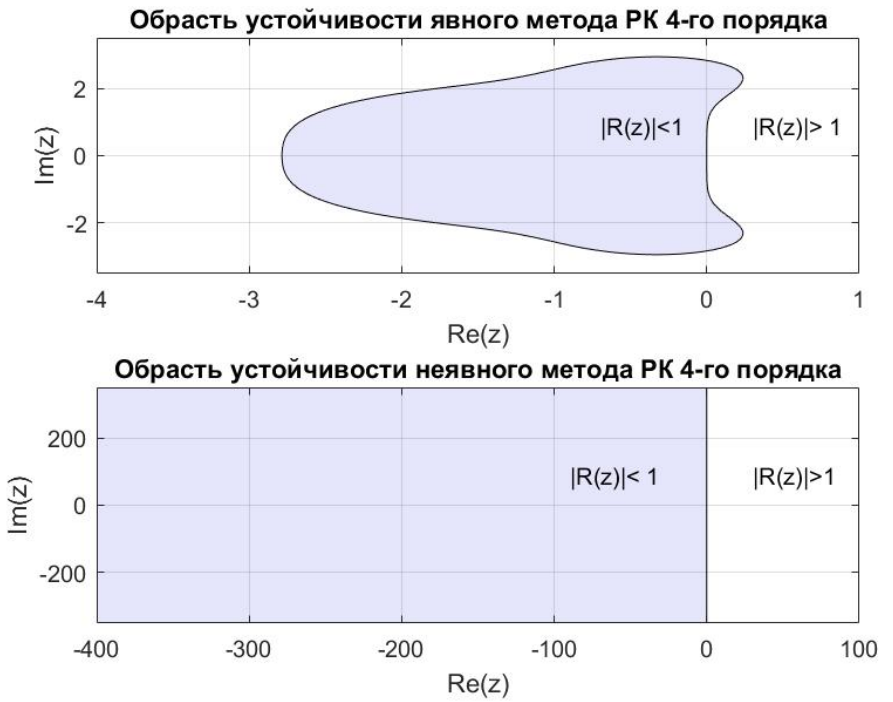


Рис. 1.5. Области устойчивости явного и неявного методов Рунге – Кутты четвертого порядка точности.

частности, некоторые варианты неявных схем реализованы в функциях MATLAB[1] **ode23tb**, **ode23t** и **ode23s**. Многошаговые методы переменного порядка, включая чисто неявные формулы дифференцирования назад, реализованы в функции **ode15s**. Сравнительный анализ вычислительной эффективности упомянутых методов рассмотрен на примере решения тестовой задачи ниже.

Пример 1.4. В качестве тестовой задачи для сравнительной оценки эффективности функций MATLAB, реализующих различные методы решения задачи Коши, рассмотрим систему Ван дер Поля:

$$\begin{aligned} \tfrac{du}{dt} &= v, \\ \tfrac{dv}{dt} &= \mu(1-u^2)v - u, \end{aligned} \tag{1.71}$$

$$u(0) = -2, \quad v(0) = 0, \quad t = [0, T_\mu]. \tag{1.72}$$

Использованы значения параметров $\mu = 1;\ 10;\ 100;\ 1000$ и $T_\mu = 5\mu$. Рассмотренная задача является классическим примером жесткой системы с коэффициентом

[1] См. подробнее Ashino, R., Nagase, M., and Vaillancourt, R. (2000). Behind and beyond the MATLAB ODE suite. Computers and Mathematics with Applications, 40(4), 491-512.

жесткости, определяемым значением параметра μ. В области значений параметра от $\mu = 1$ до $\mu = 1000$ мы можем проследить поведение различных методов при изменении свойств системы от не жесткой до сильно жесткой.

Наша цель состоит в сравнении эффективности различных неявных методов применительно к анализу жестких систем ОДУ. Полезно также проиллюстрировать некоторые возможности управления параметрами численных методов с целью достижения их максимальной эффективности.

Программная реализация рассмотренной задачи и результаты численных экспериментов представлены ниже.

```
%% Жесткая система Ван дер Поля system.%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% неявный метод РК 2-3-го порядка  (функция ODE23)
%% Формулы обратного дифференцирования 1-5го порядка (функция ODE15s )
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clear
settings = 'default';
settings = 'precalc';
k = 0;  Mu = [1 10 100 1000];
for M = Mu
k = k+1;  mu = M;
T=10*mu
options = odeset('RelTol',1.e-7); % Relative Tolerance
%%   Функция правой части и матрица Якоби
dydt = @(t,y)[y(2); mu*(1-y(1)^2)*y(2)-y(1)];
dfdt = @(t,y)[0 1; -2*mu*y(1)*y(2)-1 mu*(1-y(1)^2)];
if settings == 'default'
else
options = odeset('Jacobian',dfdt ); % Jacobian func.
end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
tic
sol_bdf = ode15s(dydt,[0 T],[-2 0],options); % BDF
time_BDF(k) = toc
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
tic
sol_irk = ode23s(dydt,[0 T],[-2 0],options); % IRK
time_IRK(k) = toc
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
tic
sol_irk = ode23tb(dydt,[0 T],[-2 0],options); % IRK
time_IRB(k) = toc
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
end
if settings == 'default'
subplot(2,1,1),
```

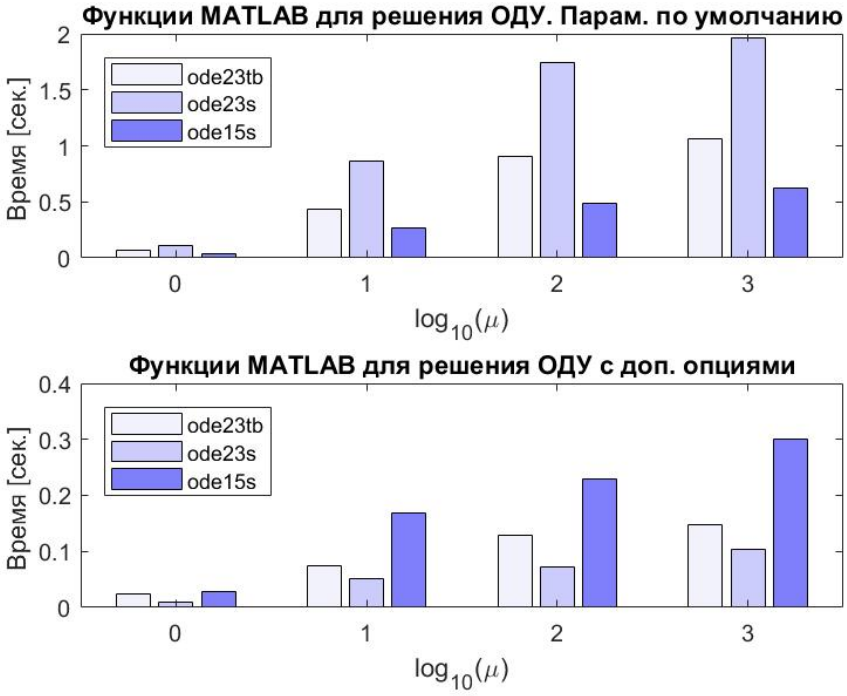


Рис. 1.6. Сравнение вычислительной эффективности функций MATLAB на примере моделирования жесткой системы Ван дер Поля (1.71), (1.72) при использовании параметров численного метода по умолчанию и при заданных формулах вычисления матрицы Якоби для вектор-функции правой части системы

```
bar(log10(Mu),[time_IRB(:) time_IRK(:), time_BDF(:)])
title('Функции MATLAB для решения ОДУ. Парам. по умолчанию')
else
subplot(2,1,2),
bar(log10(Mu),[time_IRB(:) time_IRK(:), time_BDF(:)])
title('Функции MATLAB для решения ОДУ с доп. опциями')
end
colormap([0.95 0.95 0.99; 0.8 0.8 0.99;0.5 0.5 0.99]);
legend('ode23tb','ode23s','ode15s','Location', 'NorthWest')
xlabel('log_{10}(\mu)');  ylabel('Время [сек.] ')
```

Результаты численного моделирования, представленные на рис. 1.6, позволяют заметить, что в случае, когда Якобиан системы не задан аналитически, существенное преимущество демонстрирует функция **ode15s**, реализующая неявные многошаговые методы переменного порядка. Эффективность многошаговых методов практически не изменяется при использовании аналитического вида Якобиана для реализации неявной нелинейной схемы, в то время как неявные методы Рунге – Кутты,

реализованные в функциях **ode23s** и **ode23tb** получают при этом почти десятикратное ускорение.

Упражнение 1.5.

1. Сравните области устойчивости неявного и неявно-диагонального метода Рунге – Кутты, заданных коэффициентами (1.66) и (1.70).

2. Сравните области устойчивости явных методов Рунге – Кутты четвертого и пятого порядков точности.

3. Вычислите коэффициенты неявного метода Гира 7-го порядка. Определите область устойчивости данного метода.

4. Внося необходимые модификации в программу рассмотренного выше примера сравните эффективность функций MATLAB **ode23tb**, **ode23s** , **ode15s**, **ode45** используя в качестве тестовой задачи систему Лоренца:

$$\begin{cases} \dfrac{dx}{dt} = \sigma(y - x), \\ \dfrac{dy}{dt} = x(\rho - z) - y, \\ \dfrac{dz}{dt} = xy - \beta z. \end{cases} \tag{1.73}$$

$$\sigma = 10, \quad \beta = 8/3, \quad \rho = 28, \quad x(0) = y(0) = 1, \quad z(0) = 0, , \quad t \in [0; 100]. \tag{1.74}$$

Сравните полученные результаты оценки эффективности на примерах систем Ван дер Поля и Лоренца.

Глава 2

Краевые задачи

2.1. Постановка задачи

Для дифференциальных задач порядка $n > 1$ дополнительные условия могут определяться как при одном, так и при разных значениях независимой переменной. Рассмотрим пример дифференциального уравнения второго порядка

$$\frac{d^2u}{dx^2} + a(x)u = f(x), \quad x \in (0,1), \tag{2.1}$$

с нулевыми краевыми условиями на концах отрезка:

$$u(0) = 0, \quad u(1) = 0. \tag{2.2}$$

Задачи такого типа, когда дополнительные условия задаются в разных точках, называются **краевыми задачами**.

Задача (2.1)–(2.2) допускает эквивалентное представление в виде системы двух дифференциальных уравнений первого порядка:

$$\begin{cases} \dfrac{dv}{dx} + a(x)u = f(x), \\ \dfrac{du}{dx} = v, \end{cases} \tag{2.3}$$

Формулировка краевой задачи в виде системы дифференциальных уравнений первого порядка в некоторых случаях представляется предпочтительной, особенно если искомые функции такой системы наделены конкретным смыслом. В общем случае двухточечная краевая задача для системы дифференциальных уравнений N-го порядка имеет следующий вид

$$\frac{d\mathbf{u}}{dx} = \mathbf{f}(\mathbf{u}, x), \quad \mathbf{f}(\mathbf{u}, x), = [f_1(\mathbf{u}, x), f_2(\mathbf{u}, x), \ldots, f_N(\mathbf{u}, x)]^T, \quad \mathbf{u} = [u_1, u_2, \ldots, u_N]^N \tag{2.4}$$

с краевыми условиями

$$\mathbf{B}^L(\mathbf{u})\,|_{x=L} = 0, \quad \mathbf{B}^R(\mathbf{u})\,|_{x=R} = 0\ , \tag{2.5}$$

где $x = L$ и $x = R$ — левая и правая границы отрезка, $\mathbf{B}^L$, $\mathbf{B}^R$ — вектор-функции,

$$\mathbf{B}^L = [b_1^L(\mathbf{u}), b_2^L(\mathbf{u}), \ldots, b_n^L(\mathbf{u})], \quad \mathbf{B}^R = [b_1^R(\mathbf{u}), b_2^R(\mathbf{u}), \ldots, b_{N-n}^R(\mathbf{u})],$$

такие, что общее число независимых условий (2.5) равняется числу уравнений системы (2.4).

2.1.1. Метод стрельбы

Идея **метода стрельбы** состоит в сведении исходной краевой задачи к последовательности задач Коши, решения которых сходятся и в пределе удовлетворяют краевым условиям на обоих концах интервала. Пусть требуется найти решение системы N дифференциальных уравнений (2.4), удовлетворяющих n условиям на левой границе, $x = L$, и $(N - n)$ условиям на правой границе, $x = R$ (2.5).

В методе стрельбы мы задаем начальные условия для всех искомых функций u_1, u_2, …, u_N на одной из границ отрезка. Эти начальные условия должны быть согласованы с оригинальными краевыми условия на этой границе. Поскольку число искомых функций больше, числа краевых условий на любой из границ, то недостающие условия рассматриваются как параметры пристрелки, значения которых следует задать таким образом, чтобы обеспечить выполнение краевых условий на противоположной границе.

Таким образом, задача состоит в том, чтобы найти такие значения параметров пристрелки чтобы решение задачи Коши с начальными условиями на одной границе удовлетворило краевым условиям на противоположной границе. Например, если задали краевые условия со свободными параметрами пристрелки $p_1, p_2, \ldots, p_n$ на левой границе ($x = L$) и решаем соотвествующую задачу Коши, тогда задача сводится к поиску корней системы уравнений

$$\mathbf{B}^R(\tilde{\mathbf{u}}(\mathbf{p}))\,|_{x=R} = 0\ , \tag{2.6}$$

где $\tilde{\mathbf{u}}(\mathbf{p})$ — решение задачи Коши, зависящее от параметров пристрелки. Для решения данной системы (в общем случае нелинейных) уравнений (2.6) может быть использован итерационный метод Ньютона с параметром:

$$p^{(k+1)} = p^{(k)} - \tau_k J^{-1}\left\{B^R(\tilde{u}^{(k)}, p^{(k)})\right\} B^R(\tilde{u}^{(k)}, p^{(k)}), \quad k = 1, 2, \ldots, \tag{2.7}$$

где итерационный параметр $0 < \tau_k \leqslant 1$ позволяет обеспечить глобальную сходимость метода при произвольном начальном приближении, $J\left\{B(\tilde{u}^{(k)}, p^{(k)})\right\}$ — матрица Якоби:

$$J\left\{B^R(\tilde{u},p)\right\} = \begin{bmatrix} \dfrac{\partial B_1^R(\tilde{u},p)}{\partial p_1} & \dfrac{\partial B_1^R(\tilde{u},p)}{\partial p_2} & \cdots & \dfrac{\partial B_1^R(\tilde{u},p)}{\partial p_{N-n}} \\ \dfrac{\partial B_2^R(\tilde{u},p)}{\partial p_1} & \dfrac{\partial B_2^R(\tilde{u},p)}{\partial p_2} & \cdots & \dfrac{\partial B_2^R(\tilde{u},p)}{\partial p_{N-n}} \\ \cdots & \cdots & \cdots & \cdots \\ \dfrac{\partial B_{N-n}^R(\tilde{u},p)}{\partial p_1} & \dfrac{\partial B_{N-n}^R(\tilde{u},p)}{\partial p_2} & \cdots & \dfrac{\partial B_{N-n}^R(\tilde{u},p)}{\partial p_{N-n}} \end{bmatrix} \tag{2.8}$$

Преимущество метода стрельбы состоит в относительной простоте и возмоности использовать широкий арсенал методов решения задач Коши, к решению которых сводится краевая задача. Точность решения определяется главным образом точностью методов, использованных для решения задач Коши. Метод стрельбы удобен в случае неединственности решения, например, при решении задач на собственные функции и собственные значения, при исследованиий мультистабильности нелинейных пространственно распределенных систем.

Вместе с тем, отметим слоности, которые могут возникать при использовании метода стрельбы. Первое, это то, что задача Коши к которой мы пытаемся свести краевую задачу, может оказаться неустойчивой и ее решение уходит в бесконечность, не позволяя достичь противоположной границы. Второе, система уравнений для определения параметров пристрелки (2.6) может оказаться вырожденной или плохо обусловленной. При возникновении отмеченных пробле можно попытаться выполнить пристрелку с противоположной границы. Если это не решает проблему, то есть основание попробовать какой-либо алтернативный подход, например, метод конечных разностей или спектральный метод.

Сходимость итерационной процедуры, используемой для опредедения параметров пристрелки, также может представлять отдельную задачу. Обычно проблема сходимости может быть решена путем выбора подходящего начального приближения и значения итерационного параметра. Дополнительно, можно попытаться найти такие значения параметров задачи (коэффициентов уравнения или значения начальных условий) при которых итерационный процесс сходится. Далее, используя полученные результаты в качестве начальных условий и плавно подстраивая параметры к требуемым значениям, шаг за шагом приближаем решение поставленной задачи.

Собственные моды диэлектрического волновода удовлетворяют дифференциальному уравнению

$$\frac{d^2u}{dx^2} = (\varepsilon(x) - \lambda^2)u, \tag{2.9}$$

и краевым условиям

$$\lim_{x\to\pm\infty} u(x) = 0, \tag{2.10}$$

Пусть требуется найти несколько первых собственных функций и собственных значений дифференциальной задачи (2.9)–(2.10), для профиля коэффициента преломления волновода $\varepsilon(x) = 4(1 - \exp(-(x/2)^4)$.

Полагаем, что собственные моды низкого порядка локализованы и быстро затухают при удалении от волновода. Таким образом, без потери точности мы можем заменить краевые условие на бесконечности на соотвествующее краевые условие на достаточно большом, но конечном расстоянии от оси волновода, $x = \pm L$. Сформулируем задачу в виде системы дифференциальных уравнений первого порядка:

$$\begin{cases} \dfrac{dv}{dx} = (\varepsilon(x) - \lambda^2)u, \\ \dfrac{du}{dx} = v, \\ \dfrac{d\lambda}{dx} = 0, \end{cases} \tag{2.11}$$

Отметим, что мы включили неизвестное собственное значение $\lambda = \text{const}$, в качестве дополнительной искомой функции в систему дифференциальных уравнений, Отметим также, что искомые функции u и v определены с точностью до постоянного мложителя, пэтому начальные или краевые условия для одной из компонент могут быть заданы произвольным значением.

Таким образом, мы решаем систему (2.11) на отрезке $(-L, L)$ с начальными условиями

$$v(-L) = 1.e-3,\ u(-L) = 0, \lambda(-L) = p, \tag{2.12}$$

где p — параметр пристрелки, значение которого должно быть определено как корень уравнения $u(L, p) = 0$, чтобы удовлетворить краевому условию на правой границе отрезка. Для решения уравнения $u(L, p) = 0$ будем использовать итерационный метод Ньютона:

$$p^{(k+1)} = p^{(k)} - u(L, p^{(k)}) \left[\frac{\partial u(L, p^{(k)})}{\partial p} \right]^{-1}, \quad k = 0, 1, 2, \ldots. \tag{2.13}$$

Поскольку функция $u(L, p)$ задана неявно, то для вычисления ее производной по переменной p воспользуемся конечно-разностным приближением,

$$\frac{\partial u(L, p^{(k)})}{\partial p} \simeq \frac{u(L, p^{(k)}) - u(L, p^{(k-1)})}{p^{(k)} - p^{(k-1)}},$$

выполняя две предварительные итерации с близкими по величения значениями параметра пристрелки $p = p^{(0)}, p^{(1)}$

Для вычисления $u(L, p^{(k)})$ воспользуемся стандартной функцией MATLAB `ode45`, применяя ее к решению задачи (2.11)–(2.12). Пример реализации метода стрельбы и результаты численных экспериментов представлены ниже.

```
%% Проблема собственных функций и собственных значений
```

```
%%  Метод стрельбы для системы ОДУ
U = @(x,u)[-u(2)*(u(3)^2-4*(1-exp(-x^4/2^4))); u(1); 0];
L=5.0;
Shot_0 = 1.3;
Shot = Shot_0*1.01;
Tol=1.e-9;
% two preliminary shoots
% f Newton's iterations
[x1,y1] = ode45(U,[-L L],[1.e-4, 0, Shot_0],Tol);
[x2,y2] = ode45(U,[-L L],[1.e-4, 0, Shot],Tol);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
plot(x2,y2(:,2),'.-')
f1 = y2(end,2);
f0 = y1(end,2);
df = (f1-f0)/(Shot-Shot_0);
k=0;
while abs(f1) > 1.e-6 & k<50
f0=f1;
Shot_0=Shot;
Shot = Shot-f1/df; % Итерации метода Ньютона
[x2,y2]=ode45(U,[-L:0.1:L],[1.e-4, 0, Shot],Tol);
f1 = y2(end,2);
df = (f1-f0)/(Shot-Shot_0);
k=k+1
Y(:,k)=y2(:,2)/max(abs(y2(:,2)));
s{k}=['u^{(k)} k= ',num2str(k)];
end
plot(x2,Y(:,1:end-1),'.-')
hold on
plot(x2,Y(:,end),'o-')
title(['The Shooting method for eigenfunctions problem:...
\lambda^2=',num2str(Shot^2,6)])
axis([-L,L,-1.4,1.4])
grid
legend(s{1:end});
```

1. Выполните расчет собственных значений в задаче (2.9)–(2.10) используя различные значения $L = 5; 7; 10$, и сравните полученные значения величины λ. Какое заключение может быть сделано относительно корректности замены краевых условий на бесконечности краевыми условиями на концах отрезка конечной длины.

2. По аналогии с рассмотренным выше примером, реализуйте метод стрельбы для решения следующей краевой задачи:

$$\begin{cases} \dfrac{du}{dx} = -v^2 u, \\ \\ \dfrac{dv}{dx} = -u^2 v, \end{cases} \tag{2.14}$$

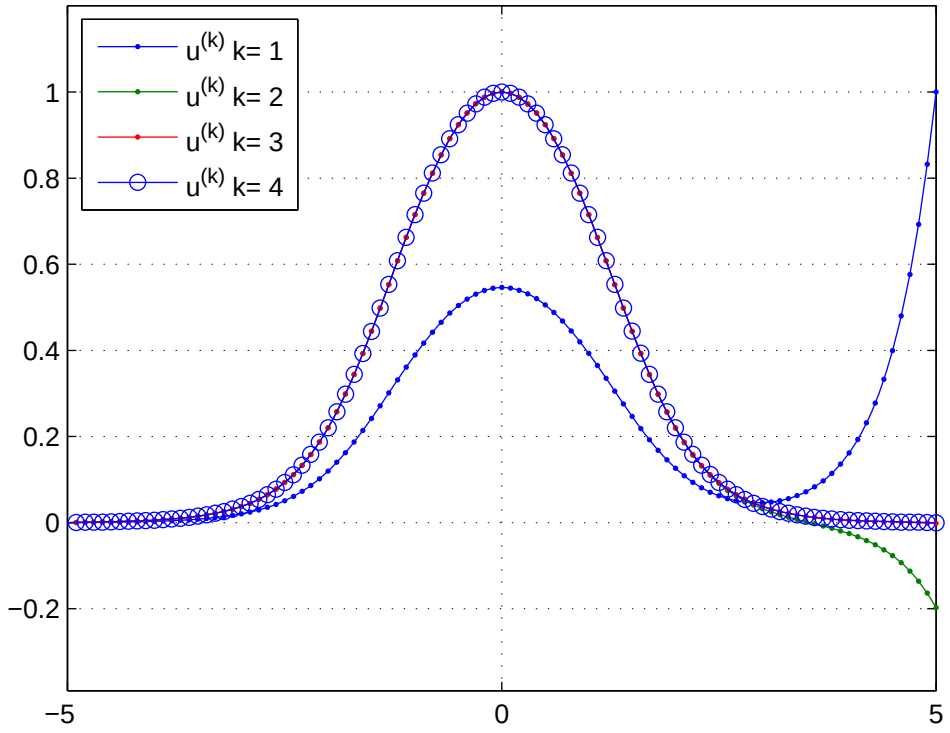


Рис. 2.1. Сходимость метода стрельбы при решении задачи расчета собственных собственных функций (2.9)–(2.10) с начальным приближением $\lambda^{(0)2} = 0.8$: $\lambda^2 = 0.61360$. Приближенные решения начиная с третьей итерации становятся графически неразличимыми.

$$u(0) = 2, \quad v(4) = 0.01 \tag{2.15}$$

Попытайтесь использовать метод стрельбы для решения уравнений (2.14) –(2.15) с начальными условиями на правой границе отрезка. Объясните причину невозможности пристрелки от правой границы при решении рассмотренной задачи.

3. Для решение уравнений (2.14) имеет место следующий инвариант:

$$\frac{du^2}{dx} - \frac{dv^2}{dx} = 0. \tag{2.16}$$

Докажите его выполнение. Убедитесь, что для приближенного решения $\tilde{u}, \tilde{v}$, полученного методом стрельбы выполняется равенство

$$\tilde{u}^2(x) - \tilde{v}^2(x) = \text{const} \pm |\delta|, \tag{2.17}$$

где δ в пределах точности приближенного решения.

4. Попытайтесь решить задачу (2.14)–(2.15) используя в методе стрельбы для решения задачи Коши функцию ode113 вместо ode45. Сравните точность выполнения тождества (2.17) в этих двух случаях.

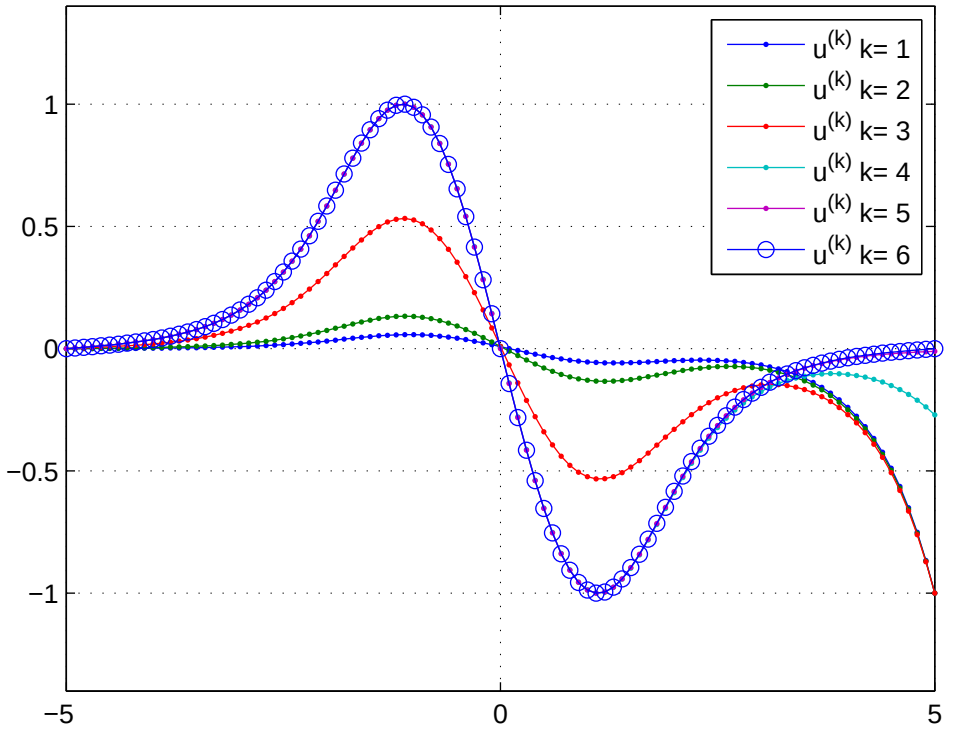


Рис. 2.2. Сходимость метода стрельбы при решении задачи расчета собственных собственных функций (2.9)–(2.10) с начальным приближением $\lambda^{(0)2} = 1.3$: $\lambda^2 = 2.09056$. Приближенные решения начиная с пятой итерации становятся графически неразличимыми.

5. Модифицируйте программную реализацию рассмотренного выше примера для решения следующей краевой задачи:

$$\begin{cases} \dfrac{du}{dx} = iv + i\left(|u|^2 u + 2|v|^2\right) u \\ \dfrac{dv}{dx} = -iu - i\left(|v|^2 u + 2|u|^2\right) v, \end{cases} \tag{2.18}$$

$$u(0) = p, \quad v(2) = 0. \tag{2.19}$$

Здесь $i = \sqrt{-1}$. Просканируйте параметр p от $p = 0.2$ до $p = 2$ и обратно с шагом 0.1 используя каждый раз полученное решение в качестве начального приближения для следующего значения p. Постройте зависимость $v(x = 0, p)$ при возрастании и убывании значения параметра p. Полученная зависимость иллюстрирует наличие бистабильности в рассмотренной задаче.[1]

[1] Подробнее см. H. G. Winful, J. H. Marburger, and E. Garmire. *Theory of bistability in nonlinear distributed feedback structures*, Appl. Phys. Lett. 35, 379–381 (1979)

2.2. Метод конечных разностей и конечных элементов для решения краевых задач

Основная идея большинства методов численного анализа дифференциальных уравнений состоит в переходе от непрерывной задачи в бесконечномерном функциональном пространстве V к дискретной задаче, формулируемой в конечномерном векторном пространстве V_N. Среди наиболее универсальных подходов к решению краевых задач для дифференциальных уравнений следует в первую очередь отметить **методы конечных разностей** и **конечных элементов** (finite difference (FD) and finite element methods (DEM)).

В методе конечных разностей, в отличие от исходной дифференциальной задачи, искомое решение является не функцией, заданной в некоторой ограниченной области, а множество приближенных значений этой функции, определенных в конечном числе точек, принадлежащих области определения решения.

Теоретическую основу метода конечных элементов составляют слабая формулировка задачи, метод Галеркина и представление решения в виде линейной комбинаций финитных базисных функций $\Psi_n(x)$, $n = 1, \ldots, N$, имеющих компактный носитель и удовлетворяющих краевым условиям:

$$U(x) = \sum_{n=1}^{N} c_n \Psi_n(x), \quad \Psi_n(x) \neq 0, \ |x - x_n| \leqslant \Delta x, \quad \Psi_n(x) = 0, \ |x - x_n| > \Delta x, \tag{2.20}$$

Несмотря на принципиальное отличие методов конечных разностей и конечных элементов, построенные в рамках данных подходов дискретные модели могут при определенных условиях совпадать. Рассмотрим подробнее эти два метода на примере решения дифференциального уравнения второго порядка на единичном отрезке с нулевыми краевыми условиями Дирихле:

$$\frac{d}{dx}\lambda(x)\frac{du}{dx} - q(x)u = f(x), \quad x \in (0,1), \tag{2.21}$$

$$u(0) = 0, \quad u(1) = 0. \tag{2.22}$$

Для построение дискретной разностной модели на отрезке $\Omega = [0,1]$ определим множество точек $\omega_h = \{x_0, x_1, \ldots, x_N\}$, разбивающее область определения решения дифференциальной задачи на некоторое число интервалов. Как правило $0 \leqslant x_0 < x_1 < \ldots < x_{N-1} < x_N \leqslant 1$. Множество ω_h называется **сеткой**, а сами точки x_k, $k = 0, \ldots, N$, — **узлы сетки** . Расстояние между соседними узлами называют **шагом сетки**, $h_k = x_k - x_{k-1}$. Для простоты мы ограничимся рассмотрением случая **равномерной сетки**, когда все шаги одинаковы: $h_1 = h_2 = \ldots h_n = h$. Для точного и приближенного решений в узлах сетки мы будем использовать соответственно следующие обозначения: $u(x_k) = u_k$, $U(x_k) = U_k$.

Конечно–разностный метод основан на замене производных дифференциальной задачи на соответствующие разностные приближения. Мы будем использовать следующие приближенные соотношения для представления производных на сетке:

$$\frac{du(x_k)}{dx} = \frac{u_{k+1} - u_{k-1}}{2h} + \frac{h^2}{6}u''' + O(h^4), \tag{2.23}$$

$$\frac{d^2u(x_k)}{dx^2} = \frac{u_{k+1} - 2u_k + u_{k-1}}{h^2} + \frac{h^2}{12}u^{(4)} + O(h^4), \tag{2.24}$$

$$\frac{d}{dx}\lambda(x_k)\frac{du(x_k)}{dx} = \frac{\lambda_{k+1/2}\left(u_{k+1} - u_k\right) - \lambda_{k-1/2}\left(u_k - u_{k-1}\right)}{h^2} + O(h^2), \tag{2.25}$$

где $\lambda_{k\pm1/2} = \lambda((x_k + x_{k\pm1})/2)$ или $\lambda_{k\pm1/2} = \left(\lambda(x_k) + \lambda(x_{k\pm1})\right)/2$.

Уравнения (2.23)–(2.25) получены на основе представления решения и коэффициентов дифференциальной задачи отрезком степенного ряда:

$$u_{k\pm1} = u_k \pm \frac{u'(x_k)}{1!}h + \frac{u''(x_k)}{2!}h^2 \pm \frac{u'''(x_k)}{3!}h^3 + \frac{u^{(4)}(x_k)}{4!}h^4 + O(h^5). \tag{2.26}$$

$$\lambda_{k\pm1/2} = \lambda_k \pm \frac{\lambda'(x_k)}{2\cdot 1!}h + \frac{\lambda''(x_k)}{4\cdot 2!}h^2 \pm \frac{\lambda'''(x_k)}{8\cdot 3!}h^3 + \frac{\lambda^{(4)}(x_k)}{16\cdot 4!}h^4 + O(h^5). \tag{2.27}$$

Подстановка выражений (2.25) в (2.21) приводит к следующим уравнениям

$$\frac{\lambda_{k+1/2}\left(u_{k+1} - u_k\right) - \lambda_{k-1/2}\left(u_k - u_{k-1}\right)}{h^2} - q(x_k)u_k - f(x_k) = R(h), \quad x_k \in \omega_h, \tag{2.28}$$

где $R(h) = O(h^2)$ — **погрешность аппроксимации**, возникающая в силу приближенности формул разностного дифференцирования (2.23) – (2.25). Отметим, что если в разложении (2.26)–(2.27) производные высших порядком существуют и ограничены, то погрешность аппроксимации стремиться к нулю при убывании шага сетки:

$$\lim_{h\to 0} R(h) = 0.$$

Если шаг сетки достаточно мал, мы можем пренебречь погрешностью аппроксимации, что приводит к следующему уравнению для приближенного решения задачи (2.21))–(2.22):

$$\frac{\lambda_{k+1/2}\left(U_{k+1} - U_k\right) - \lambda_{k-1/2}\left(U_k - U_{k-1}\right)}{h^2} - q(x_k)U_k - f(x_k) = 0, \quad k = \overline{1, N-1}, \tag{2.29}$$

$$U_0 = U_N = 0. \tag{2.30}$$

Уравнения вида (2.29)–(2.30), полученные путем замены производных дифференциальной задачи соответствующими формулами разностного дифференцирования, называются **разностной схемой**.

Разностные схемы являются приближенными дискретными моделями дифференциальных задач. В рассмотренном случае разностная модель имеет вид системы линейных алгебраических уравнений с трехдиагональной матрицей. Отметим, что различие между конечно–разностным уравнением (2.28) для точного решения исходной дифференциальной задачи и разностной схемой (2.29)–(2.30) определяется величиной $R(h)$, которая может быть интерпретирована как **невязка** — дисбаланс в приближенном разностном уравнении при подстановке в него точного решения исходной дифференциальной задачи. Мы будем говорить, что разностная схема **согласована** или разностная схема **аппроксимирует** дифференциальную задачу, если

невязка (погрешность аппроксимации) разностной схемы на решении соответствующей дифференциальной задачи стремится к нулю при шаге сетки, стремящемся к нулю.

Как правило, порядок малости невязки разностной схемы характеризует скорость сходимости ее решения. Напомним, что **скорость сходимости** или **порядок точности** численного метода — это асимптотическая скорость убывания разницы между приближенным решением U_k и его точными значениями $u_k = u(x_k)$ в узлах сетки при шаге сетки стремящемся к нулю. Скорость сходимости является одной из важнейших характеристик разностных схем — количественной мерой их точности. Говоря о том, что численный метод имеет n-й порядок точности мы подразумеваем, что при достаточно малом h погрешность приближенного решения $\delta(x_k) = U(x_k) - u(x_k)$, $x_k \in \omega_h$, имеет такой же порядок малости: $\|\delta\| \leqslant Ch^n = O(h^n)$, где C — постоянная не зависящая от h.

Основное отличие метода конечных элементов от разностных методов состоит в так называемой слабой формулировке дифференциальной задачи. Несложно заметить в уравнениях (2.23)–(2.25), что для согласованности (аппроксимации) разностной схемы требуется существование и ограниченность производных решения высшего порядка. Например, для получения второго порядка аппроксимации разностной схемы (2.28) мы должны иметь как минимум четырежды дифференцируемое решение дифференциальной задачи, в то время как соответствующее дифференциальное уравнение содержит только вторые производные. Такого рода "сильные"требования, предъявляемые к дифференцируемости решения задачи, могут быть преодолены при использовании слабой формулировки. Слабая формулировка задачи получается путем умножения дифференциального уравнения (2.31) на некоторую дифференцируемую тестовую функцию $\Psi(x)$ с последующим интегрированием полученного равенства, используя формулы интегрирования по частям:

$$\int_0^1 \lambda(x)\frac{du}{dx}\frac{d\Psi(x)}{dx}\,dx + \int_0^1 q(x)u(x)\Psi(x)\,dx + \int_0^1 f(x)\Psi(x)\,dx = 0. \qquad (2.31)$$

Интегральное уравнение (2.31) называется слабой формулировкой задачи, а функция $u(x)$, удовлетворяющее данному уравнению, называется слабым решением дифференциальной задачи (2.21)–(2.22).

При построении дискретной модели на основе методов конечных элементов мы также используем дискретизацию области определения решения, разбивая ее на подобласти множеством узлов сетки. Кроме того, мы определяем множество базисных функций — **функций формы**. Обычно в качестве базисных функций используются кусочно–полиномиальные функции (B-сплайны), отличные от нуля в пределах ячейки сетки. Приближенное решение задачи находится в виде линейной комбинации базисных функций (2.20), с коэффициентами c_n, которые определяются на основе **метода Галеркина**: коэффициенты c_n таковы, что невязка дискретной модели ортогональна всем базисным функциям. Фактически метод Галеркина соответствует ортогональной проекции решения задачи на линейную оболочку базисных функций Ψ_n, $n = 1, \ldots, N$, что обеспечивает минимальную погрешность приближения в гильбертовом пространстве.

Ортогональность понимается здесь в смысле равенства нулю скалярного произведения векторов. Скалярное произведение функций на отрезке $[a, b]$ определяется следующим образом:

$$(\Psi_n, \Psi_m) = \int_a^b \Psi_n(x)\Psi_m(x)\,dx. \tag{2.32}$$

Две функции считаются ортогональными если их скалярное произведение равно нулю: $(\Psi_n, \Psi_m) = 0$.

Для вычисления коэффициентов c_n, $n = 1, \ldots, N$, подставляем решение (2.20) в уравнение (2.31):

$$\sum_{n=1}^{N} \left\{ \frac{d}{dx}\lambda(x)\frac{d\Psi_n(x)}{dx}c_n - q(x)c_n\Psi_n(x) - f(x) \right\} = R(x). \tag{2.33}$$

После этого умножим полученное уравнение (2.33) поочередно на базисные функции $\Psi_m(x)$, $m = 1, \ldots, N$ и проинтегрируем полученные равенства:

$$\sum_{n=1}^{N} c_n \left\{ \int_0^1 \frac{d}{dx}\lambda(x)\frac{d\Psi_n(x)}{dx}\Psi_m(x)\,dx - \int_0^1 q(x)\Psi_n(x)\Psi_m(x)dx \right\} - \int_0^1 f(x)\Psi_m(x)\,dx - \int_0^1 R(x)\Psi_m(x)\,dx = 0, \quad m = 1, \ldots, N. \tag{2.34}$$

Последний интеграл в уравнении (2.34) равен нулю в силу требования ортогональности невязки и пробных функций. Вычисление первого интеграла в уравнении (2.34) по частям приводит к выражению

$$-\sum_{n=1}^{N} \left\{ \int_0^1 \lambda(x)\frac{d\Psi_m(x)}{dx}\frac{d\Psi_n(x)}{dx}\,dx - \int_0^1 q(x)\Psi_n(x)\Psi_m(x)\,dx \right\} c_n = \int_0^1 f(x)\Psi_m(x)\,dx. \tag{2.35}$$

Уравнения (2.35), $m = 1, \ldots, N$, соответствуют слабой формулировке задачи (2.21)–(2.22). Коэффициенты c_n находятся как решение следующей системы линейных алгебраических уравнений:

$$Ac = g, \tag{2.36}$$

где $c = [c_1, c_2, \ldots, c_N]^T$, $g = [g_1, g_2, \ldots, g_N]^T$,

$$\{A_{mn}\} = \int_0^1 \lambda(x)\frac{d\Psi_m(x)}{dx}\frac{d\Psi_n(x)}{dx}\,dx + \int_0^1 q(x)\Psi_n(x)\Psi_m(x)\,dx, \tag{2.37}$$

$$g_m = \int_0^1 f(x)\Psi_m(x)\,dx. \tag{2.38}$$

В общем случае матрица системы уравнений (2.36) является симметричной и полной (отсутствует преобладание нулевых элементов). Однако, если используемые базисные функции $\Psi_m(x)$ имеют финитный носитель на отрезке $x \in [x_{m-1}, x_{m+1}]$, то матрица системы (2.36) будет иметь трехдиагональную структуру, аналогичную структуре матрицы разностной модели: $\{A_{m,n}\} = 0$, когда $|n-m|1$.

Пример 2.1. Сравним возможности методов конечных разностей и конечных элементов на примере решения задачи (2.21)–(2.22) для следующих входных данных: $\lambda(x) = 1$, $q(x) = 0$, $f(x) = 32(x-0.5)^4 - 4(x-0.5)^2$. Разностный метод в данном случае приводит к следующей системе линейных алгебраических уравнений

$$AU = b, \tag{2.39}$$

где $b = [f(x_1), f(x_2), \ldots, f(x_{N-1})]^T, U = [U_1, U_2, \ldots, U_{N-1}]^T$ — приближенное решение задачи в точках равномерной сетки ω_h, A — симметричная трехдиагональная матрица размерности $(N-1)\times(N-1)$:

$$A = \frac{1}{h^2}\begin{bmatrix} -2 & 1 & & & & & \\ 1 & -2 & 1 & & & & \\ & \ddots & \ddots & \ddots & & 0 & \\ & & & & & & \\ & & & & & & \\ & 0 & & & \ddots & \ddots & \ddots \\ & & & & & 1 & -2 & 1 \\ & & & & & & 1 & -2 \end{bmatrix} \tag{2.40}$$

Отметим, что в силу нулевых краевых условий мы формулируем разностную задачу только для внутренних точек сетки (x_k, $k = 1, \ldots, N-1$). Для учета ненулевых условий Дирихле ($u(0) = \alpha$, $u(1) = \beta$) следует воспользоваться соответствующей коррекцией первой и последней компонент вектора правой части системы:: $b(1) = f(x_1) - \alpha/h^2$, $b(N-1) = f(x_{N-1}) - \beta/h^2$.

В случае краевых условий Неймана, входящие в них производные следует аппроксимировать конечными разностями и приближенное решение в граничных точках вместе с соответствующими уравнениями включаются в дискретную разностную модель. Например, в случае **краевых условий Неймана** общего вида

$$\left[\lambda(x)\frac{\partial u}{\partial x} + \alpha_0(x)u + \gamma_0(x)\right]_{x=0} = 0, \quad \left[\lambda(x)\frac{\partial u}{\partial x} + \alpha_L(x)u + \gamma_L(x)\right]_{x=L} = 0, \tag{2.41}$$

наиболее подходящим представляется использование модифицированной сетки $\overline{\omega}_h$, в которой первый и последний узлы располагаются на расстоянии в половину шага снаружи области определения решения $x \in [0, L]$:

$$\overline{\omega}_h = \{x_k = (k-1/2)h, \quad k = 0, 1, \ldots, N, \quad h = L/(N-1)\}. \tag{2.42}$$

Использование такой сетки с фиктивными узлами позволяет легко получить второй порядок аппроксимации краевых условий, поскольку граничные точки в данном случае является средними точками крайних интервалов сетки, в которой разностные производные и усредненные по двум ближайшим узлам значения сеточных функций обеспечивают второй порядок точности:

$$\lambda(x_0 + h/2)\,(U_1 - U_0)\,/h + \alpha_0(x_0 + h/2)\,(U_1 + U_0)\,/2 + \gamma_0(x_0 + h/2). \tag{2.43}$$

$$\lambda(x_N - h/2)\,(U_N - U_{N-1})\,/h + \alpha_0(x_N - h/2)\,(U_N + U_{N-1})\,/2 + \gamma_0(x_N - h/2). \tag{2.44}$$

Эти дополнительные два уравнения должны быть добавлены в качестве первого и последнего уравнений системы (2.39). Как следствие, размерность системы разностных уравнений возрастает с $(N-1)\times(N-1)$ до $(N+1)\times(N+1)$.

Рассмотрим также метод конечных элементов на основе линейных функций формы (линейных B-сплайнов):

$$\Psi_k(x) = \begin{cases} 1 - \dfrac{|x - x_k|}{h}, & x_{k-1} \leqslant x \leqslant x_{k+1}, \\ \\ 0, & x_{k-1} > x > x_{k+1}. \end{cases} \tag{2.45}$$

Для производных данных функций мы имеем

$$\frac{d\Psi_k(x)}{dx} = \begin{cases} \dfrac{1}{h}, & x_{k-1} \leqslant x \leqslant x_k, \\ \\ -\dfrac{1}{h}, & x_k \leqslant x \leqslant x_{k+1}, \\ \\ 0, & x_{k-1} > x > x_{k+1}. \end{cases} \tag{2.46}$$

Для расчета элементов матрицы и компонент вектора правой части системы (2.36) мы можем использовать точные выражения (2.37)–(2.38), или их приближения, например, второго порядка точности:

$$\int\limits_0^1 \lambda(x)\frac{d\Psi_m(x)}{dx}\frac{d\Psi_n(x)}{dx}\,dx \simeq \frac{\lambda(x_m) + \lambda(x_n)}{2}\int\limits_0^1 \frac{d\Psi_m(x)}{dx}\frac{d\Psi_n(x)}{dx}\,dx, \tag{2.47}$$

$$\int\limits_0^1 q(x)\Psi_m(x)\Psi_n(x)\,dx \simeq \frac{q(x_m) + q(x_n)}{2}\int\limits_0^1 \Psi_m(x)\Psi_n(x)\,dx, \tag{2.48}$$

$$\int\limits_0^1 f(x)\Psi_m(x)\,dx \simeq f(x_m)\int\limits_0^1 \Psi_m(x)\,dx. \tag{2.49}$$

Интегралы в формулах (2.47)–(2.49) для конкретного вида функций формы (2.45) могут быть вычислены аналитически:

$$\int_0^1 \frac{d\Psi_m(x)}{dx}\frac{d\Psi_n(x)}{dx}dx = \begin{cases} -\dfrac{1}{h}, & n = m \pm 1, \\ \dfrac{2}{h}, & n = m, \\ 0, & |n-m| > 1, \end{cases} \tag{2.50}$$

$$\int_0^1 \Psi_m(x)\Psi_n(x)\,dx = \begin{cases} \dfrac{h}{6}, & n = m \pm 1, \\ \dfrac{2h}{3}, & n = m, \\ 0, & |n-m| > 1, \end{cases} \tag{2.51}$$

$$\int_0^1 \Psi_m(x)\,dx = h. \tag{2.52}$$

Принимая во внимание выражения (2.50)–(2.52), метод конечных элементов приводит к дискретной модели (2.36) для расчета коэффициентов c_k, $k = 2, \ldots, N-1$, и данная модель формально является идентичной разностной модели (2.40). Как следствие, мы имеем $c_k = U_k$. Тем не менее, следует подчеркнуть еще раз, что, несмотря на формальное совпадение значений c_k, и U_k, их смысл остается различным. В методе конечных разностей решение имеет смысл приближенных значений искомого решения в узлах сетки, в то время как в методе конечных элементов решение представлено коэффициентами разложения по базису кусочно линейных функций. Кроме того, метод конечных элементов имеет более широкие возможности повысить точность приближенного решения. Например, если мы вычислим значение интеграла (2.49) аналитически (точно), то метод конечных элементов позволяет получить в рассмотренном случае практически точное решение дифференциальной задачи.

Программная реализация рассмотренного примера и результаты численных экспериментов представлены ниже (см. рис. 2.3).

```
%% Методы конечных разностей и конечных элементов
%% для решения краевой задачи
%% u''=-32(x-0.5)^4-4(x-0.5)^2
%% u(0)=u(1)=1.
%% Метод конечных элементов с линейными B-сплайнами
syms t v(t)
N=20; h = 1/(N-1);
x = 0:h:1;      %
x0 = x(2:end-1);  % Сетка
Nx = length(x0);
```

```
%Точное решение на сетке
v = dsolve(diff(v,2)==-32*(t-0.5).^4-4*(t-0.5).^2,v(0)==0,v(1)==0);
u=@(t) eval(v)
%Вычисление функции правой части
f = @(x)-32*(x-0.5).^4-4*(x-0.5).^2;
for m=1:Nx
b0(m) = int((-32*(t-0.5)^4-4*(t-0.5)^2)*(1+(t-x0(m))/h),x(m),x0(m))+...
int((-32*(t-0.5)^4-4*(t-0.5)^2)*(1-(t-x0(m))/h),x0(m),x(m+2));
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
bb=eval(b0);
e=ones(Nx,1);
A = spdiags([e,-2*e, e],-1:1,Nx,Nx)/h;
C=A\bb(:);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% FDM
A = spdiags([e,-2*e, e],-1:1,Nx,Nx)/h^2;
b= f(x0);
U=A\b(:);
subplot(1,2,1), plot(x,u(x),'.-','LineWidth',1)
xlabel('x'); ylabel('u(x)'); grid on
title('Точное решение')
subplot(1,2,2),...
semilogy(x0,abs(C-u(x0(:))),'.-',x0,abs(U-u(x0(:))),'og','LineWidth',1)
xlabel('x'); ylabel('\delta=|U(x_k)-u(x_k)|')
legend('FEM','FDM'); title('Погрешность'); grid
FDM_ERR=norm(U-u(x0(:)))/norm(u(x0(:)))
FEM_ERR=norm(C-u(x0(:)))/norm(u(x0(:)))
```

Применение разностных методов и метода конечных элементов при решении линейной краевой задачи для уравнения второго порядка приводит к дискретным моделям в виде системы линейных алгебраических уравнений с разреженной ленточной матрицей трехдиагонального вида. Такого рода системы могут быть эффективно реализованы с помощью метода Гаусса или его модификаций — метода прогонки. **Вычислительная сложность** реализации такой системы имеет порядок $O(N)$ вместо $O(N^3)$, в случае численного анализа линейных систем с полной матрицей.

При решении нелинейных дифференциальных задач соответствующая дискретная модель также будет нелинейной. В этом случае для анализа алгебраической задачи необходимо использовать подходящий итерационный метод, выбор которого чаще всего зависит от вида нелинейности исходной задачи.

Например, предположим, что коэффициент λ в уравнении (2.21) зависит от решения: $\lambda = \lambda(x, u)$. В этом случае дискретная модель будет аналогична линейной задаче (2.39), однако элементы матрицы будут зависеть от решения:

$$A(U)U = b. \tag{2.53}$$

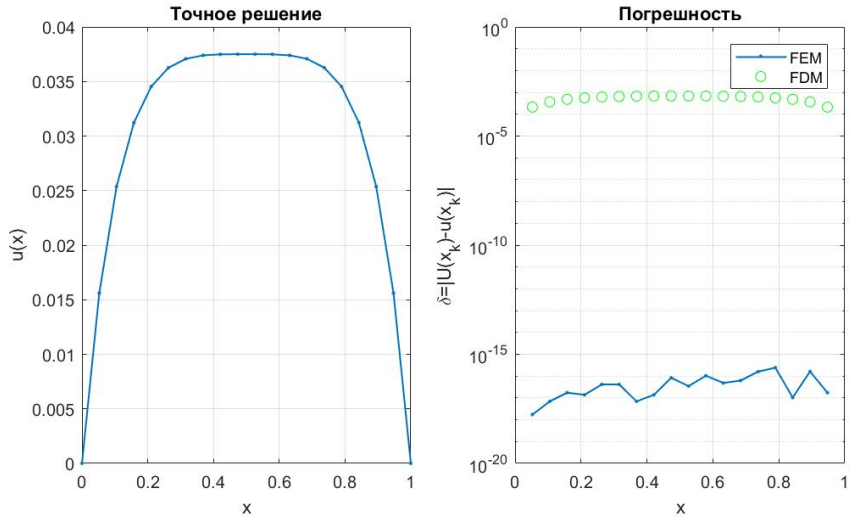


Рис. 2.3. Точное решение задачи (2.21)–(2.22) для следующих значений входных данных: $\lambda(x) = 1$, $q(x) = 0$, $f(x) = 32(x - 0.5)^4 - 4(x - 0.5)^2$ (слева) и погрешности метода конечных разностей (FDM) и конечных элементов (FEM) (справа).

Для решения такой нелинейной системы (2.53) во многих практически значимых случаях может быть эффективно использован следующий итерационный метод:

$$A(U^{(k)})U^{(k+1)} = b, \quad k = 1, 2, \ldots, \quad U^{(0)} = 0. \tag{2.54}$$

Возможные проблемы, связанные с отсутствием сходимости итераций (2.54), могут быть устранены с помощью введения дополнительного итерационного параметра и релаксационной процедуры:

$$A(\tilde{U}^{(k)})U^{(k+1)} = b, \quad \tilde{U}^{(k)} = (1 - \sigma)\tilde{U}^{(k-1)} + \sigma U^{(k)}, \quad 0 < \sigma \leqslant 1. \tag{2.55}$$

Подводя итог отметим наиболее примечательные особенности методов конечных разностей и конечных элементов. Прежде всего, на протяжении более половины столетия эти два подхода зарекомендовали себя как наиболее универсальные методы численного анализа дифференциальных краевых задач. Эффективность данных методов во многом связана со структурой получаемых при этом дискретных моделей в виде систем с разреженной матрицей ленточного типа, допускающей эффективное обращение. Как следствие, методы конечных разностей и конечных элементов во многом отвечают компромиссу между достаточно высокой точностью и умеренной вычислительной сложностью.

Эти методы легко обобщаются на случай неоднородных и адаптивных сеток, что важно при моделировании задач с неоднородными решениями, включая сингулярные режимы и пространственно-локализованные особенности поведения. Наконец, за последние десятилетия разработано большое число модификаций данных методов, которые позволяют улучшить их вычислительные характеристики как в общем

случае, так и в случае решения важных частных задач. Достоинства и недостатки этих двух классов методов могут быть кратко резюмированы следующим образом: если вам необходимо решить дифференциальную краевую задачу как можно быстрее и вы не имеете опыта в области численного анализа, тогда разностный метод, пожалуй, может быть лучшим выбором. Однако, если вы ищите мощный вычислительный аппарат, который позволит решить не только текущие проблемы, но способен обеспечить надежную поддержку при развитии и совершенствовании дифференциальной модели, тогда стоит обратить внимание на метод конечных элементов. Преимущества метода конечных элементов проявляется в еще большей степени при решении дифференциальных задач с частными производными.

Упражнение 2.1.

1. Оцените порядок аппроксимации разностной схемы для случая задачи, рассмотренной в примере выше, когда правая часть уравнения аппроксимируется следующим образом

$$b_n \simeq f(x_n) + \left(f(x_{n-1}) - 2f(x_n) + f(x_{n+1})\right)/12.$$

2. Проверьте ответ на вопрос упражнения 1 с помощью численного эксперимента, используя соответствующие изменения в программной реализации алгоритма для оценки фактического порядка точности метода. Сравните оценки точности и порядка аппроксимации схемы.

3. Вычислите значения компонент матрицы $\{A_{km}\}$ для метода конечных элементов (2.36)–(2.37) в случае краевой задачи (2.21)–(2.22) с коэффициентами $\lambda(x) = 1+x^2$ и $q(x) = 0$. Какие квадратурные формулы следует использовать, чтобы полученная в итоге схема конечных элементов была эквивалентна разностной схеме (2.28)?

4. Сравните время решения задачи с использованием метода конечных разностей и конечных элементов в рассмотренном выше примере, используя функции Matlab **tic** и **toc**. Какой из методов представляется более эффективным?

5. Для решения краевой задачи (2.21)–(2.22) выполняется следующее интегральное тождество:

$$\int_0^1 f\,dx = \lambda(x)\,\frac{du(x)}{dx}\bigg|_{x=1} - \lambda(x)\,\frac{du(x)}{dx}\bigg|_{x=0}. \tag{2.56}$$

Выведите дискретный аналог данного тождества для разностной схемы (2.29). Дискретный аналог подразумевает использование конечных разностей и квадратурных формул вместо соответствующих производных и интегралов в исходной непрерывной модели. Проверьте выполнение полученного тождества на основе численных экспериментов с использованием соответствующих дополнений в программную реализацию рассмотренного выше примера.

6. Предложите разностную аппроксимацию краевых условий Неймана:

$$\lambda(x)\frac{du(x)}{dx}\,|_{x=0} = \mu u(x)\,|_{x=0}\,, \quad \lambda(x)\frac{du(x)}{dx}\,|_{x=0} = \mu. \tag{2.57}$$

Одно из возможных решений состоит в использовании уравнения (2.23).

7. Постройте конечно-разностный и конечно-элементный методы для численного решения задачи:

$$\frac{1}{r}\frac{d}{dr}\left(r\frac{du}{dr}\right) = f(r), \quad x \in (0,1), \tag{2.58}$$

$$\left.\frac{du}{dr}\right|_{r=0} = 0, \quad u(1) = 0. \tag{2.59}$$

Используйте дискретизацию задачи на сетке с фиктивным узлом в точке $r = -h/2$, где h — размер шага сетки.

2.3. Спектральные методы

Идея спектральных методов состоит в том, что искомое приближенное решение представляется в виде линейной комбинации некоторого множества базисных бесконечно дифференцируемых функций $\psi_n(x)$, $n = 0, 1, 2, \ldots, N-1$:

$$u(x) \approx \tilde{u}(x) = \sum_{n=0}^{N-1} a(n)\psi_n(x), \quad x \in [a,b], \tag{2.60}$$

где коэффициенты $a(n)$ определяются таким образом, что разность между искомой функция $u(x)$ и ее приближенным представлением $\tilde{u}(x)$ была минимальной в определенном смысле. В отличие от метода конечных элементов, для спектральных методов не требуется финитности носителя базисных функций (сравните представления (2.60) и (2.20)).

Для **спектральных методов коллокации** искомое решение представляется в виде:

$$u(x) \approx \tilde{u}(x) = \sum_{n=0}^{N-1} \frac{\alpha(x)}{\alpha(x_n)}\hat{u}_n\phi_n(x), \quad x \in [a,b] \tag{2.61}$$

где $\alpha(x)$ — некоторая весовая функция, $u(x_n) = \tilde{u}(x_n) = \hat{u}_n$, а узлы интерполяции x_n, $n = 0, 1, \ldots, N-1$ называются **точками коллокации**: $a \leqslant x_0 < x_1 < \cdots < x_N \leqslant b$. В качестве множества базисных функций $\phi_n(x)$ наиболее предпочтительным представляется использование систем ортогональных алгебраических или тригонометрических полиномов, $(\phi_n(x), \phi_k(x)) = \delta_{nk}$, где $(\cdot,\cdot)$ и δ_{nk} — соответственно скалярное произведение и символ Кронекера:

$$(\phi_n(x), \phi_k(x)) = \int_a^b \eta(x)\phi_n(x)\phi_k(x)\,dx, \quad \delta_{nk} = \begin{cases} 1, & n = k, \\ 0, & n \neq k. \end{cases} \tag{2.62}$$

Здесь $\eta(x)$ — некоторая весовая функция. В случае ортогонального базиса коэффициенты $a(n)$ в разложении (2.60) вычисляются следующим образом

$$a(n) = (u(x), \phi_n(x))\,. \tag{2.63}$$

Для вычисления интегралов в скалярном произведении (2.63) могут быть использованы квадратурные формулы, определенные на множестве узлов сетки $x_k \in [a, b]$. Спектральные методы, использующие дискретное представление функций принято называть **псевдоспектральными**.

Типичным примером спектральных методов является **метод Фурье**, основанный на использовании тригонометрических базисных функций, периодических на некотором интервале $x \in [0, L]$:

$$\psi_n(x_k) = \exp\left(\frac{i2\pi n}{L} x_k\right), \quad x_k = kh, \quad k = 0, 1, 2, \ldots N-1, \quad h = L/N. \tag{2.64}$$

Отметим, что переход от сеточной функции $\tilde{u}(x_k)$ к коэффициентам Фурье $a(n)$ и обратно,

$$a(n) = \sum_{k=0}^{N-1} \tilde{u}(x_k) \exp\left(-\frac{i2\pi n}{L} x_k\right), \quad n = 0, 1, \ldots N-1, \tag{2.65}$$

$$\tilde{u}(x_k) = \frac{1}{N} \sum_{n=0}^{N-1} a(n) \exp\left(\frac{i2\pi n}{L} x_k\right), \qquad k = 0, 1, \ldots N-1, \tag{2.66}$$

известен как **прямое** и **обратное дискретное преобразование Фурье**. Данное преобразование может быть представлено как произведения матрицы преобразования Фурье на соответствующий вектор:

$$\mathbf{a} = F\mathbf{u}, \quad \mathbf{u} = F^{-1}\mathbf{a}, \tag{2.67}$$

где

$$F = \frac{1}{\sqrt{N}} \begin{bmatrix} \psi_{0,0} & \psi_{0,1} & \ldots & \psi_{0,N-1} \\ \psi_{1,0} & \psi_{1,1} & \ldots & \psi_{1,N-1} \\ \ldots & \ldots & \ldots & \ldots \\ \psi_{N-1,0} & \psi_{N-1,1} & \ldots & \psi_{N-1,N-1} \end{bmatrix}, \tag{2.68}$$

$$F^{-1} = \frac{1}{\sqrt{N}} \begin{bmatrix} \psi^{-1}_{0,0} & \psi^{-1}_{0,1} & \ldots & \psi^{-1}_{0,N-1} \\ \psi^{-1}_{1,0} & \psi^{-1}_{1,1} & \ldots & \psi^{-1}_{1,N-1} \\ \ldots & \ldots & \ldots & \ldots \\ \psi^{-1}_{N-1,0} & \psi^{-1}_{N-1,1} & \ldots & \psi^{-1}_{N-1,N-1} \end{bmatrix}, \tag{2.69}$$

$$\mathbf{u} = (\tilde{u}(x_0), \tilde{u}(x_1), \ldots, \tilde{u}(x_{N-1}))^T, \quad \mathbf{a} = (a(0), a(1), \ldots, a(N-1))^T, \tag{2.70}$$

$$\psi_{km} = \exp(-i2\pi km/N). \tag{2.71}$$

Постоянные $\omega_k = 2\pi k/L$ имеют смысл частот соответствующих базисных функций. Матрицы F и $F^{-1} = F^*$ — матрицы прямого и обратного дискретных преобразований Фурье. Вычисление прямого и обратного преобразований Фурье согласно (2.67) имеет вычислительную сложность $O(N^2)$.

Одно из замечательных свойств метода Фурье состоит в том, что дискретное преобразование Фурье может быть реализовано с вычислительными затратами порядка $O(N\log(N))$ при использовании **алгоритма быстрого преобразования Фурье БПФ** . В случае когда N равно целой степени двойки, $N = 2^m$, $m = 1, 2, 3, \ldots$, алгоритм БПФ может быть представлен в виде особого рода факторизации **циркулянтной матрицы**, каковой является матрица преобразования Фурье [1]:

$$F = S_m S_{m-1}, \cdots S_1, \tag{2.72}$$

где S_k — разреженные матрицы с не более чем двумя ненулевыми элементами в каждой строке, $k = 1, 2, \ldots m$, $2^m = N$. Как следствие, вычислительная сложность умножения $S_k\mathbf{u}$ имеет порядок $O(N)$. В итоге вычислительная сложность цепочки произведений $S_m S_{m-1}, \cdots S_1\mathbf{u}$ составляет $O(Nm) = O(N\log N)$.

Коэффициенты Фурье $a(k)$, равно как и преобразуемый вектор $u(x)$, полагаются периодическими:

$$u(x) = u(x + L), \quad a(k) = a(k + sN), \quad s = \pm 1, \pm 2, \ldots. \tag{2.73}$$

С учетом периодичности можно использовать произвольное множество из N последовательных значений в дискретном представлении функции $u(x_k)$ и ее коэффициентов Фурье $a(k)$, $k = m, m+1, \ldots, m+N-1$. Для многих приложений используется симметричная частотная полоса:

$$\tilde{u}(x_k) = \frac{1}{N}\sum_{n=-N/2}^{N/2-1} a(n)\exp\left(\frac{i2\pi n}{L}x_k\right), \qquad k = 0, 1, \ldots N-1. \tag{2.74}$$

Приближенное представление периодической функции (2.74) может рассматриваться как конечный отрезок ряда Фурье. Если функция $u(x)$ является достаточно гладкой (имеет p непрерывных производных) тогда коэффициенты Фурье быстро убывают

$$a(n) = O(n^{-(p+2)}). \tag{2.75}$$

Погрешность метода Фурье ассоциируется с **эффектом маскировки частот**, когда высокочастотные компоненты $a(k)$, $k \geqslant N/2$, прибавляется к компоненте $a(m)$ согласно соотношению (2.73), $m = k - N[k/N]$. Здесь $[k/N]$ означает целую часть отношения k/N. Для получения приемлемой точности частота дискретизации должна выбираться таким образом, чтобы коэффициенты Фурье за пределами заданного спектрального диапазона были пренебрежимо малы. Это может быть обеспечено выбором достаточно большого значения N.

[1] Специальный класс теплицевых матриц в которых каждая следующая строка (столбец) получается путем циркулярной перестановки предыдущей строки (столбца). Замечательные свойства циркулянтных матриц в численном анализе с связаны с возможностью приведения их к диагональному виду с помощью алгоритма быстрого дискретного преобразования Фурье

Дискретное преобразование Фурье с использованием алгоритма БПФ можно примять для численного дифференцирования. Действительно, как следует из уравнения (2.74)

$$\frac{d^m\tilde{u}(x)}{dx^m} = \frac{1}{N}\sum_{n=-N/2}^{N/2-1}\left(\frac{i2\pi n}{L}\right)^m a(n)\exp\left(\frac{i2\pi n}{L}x\right). \tag{2.76}$$

Таким образом, численное дифференцирование с использованием преобразования Фурье сводится к поэлементному умножению коэффициентов Фурье $a(n)$ на соответствующий вектор частот $i\omega_n$, $n = -N/2, -N/2-1, \ldots N/2-1$.

Используя разложение (2.61) оператор дифференцирования функции, заданной вектором значений на равномерной сетке, может быть представлен в виде матрицы $D^{(m)}$ размерности $N \times N$, которая называется **спектральной матрицей дифференцирования**

$$D^{(m)}_{k,n} = \frac{d^m}{dx^m}\left[\frac{\alpha(x)}{\alpha(x_n)}\phi_n(x)\right]_{x=x_k} \tag{2.77}$$

Процедура численного дифференцирования сводится к умножению спектральной матрицы дифференцирования на соответствующую сеточную функцию:

$$\mathbf{u}^{(m)} = D^{(m)}\mathbf{u}, \tag{2.78}$$

где $\mathbf{u} = (\tilde{u}_0, \tilde{u}_1, \ldots, \tilde{u}_{N-1})^T$, $\tilde{u}_k = \tilde{u}(x_k)$, $\mathbf{u}^{(m)} = (\tilde{u}_0^{(m)}, \tilde{u}_1^{(m)}, \ldots, \tilde{u}_{N-1}^{(m)})^T$, $\tilde{u}_k^{(m)} = \left[\dfrac{d^m\tilde{u}}{dx^m}\right]_{x=x_k}$.

Матрица спектрального дифференцирования произвольного порядка m в методе Фурье может быть вычислена следующим образом:

$$D^{(m)} = F^{-1}\Omega^m F, \tag{2.79}$$

где F и F^{-1} — матрицы прямого и обратного преобразования Фурье, Ω — диагональная матрица $N \times N$, такая, что $\Omega_{k,k} = i\omega_k$, $\Omega_{k,m\neq k} = 0$.

В отличие от методов конечных разностей и конечных элементов, где матрица оператора дифференцирования является разреженной ленточной, матрица спектрального дифференцирования является полной. Как следствие, вычислительная сложность спектральных методов существенно выше по сравнению с разностными и конечно элементными. Тем не менее, высокая вычислительная сложность спектральных моделей в случае достаточно гладких решений компенсируется высокой точностью, что обеспечивает в итоге преимущества данного класса методов. Кроме того, спектральные методы с матрицей дифференцирования теплицева типа при использовании БПФ алгоритма способны обеспечить непревзойденную эффективность.

Пример 2.2. Рассмотрим пример спектрального дифференцирования с использованием метода Фурье, применительно к функциям, имеющим различную степень гладкости:

$$u_1(x) = \exp(-16x^4), \quad u_2(x) = \begin{cases} \exp(-16x^2), & x \in [-\pi, 0], \\ \exp(-16x^4), & x \in [0, \pi]. \end{cases} \tag{2.80}$$

Отметим, что функция $u_1(x)$ является бесконечно дифференцируемой, в то время как функция $u_2(x)$ имеет лишь одну непрерывную производную. Результаты численных экспериментов и программная реализация спектрального дифференцирования Фурье представлены ниже.

```
% Погрешность спектрального метода дифференцирования Фурье
%  в зависимости от гладкости дифференцируемой функции
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
N = 128;
h = 2*pi/N;
x = -pi:h:pi-h; %Note, the last point is omitted
d = 1i*fftshift(-N/2:N/2-1);
%%%% гладкая функция u_1(x)%%%%%%%%%%%%%%%%%%%%%%%%
u1 = exp(-8*x.^4);
%%%% не гладкая функция u_1(x)%%%%%%%%%%%%%%%%%%%
u2(1:N/2) = exp(-8*x(1:N/2).^4);
u2(N/2+1:N) = exp(-8*x(N/2+1:N).^2);
%%% точные производные %%%%%%%%%%%%%%%%%%%%%%%%%%%
du1 = -32*x.^3.*exp(-8*x.^4);
du2(1:N/2) = -32*x(1:N/2).^3.*exp(-8*x(1:N/2).^4);
du2(N/2+1:N) = -16*x(N/2+1:N).*exp(-8*x(N/2+1:N).^2);
%%% Спектральные производные %%%%%%%%%%%%%%%%%%%%%%%
Du1 = ifft(d.*fft(u1));
Du2 = ifft(d.*fft(u2));
%%%%error of the spectral differentiation %%%%%%
err1 = abs(Du1-du1);
err2 = abs(Du2-du2);
m=1:4:length(x);
semilogy(x(m),err1(m),'o-b',x(m),err2(m),'.-r','LineWidth',1);
legend('гладкая функция','не гладкая функция');
xlabel('x'); ylabel('Погрешность производной')
axis([-pi,pi,1.e-15,1])
grid
```

Представленные на рис. 2.4 результаты показывают существенную зависимость погрешности спектрального дифференцирования Фурье от гладкости дифференцируемой функции. Для бесконечно дифференцируемой функции $u_1(x)$ погрешность спектрального дифференцирования достигает порога вычислительной погрешности на сравнительно грубой сетке $N \geqslant 128$. В случае функции $u_2(x)$ производные высшего порядка разрывные при $x = 0$. Как следствие, ошибка спектрального дифференцирования в последнем случае существенно выше и медленно убывает с уменьшением шага сетки.

Рассмотренный выше пример показывает высокую эффективность спектрального метода в случае достаточной гладкости искомого решения. Однако, если функция или ее производные разрывны, то спектральные методы теряют свои преимущества. Отметим, что в спектральном методе Фурье требования гладкости распространяются также на периодическое продолжение функции за пределы рассматри-

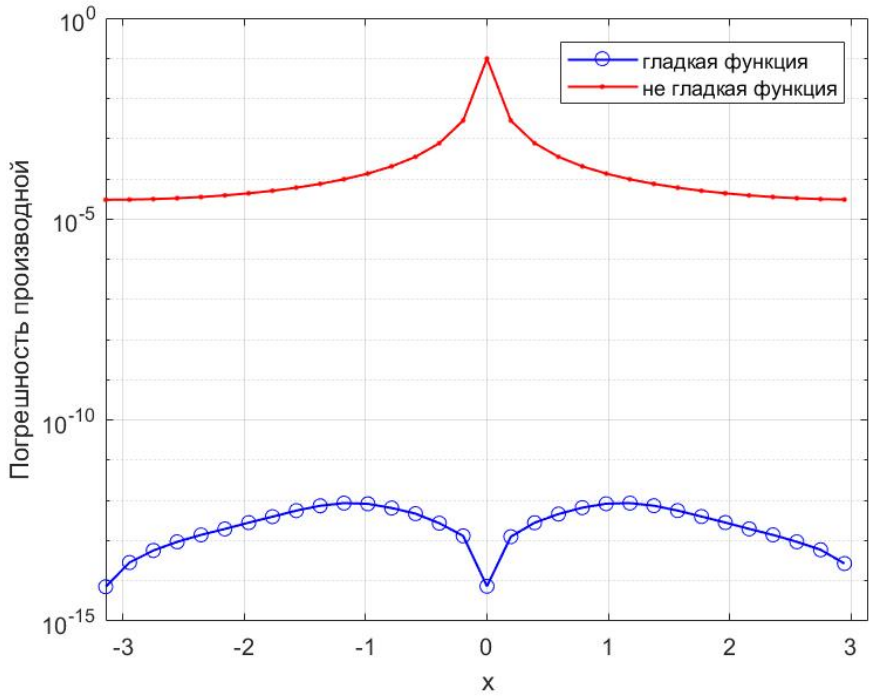


Рис. 2.4. Погрешности спектрального дифференцирования Фурье в случае гладкой и не гладкой функции при $N = 128$.

ваемого интервала. В силу последнего, метод Фурье может работать только с периодическими краевыми условиями или с финитными функциями, обращающимися в ноль вне некоторого интервала. Для нулевых краевых условий наиболее подходящим представляется спектральный метод, использующий синус преобразование Фурье.

Наряду с методом Фурье, один из наиболее эффективных методов решения дифференциальных задач является **спектральный метод Чебышева**, который может использоваться для произвольных краевых условий.

Матрица спектрального дифференцирования Чебышева вычисляется на основе интерполяционного представления (2.61) с весовой функцией $\alpha(x) \equiv 1$ и интерполяционными узлами, определяемыми точками экстремумов полинома Чебышева первого рода:

$$T_n(x) = \cos\left(n \arccos(x)\right), \quad x \in [-1, 1], \quad n = N - 1 :$$

$$x_k = -\cos\frac{(k-1)\pi}{N-1}, \quad k = 1, ..., N \tag{2.81}$$

Базисные функции определяются следующим образом:

$$\phi_k(x) = \frac{(-1)^k}{c_k}\frac{1-x^2}{(N-1)^2}\frac{T'_N(x)}{x-x_k}, \quad c_1 = c_N = 2, \quad c_2 = ... = c_{N-1} = 1. \tag{2.82}$$

Согласно (2.64), (2.82) элементы матрицы спектрального дифференцирования Чебышева вычисляются по следующим формулам:

$$D^{(1)}_{1,1} = \frac{2(N-1)^2+1}{6}, \tag{2.83}$$

$$D^{(1)}_{N,N} = -\frac{2(N-1)^2+1}{6}, \tag{2.84}$$

$$D^{(1)}_{k,k} = \frac{-x_k}{2(1-x_k^2)}, \quad k = 2, ..., N-1, \tag{2.85}$$

$$D^{(1)}_{k,n} = \frac{c_k}{c_n}\frac{(-1)^{k+n}}{(x_k - x_n)}, \quad k \neq n, \quad k, n = 1, ..., N, \tag{2.86}$$

Фактически, каждый элемент матрицы дифференцирования $D^{(1)}_{k,n}$ численно равен производной интерполяционного полинома $\phi_n(x)$ (2.82) в точке $x = x_k$. Для вычисления производных более высокого порядка с помощью матрицы спектрального дифференцирования Чебышева может быть использовано следующее тождество:

$$D^{(m)} = {D^{(1)}}^m. \tag{2.87}$$

Отметим, что для некоторого вида матриц дифференцирования формула (2.87), вообще говоря, не вполне подходит для вычисления производных высшего порядка. Например, матрица разностного дифференцирования $D^{(1)}_C$ имеет вид

$$D^{(1)}_C = \frac{1}{2h}\begin{bmatrix} 0 & 1 & 0 & & 0 \\ -1 & 0 & \ddots & \ddots & \\ 0 & \ddots & \ddots & \ddots & 0 \\ & \ddots & \ddots & 0 & 1 \\ 0 & & 0 & -1 & 0 \end{bmatrix}. \tag{2.88}$$

Однако, для аппроксимации второй производной более предпочтительным представляется использовать произведение матриц правой и левой разностных производных вместо квадрата матрицы центральной производной:

$$D^{(2)} = D^{(1)}_B D^{(1)}_F = D^{(1)}_F D^{(1)}_B, \tag{2.89}$$

где

$$D^{(1)}_B = \frac{1}{h}\begin{bmatrix} 1 & 0 & 0 & & 0 \\ -1 & 1 & \ddots & \ddots & \\ 0 & -1 & \ddots & \ddots & 0 \\ & \ddots & \ddots & 1 & 0 \\ 0 & & 0 & -1 & 1 \end{bmatrix} \quad D^{(1)}_F = \frac{1}{h}\begin{bmatrix} -1 & 1 & 0 & & 0 \\ 0 & -1 & \ddots & \ddots & \\ 0 & \ddots & \ddots & \ddots & 0 \\ & \ddots & \ddots & -1 & 1 \\ 0 & & 0 & 0 & -1 \end{bmatrix}. \tag{2.90}$$

Используя понятие матрицы дифференцирования мы получаем вполне прозрачное понимание общих принципов построения спектральных численных методов. В частности, построение спектральной дискретной модели состоит в замене дифференциального бесконечномерного оператора на спектральную матрицу дифференцирования, в которой учтены краевые условия.

В случае метода Фурье краевые условия полагаются периодическими, что автоматически учитывается выбором периодических базисных функций. Для спектрального метода Чебышева краевые условия произвольного вида могут быть заданы с помощью соответствующей модификации крайних строк матрицы дифференцирования. Рассмотрим простой пример, демонстрирующий постановку краевых условий.

Пример 2.3. Построим спектральный метод Чебышева для уравнения второго порядка с краевыми условиями Дирихле и исследуем его эффективность на задачах с различной степенью гладкости входных данных:

$$\frac{d^2u}{dx^2} = f(x), \quad x \in [-1, 1], \tag{2.91}$$

$$u(-1) = a, \quad u(1) = b. \tag{2.92}$$

В первом случае полагаем

$$f(x) = -x^2/12, \tag{2.93}$$

и решение задачи является гладким:

$$u(x) = x^4 - 1. \tag{2.94}$$

Во втором случае функция правой части полагается разрывной

$$f(x) = 50\,\mathrm{sign(x)}. \tag{2.95}$$

Как следствие, производные решения задачи второго и более высокого порядка также будут разрывными. Точное решение имеет вид

$$u(x) = 25\,\mathrm{sign(x)}\,\left(\mathrm{x}^2 - |\mathrm{x}|\right). \tag{2.96}$$

Дискретизация области определения решения и замена дифференциального оператора соответствующей матрицей дифференцирования приводит к следующей системе дифференциальных уравнений

$$D^{(2)}U = F, \tag{2.97}$$

где $F = (a, f(x_1), f(x_2), \ldots, f(x_{N-2}), b)^T$. Отметим, что первую и последнюю строки матрицы дифференцирования следует модифицировать согласно краевым условиям, чтобы решение алгебраической задачи (2.97) было определено однозначно. В случае нулевых краевых условий эти строки (равно как первый и последний столбцы матрицы $D^{(2)}$) могут быть просто удалены. В соответствии с этим удаляются первый и последний элементы вектора F.

Для краевых условий Дирихле общего вида элементы первой и последней строк полагаются равными нулю, за исключением диагональных элементов, значения которых задаются равными единице. При этом первый и последний элемент вектора правой части F задаются равными соответствующим краевым значениям.

В случае краевых условий Неймана, например,

$$\left(\frac{du}{dx} - \alpha u\right)\Big|_{x=-1} = \beta, \tag{2.98}$$

первая строка матрицы дифференцирования $D^{(2)}$, отвечающая граничной точке $x = -1$, заменяется соответствующей строкой матрицы $D^{(1)}$ с добавлением к диагональному элементу этой строки слагаемого $-\alpha$. Первому элементу вектора F соответственно присваивается значение β.

```
%% Решение уравнения u"=sign(x)*50 %%%%
%%% с разрывной правой частью %%%%%%%%
%% Решение уравнения u"=-12*x^2 %%%%
%%% с гладкой правой частью %%%%%%%%
%% Конечные разности в сравнении со спектральным методом %%
clear; k = 0;
for H = [1e-1,5e-2,2.5e-2,1e-2,5e-3,2e-3,1e-3]
k = k+1;
h = H;
x = -1+h:h:1-h;
N = length(x);
e = ones(N,1)/h^2;
%%%% конечноразностная матрица дифференцирования %%%%%%%%%%%%
A = spdiags([e,-2*e,e],-1:1,N,N);
f = -12*x.^2;    % Гладкое решение
%     f = 50*sign(x);% не гладкое решение
u = A\(f(:));
%%%% = точное решение =%%%%%%%%%%%%%%%%%%%%%%%%
U = -x.^4+1;                    % Гладкое решение
%     U =25*sign(x).*x.^2-25*x;   % не гладкое решение
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
j = 0:N-1;
xx = -cos(j.*pi/(N-1));
xx=(xx+100)-100;
%%%% матрица спектрального дифференцирования Чебышева %%%%%%%%
D = gallery('chebspec',N);
AA = D*D; %вторая производная
%%%% = Краевые Условия = %%%%%%%%%%%%%%%%%
AA(1,1) = 1;
AA(1,2:end) = 0;
AA(N,N) = 1;
AA(N,1:end-1)=AA(N,1:end-1)*0;
f = -12*xx.^2;    % Гладкое решение
%    f = 50*sign(xx);% не гладкое решение
f(end) = 0;
f(1) = 0;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
uu = AA \ f(:);
%%%% = точное решение =%%%%%%%%%%%%%%%%%%%%%%
UU = -xx.^4+1;                    % Гладкое решение
%   UU =25*sign(xx).*xx.^2-25*xx;   % не гладкое решение
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
err_fd(k) = norm(U(:)-u)/norm(U);
err_sc(k) = norm(UU(:)-uu)/norm(UU);
Ns(k) = N;
end
subplot(1,2,2), loglog(Ns,err_fd,'o-b',Ns,err_sc,'.-g','LineWidth',1)
legend('FD','SC')
title('Погрешности методов')
xlabel('N');
ylabel('||\delta||/||u||');
axis([10 4000 1e-17 1]);
grid
subplot(1,2,1), plot(xx,UU,'b','LineWidth',1)
title('Решение ур-я u"=-12x^2')
%  title('Решение ур-я u"=sign(x)*50')
xlabel('x'); ylabel('u(x)');
axis([-1 1 -0 1.1]);
grid
```

Результаты, представленные на рис. 2.5, показывают, что спектральный метод Чебышева демонстрирует очень высокую точность на относительно грубой сетке. Матрица спектрального дифференцирования Чебышева дает практически точные результаты при дифференцировании полиномиальной функции степени $(N-1)$. В этом случае погрешность ограничена только вычислительной погрешностью (погрешностью округления чисел с плавающей запятой).

Метод конечных разностей в данном случае имеет второй порядок точности. Для достижения точности, сравнимой с точностью спектрального метода, разностный метод требует очень подробную сетку с числом узлов порядка $N \simeq 10^5$.

Тем не менее, преимущества спектрального метода Чебышева могут проявляться только в случае достаточно гладкого решения. При численном решении задачи (2.91)–(2.92), (2.94) ситуация коренным образом меняется. Точное решение имеет вид кусочно квадратичной функции (2.96). Локальная погрешность метода конечных разностей определяется производной четвертого порядка от решения (см. (2.24)). Как следствие, на параболическом решении ошибка дискретизации разностной схемы обращается в ноль. Данный факт подтверждается результатами численного эксперимента, представленными на рис. 2.6.

Что касается спектрального метода, то в случае негладкого решения его погрешность достаточно велика и сравнима с погрешностью разностного метода при решении задачи (2.91)–(2.93) (сравните результаты, представленные на рис. 2.5 и рис. 2.6).

Рассмотренный пример показывает, что точность спектрального метода (в данном случае это метод Чебышева) существенно зависит от гладкости входных данных

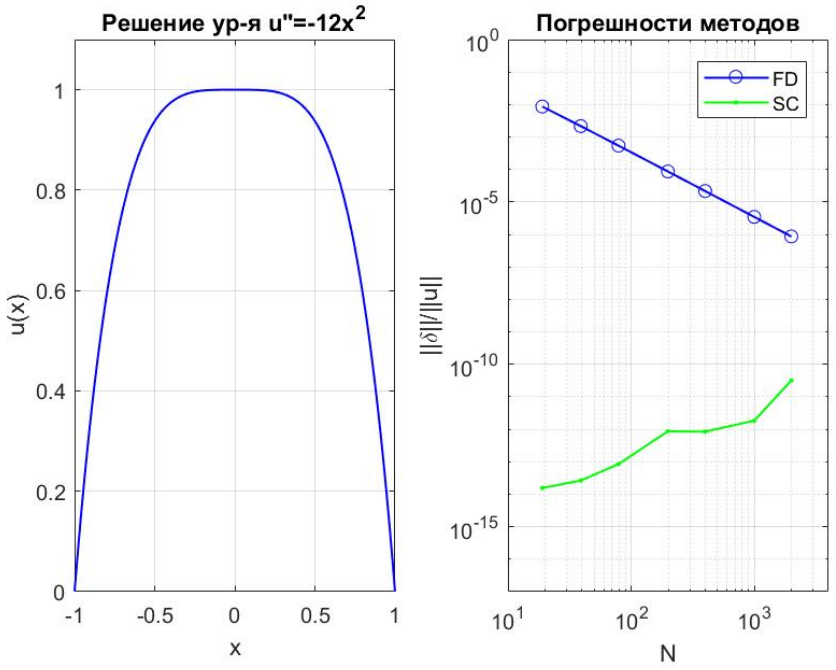


Рис. 2.5. Решение задачи (2.91)–(2.93) (слева) и динамика относительной ошибки спектрального метода Чебышева (SC) и метода конечных разностей в зависимости от числа узлов сетки (справа)

задачи. Как правило, погрешность численных методов характеризуется степенной скоростью сходимости:

$$\|\delta\| = \|U - u\| \leqslant CN^{-m}, \tag{2.99}$$

где C — некоторая постоянная, N — число узлов сетки, m — порядок скорости сходимости. Для большинства известных разностных методов и методов конечных элементов $m = 2$ или $m = 4$. Скорость сходимости спектральных методов существенно выше и в большинстве случаев ограничена не столько внутренними особенностями метода, сколько гладкостью решения. Для бесконечно дифференцируемых решений оценка (2.99) может выполняться для произвольных $m > 0$. Для решений, имеющих аналитическое продолжение на комплексную плоскость сходимость спектральных методов имеет экспоненциальный характер, т.е. быстрее, чем степенная сходимость (2.99):

$$\|\delta\| = \|U - u\| \leqslant \exp(-CN). \tag{2.100}$$

В случае функций, производные которых вплоть до порядка $p - 1$, принадлежат классу интегрируемых с квадратом, скорость сходимости ν-й производной, $\nu < p$, имеет следующую оценку:

$$\|u^{(\nu)} - D^{(\nu)}U\| \leqslant CN^{-(p-\nu+1)}, \tag{2.101}$$

для $p \geqslant 1$.

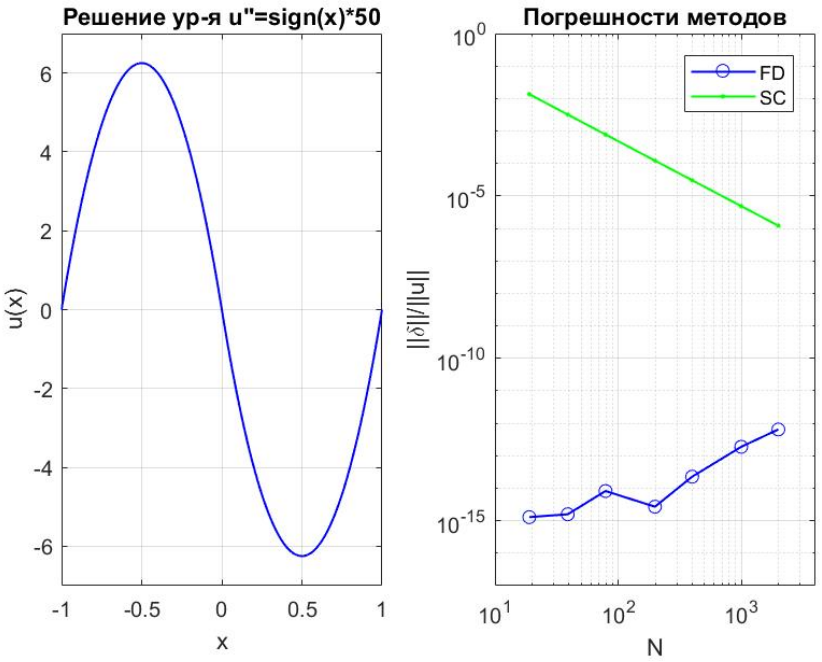


Рис. 2.6. Решение задачи (2.91)–(2.92), (2.94) (слева) и динамика погрешности спектрального метода Чебышева (SC) и метода конечных разностей (FD) в зависимости от числа узлов сетки (справа).

В случае задачи (2.91)–(2.92), (2.94), с учетом того, что вторая производная задана уравнением (2.95) и $p = 3$, согласно оценке (2.101) мы имеем второй порядок скорости сходимости, что подтверждается результатами численных экспериментов, представленных на рис. 2.3.

Для большинства практически значимых случаев оценка гладкости решения представляет собой нетривиальную задачу. В силу этого наиболее простой способ убедиться в сходимости спектрального метода состоит в сравнении приближенных решений, полученных на последовательности 2-3 сеток с различным числом узлов. Кроме того, существует несколько эмпирических правил, позволяющих оценить адекватный размер шага для получения требуемую точность. Например, при моделировании колебательных процессов разумно соизмерять выбор размера шага с периодом колебаний или длиной волны. В частности, наблюдения показывают, что для получения относительной погрешности 5% при моделировании волновых процессов со средней длиной волны λ необходимое пространственное разрешение сетки для методов конечных разностей второго порядка точности может быть обеспечено при размере шага дискретизации $h \simeq \lambda/20$, в то время как для спектральных методов Фурье (Чебышева) $h \simeq \lambda/3.5$. Для достижения относительной погрешности 1%, соответственно, требуется размер шага для разностных методов $h \simeq \lambda/40$, а для спектральных методов по-прежнему $h \simeq \lambda/3.5$ (подробнее см. [17]).

Еще одно эмпирическое правило выбора шага: при моделировании сингулярных решений на отрезке $x \in [-1,1]$, когда масштаб сингулярности $2\lambda << 1$, относительная точность порядка ε обеспечивается спектральным методом Чебышева при размере шага сетки

$$h \simeq \frac{\lambda}{2\log(\varepsilon)}. \tag{2.102}$$

Пример 2.4. Для лучшего понимания преимуществ спектральных методов сравним его эффективность с другими численными методиками, например, с методом сплайн-коллокации 5-го порядка точности, реализованным в стандартной функции MATLAB **bvp5c**.

В качестве тестовой задачи рассмотрим систему дифференциальных уравнений

$$\begin{cases} \dfrac{du}{dx} = ik\exp(i2\Delta x)v, \\ \\ \dfrac{dv}{dx} = ik\exp(-i2\Delta x)u, \end{cases} \quad , \quad x \in [-1,1], \tag{2.103}$$

где $i = \sqrt{-1}$, с краевыми условиями

$$u(-L) = a,\ v(L) = b. \tag{2.104}$$

Данная задача описывает распространение электромагнитных волн в среде с периодической модуляцией показателя преломления. Для оценки погрешности приближенного решение мы будем использовать аналитическое выражение для коэффициента отражения волны:

$$R = \frac{|v(-1)|}{|u(-1)|} = \frac{\sinh(\alpha L)}{\sqrt{\cosh^2(\alpha L) - \chi^2}} \tag{2.105}$$

где $\alpha = \sqrt{k^2 - \Delta^2}$, $\chi = \dfrac{\Delta}{k}$.

Используя сетку Чебышевских узлов

$$x_m = -\cos\frac{(m-1)\pi}{N-1}, \quad m = 1,2,\ldots,N, \tag{2.106}$$

построим дискретную модель на основе спектрального метода Чебышева, которая приводит к следующей системе линейных алгебраических уравнений:

$$AY = F, \tag{2.107}$$

где $Y = (U,V)^T$, $U = (u_0, u_2, \ldots u_{N-1})$, $V = (v_0, v_2, \ldots v_{N-1})$,

$$A = \begin{pmatrix} D^+ & G^+ \\ G^- & D^- \end{pmatrix}, \tag{2.108}$$

$G^{\pm}$ — диагональные матрицы $N \times N$:

$$g^+_{m,m} = ik\exp(i2\Delta x_m), \quad m = 2,3,\ldots,N, \quad g^+_{1,1} = 0, \tag{2.109}$$

$$g^-_{m,m} = ik\exp(-i2\Delta x_m), \quad m = 1, 2, \ldots, N-1, \quad g^-_{N,N} = 0, \tag{2.110}$$

$D^{\pm}$ — матрица спектрального дифференцирования Чебышева размерности $N \times N$, в которой первая (последняя) строка модифицирована в соответствии с граничными условиями: $d^+_{1,1} = d^-_{N,N} = 1$, $d^+_{1,m\neq 1} = d^-_{N,m\neq N} = 0$, а вектор правой части системы F с учетом краевых условий имеет вид: $F = (a, 0, \ldots, 0, b)^T$.

Программная реализация алгоритма спектрального метода и решение задачи с использованием функции **bvp5c** представлены ниже. Для более реалистичной оценки вычислительных затрат мы повторяли процедуру несколько раз, выбирая минимальное время решения.

```
%% Спектральный метод Чебышева и метод сплайн-коллокации
%%%% Сравнение эффективности методов %%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%     параметры задачи    %%%%%%%%%%%%%%%%%%%%%%%
k = 2;  d = 5;
a = 1;  b = 0;
alp = sqrt(k^2-d^2); Xi = d/k;
R0=sinh(alp*2)/sqrt((cosh(alp*2))^2-Xi^2);
%% некоторые вспомогательные функции. см >>help bvp5c %%%
dydx = @(x,y)[-1i*k*exp( 1i*2*d*x)*y(2);...
1i*k*exp(-1i*2*d*x)*y(1)];
res = @(ya,yb)[ya(1) - a; yb(2)-b ];
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
K = 12;
%%%% Задание точности метода для функции bvp5c %%%%%%%%%%%%%
RelTol = logspace(-2,-9,K);
Ns = round(44./RelTol.^(1/4));
for m=1:K
solinit = bvpinit(linspace(-1,1,Ns(m)),[1;0]);
options = bvpset('RelTol',RelTol(m));
%% процедура повторяется 10 раз и выбирается минимальное время  %
for mm=1:10
tic
sol = bvp5c(dydx,res,solinit,options);
Txx(mm) = toc;
end
R_cm(m) = abs(sol.y(2,1))./a;
T_cm(m) = min(Txx);
Err_cm(m)=abs(R0-R_cm(m))/R0;
end
%% Последовательность сеток спектрального метода %%
Ns=[18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 48, 64];
for m=1:K
N = Ns(m);
%%% Чебышевская сетка и спектральная матрица дифференцирования %%
x = -cos((0:N-1)*pi/(N-1));
C = gallery('chebspec',N);
```

```
%% процедура повторяется 55 раз и выбирается минимальное время  %
for mm=1:55
tic
%% 2N x 2N матрица спектральной задачи %
A = zeros(2*N);
G_p = -diag(1i*k*exp( i*2*d*x(:)));
G_m =  diag(1i*k*exp(-i*2*d*x(:)));
A =[ C, G_p; G_m, C];
%%%%%% краевые условия %%%%%%%%%%%%%%%%%%%%%%%%%%%%
A(1,:) = A(1,:)*0;      A(1,1) = 1;
A(end,:) = A(end,:)*0;  A(end,end) = 1;
F=zeros(2*N,1);         F(1) = a; F(end) = b;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Y = A\F;
Tx(mm) = toc;
end
R_sm(m) = abs(Y(N+1))./a;
T_sm(m) = min(Tx);
Err_sm(m) = abs(R0-R_sm(m))/R0;
end
loglog(T_cm,Err_cm,'.-b',T_sm(1:end),Err_sm(1:end),'o-r','Linewidth',1)
%title('Spectral Chebyshev vs. 5-th order Spline-Collocation' )
xlabel('Время [сек]');
ylabel('Относительная погрешность')
legend('bvp5c', 'СМЧ'); grid; %figure; spy(A);
```

Результаты численных экспериментов представлены на рис. 2.7. Сравнение вычислительных затрат для достижения заданной точности показывает, что спектральный метод Чебышева существенно превосходит по эффективности стандартные средства MATLAB для решения двухточечных краевых задач. Сетка с числом узлов $N = 32$ при использовании спектрального метода позволяет достичь предельно малой погрешности, сравнимой с вычислительной погрешностью. При одинаковых требованиях к точности время решения задачи при использовании метода сплай-коллокации более чем на три порядка превосходит данный показатель для спектрального метода.

Упражнение 2.2. 1. Воспроизведите пример, представленный выше и убедитесь, что решение задачи (2.103)–(2.104) имеет волнообразный вид и для заданных параметров $\lambda \simeq 2/3$. На примере рассмотренной задачи проверьте выполнения эмпирического правила выбора шага спектральных методов для достижения относительной погрешности 1-4%. Среднее значение шага считать равным $2/N$, где N — число узлов сетки.

2. Проверьте справедливость эмпирического правила (2.102) на примере спектрального дифференцирования функции

$$f(x) = \frac{\lambda^2}{\lambda^2 + x^2}, \quad x \in [-1, 1], \quad \lambda = 0.1, 0.05, 0.01, \tag{2.111}$$

используя матрицу спектрального дифференцирования Чебышева.

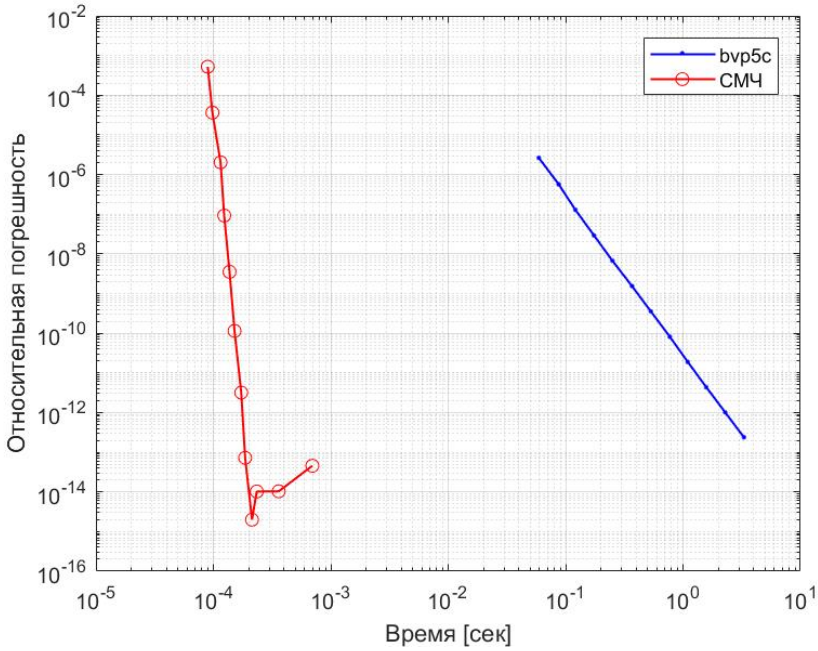


Рис. 2.7. Время решения задачи для достижения заданной точности при использовании спектрального метода Чебышева (СМЧ) и метода сплай-коллокации 5-го порядка точности, реализованного в функции **bvp5c.m**.

3. Для решения задачи (2.103)–(2.104) выполняется следующий закон сохранения:

$$\left[|u|^2 - |v|^2\right] = \text{const}. \tag{2.112}$$

Проверьте выполнение данного закона сохранения для спектрального метода и метода сплай-коллокации, используя программную реализацию методов, рассмотренную в примере выше.

4. Как можно объяснить рост ошибки разностного метода при уменьшении шага сетки в рассмотренном выше примере (см., рис. 2.6).

5. Какое минимальное число узлов сетки требуется для достижения предельной (спектральной) точности при дифференцировании полинома 6-го порядка с использованием матрицы спектрального дифференцирования Чебышева. Проверьте ответ с помощью прямых вычислений, приготовив соответствующий пример произвольного полинома 6-го порядка.

Глава 3

Численные методы для дифференциальных уравнений с частными производными

3.1. Метод конечных разностей

3.1.1. Нестационарное одномерное уравнение теплопроводности

В отличие от обыкновенных дифференциальных уравнений, дифференциальные уравнения с частными производными обеспечивают дополнительные возможности моделирования реальных прикладных задач, решения которых описываются функциями многих переменных. В большинстве случаев задачи для уравнений с частными производными могут рассматриваться как естественное обобщение соответствующих краевых задач и задач Коши для обыкновенных дифференциальных уравнений. Одним из типичных примеров такого типа обобщений может служить зависящее от времени уравнение теплопроводности, которое является нестационарным аналогом обыкновенного дифференциального уравнения (2.21), рассмотренного в предыдущей главе:

$$\frac{\partial u}{\partial t} - \frac{\partial u}{\partial x}\lambda(x)\frac{\partial u}{\partial x} - q(x,t)u = f(x,t), \quad (x,t) \in \Omega = [0,L] \times [0,T]. \tag{3.1}$$

Задачи для уравнения (3.1) подразумевают формулировку соответствующих **начальных** и **граничных условий**:

$$u(x, t=0) = u_0(x), \tag{3.2}$$

$$\left[\alpha_0(x,t)\frac{\partial u}{\partial x}+\beta_0(x,t)u+\gamma_0(x,t)\right]_{x=0}=0,$$
$$\left[\alpha_L(x,t)\frac{\partial u}{\partial x}+\beta_L(x,t)u+\gamma_L(x,t)\right]_{x=L}=0. \tag{3.3}$$

Метод конечных разностей основан на дискретизации непрерывной дифференциальной задачи путем замены производных соответствующими конечными разностями. Для этих целей в области определения решения строится сетка $\Omega_{h\tau}$ с узлами (x_k,t_n):

$$\Omega_{h\tau}=\{(x_k,t_n):\ x_k=x_{k-1}+h_k,\ k=1,2,\dots,K;\ t_n=t_{n-1}+\tau_n,\ n=0,1,\dots\} \tag{3.4}$$

Напомним, что расстояние между двумя соседними слоями сетки называется **шагом сетки**: $h_{k+1}=x_{k+1}-x_k$, $\tau_{k+1}=t_{n+1}-t_n$. Для простоты мы рассмотрим случай уравнения теплопроводности с постоянными коэффициентами,

$$\frac{\partial u}{\partial t}-\frac{\partial^2 u}{\partial x^2}=f(x),\quad (x,t)\in\Omega=[0,L]\times[0,T], \tag{3.5}$$

начальными условиями

$$u(x,t=0)=u_0(x), \tag{3.6}$$

и **краевыми условиями Дирихле**

$$u(0,t)=u(L,t)=0. \tag{3.7}$$

Ограничимся для простоты рассмотрением случая однородной сетки:

$$h_1=h_2=\dots=h;\quad \tau_1=\tau_2=\dots=\tau.$$

Для получения согласованной дискретной задачи для аппроксимации производных в уравнении (3.1) могут быть использованы различные наборы компактно расположенных узлов сетки, которые называются **сеточными шаблонами**. Наиболее типичные примеры шаблонов для аппроксимации уравнения (3.1) представлены на рис. 3.1. Далее будем использовать следующие двухиндексные обозначения U_k^n, для приближенного решения задачи: $U_k^n\simeq u(x_k,t_n)$.

Для аппроксимации производных в уравнении (3.5) используем стандартные выражения:

$$\frac{\partial u}{\partial t}=\frac{U_k^{n+1}-U_k^n}{\tau}+O(\tau),\quad \frac{\partial u}{\partial t}=\frac{U_k^{n+1}-U_k^{n-1}}{2\tau}+O(\tau^2), \tag{3.8}$$

$$\frac{\partial^2 u}{\partial x^2}=\frac{U_{k-1}^n-2U_k^n+U_{k+1}^n}{h^2}+O(h^2). \tag{3.9}$$

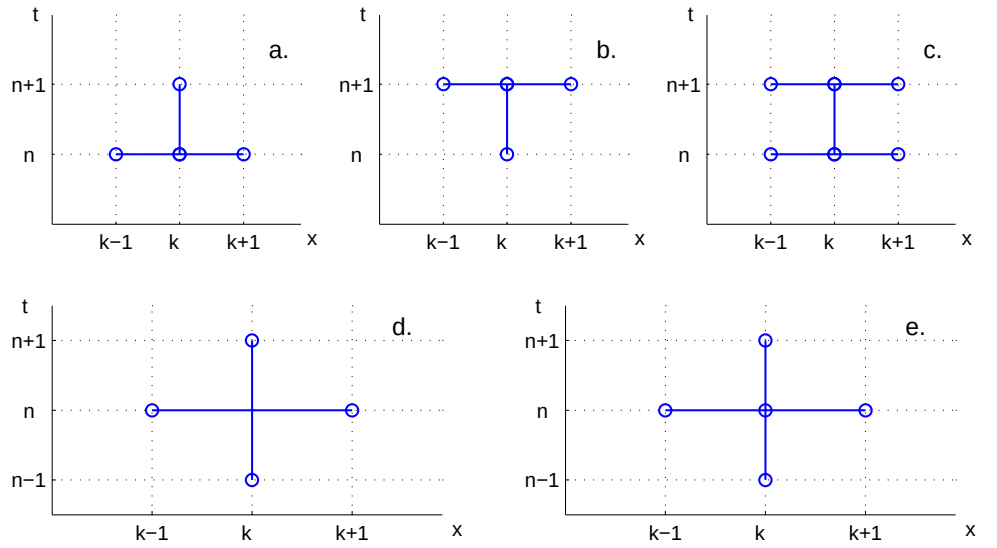


Рис. 3.1. Типовые сеточные шаблоны для аппроксимации нестационарного одномерного уравнения теплопроводности

Разностные схемы, соответствующие сеточным шаблонам, представленным на рис. 3.1, имеют следующий вид:

$$
\begin{aligned}
&\textbf{a.}\quad \frac{U_k^{n+1}-U_k^n}{\tau}-\frac{U_{k-1}^n-2U_k^n+U_{k+1}^n}{h^2}=f(x_k),\\
&\textbf{b.}\quad \frac{U_k^{n+1}-U_k^n}{\tau}-\frac{U_{k-1}^{n+1}-2U_k^{n+1}+U_{k+1}^{n+1}}{h^2}=f(x_k),\\
&\textbf{c.}\quad \frac{U_k^{n+1}-U_k^n}{\tau}-\Theta\frac{U_{k-1}^{n+1}-2U_k^{n+1}+U_{k+1}^{n+1}}{h^2}-(1-\Theta)\frac{U_{k-1}^n-2U_k^n+U_{k+1}^n}{h^2}=f(x_k),\\
&\textbf{d.}\quad \frac{U_k^{n+1}-U_k^{n-1}}{2\tau}-\frac{U_{k-1}^n-(U_k^{n-1}+U_k^{n+1})+U_{k+1}^n}{h^2}=f(x_k),\\
&\textbf{e.}\quad \frac{U_k^{n+1}-U_k^{n-1}}{2\tau}-\frac{U_{k-1}^n-2U_k^n+U_{k+1}^n}{h^2}=f(x_k),
\end{aligned}
\tag{3.10}
$$

Здесь $k=1,2,\ldots,N-1$, $0\leqslant\Theta\leqslant 1$. Разностная схема c. известна как **схема с весами** (в англоязычной литературе **Θ-метод**). Схема с весами при $\Theta=0$ эквивалентна **явной схеме** a., а при $\Theta=1$ — **чисто неявной схеме** b. Схема с весами при $\Theta=1/2$ более известна как **схема Кранка – Николсон** .

Шаблон схемы d. и e. является трехслойными. Как и в случае многошаговых методов для ОДЕ, **трехслойные схемы** d. и e. требуют, чтобы решение было известно на первых двух слоях. Вначале вычислений решение на первом слое задано

начальными условиями. Решение на втором слое может быть вычислено с помощью одной из схем a., b., или c.

Для прояснения различий между разностными схемами (3.10) рассмотрим вначале следующий численный пример.

Пример 3.1. Сравним неявную схему Кранка – Николсон c., $\Theta = 1/2$, с трехслойной **схемой Дюфорта – Франкела** d. на примере решения задачи (3.5), (3.7) с парамерами

$$f(x) = 0, \quad L = 2, \quad u_0(x) = \begin{cases} 1 - 2|x-1|, & |x-1| \leqslant 1/2, \\ 0, & |x-1| > 1/2. \end{cases} \tag{3.11}$$

Решение схемы Кранка – Николсон (3.10) c., $\Theta = 1/2$, может быть выражено следующим образом:

$$U^{n+1} = (2E - \tau A)^{-1}(2E + \tau A)U^n, \tag{3.12}$$

где E — единичная матрица, A — трехдиагональная матрица разностной производной второго порядка, вида (2.40), $U^k = (U_1^k, U_2^k \ldots, U_{N-1}^k)$. При реализации схемы Дюфорта – Франкела первый шаг мы выполним по схеме Кнанка – Николсон. Для последующих шагов решение трехслойной схемы находится по явным формулам:

$$U^{n+1} = \frac{2\tau h^2}{2\tau + h^2}\left(\frac{h^2 - 2\tau}{2\tau h^2}U^{k-1} + SU^k\right), \tag{3.13}$$

где S — двухдиагональная матрица следующего вида:

$$S = \frac{1}{h^2}\begin{bmatrix} 0 & 1 & & & & & \\ 1 & 0 & 1 & & & & \\ & \ddots & \ddots & \ddots & & & \\ & & & & & & \\ & & & & & & \\ & & 0 & & \ddots & \ddots & \ddots \\ & & & & & 1 & 0 & 1 \\ & & & & & & 1 & 0 \end{bmatrix} \tag{3.14}$$

Таким образом, решение в методе Дюфорта – Франкела (за исключением первого шага) находится явно, в то время как для схемы Кранка – Николсон на каждом шаге требуется решать систему линейных алгебраических уравнений с трехдиагональной матрицей $D = 2E - \tau A$. Программная реализация рассмотренных разностных алгоритмов и результаты численных экспериментов представлены ниже.

```
%% Сравнение разностных методов Кранка - Николсон и Дюфорта-Франклина
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
tau = 0.01; h = 0.05; %шаги сетки
T = 0.05
x =  h:h:2-h;
N = length(x);
%%%% начальные условия %%%%%%%%%%%%%%%%%%%%
```

```
u0 = 1-2*abs(x(:)-1);
m = find(u0<0);
u0(m) = u0(m)*0;
e = h^(-2)*ones(N,1);
E = speye(N,N);
A = spdiags([e, -2*e, e], -1:1, N,N);
%% схема Кранка - Николсон %%%%%%%%%%%%%%%%%%%%
D = 2*E-tau*A;
Ds = 2*E+tau*A;
u = u0;
for t = tau:tau:T
u = D\(Ds*u);
end
%% Схема Дюфорта-Франклина %%%%%%%%%%%%%%%%%%%%
w = 2*tau*h^2/(h^2+2*tau);
s = 1/(2*tau)-1/h^2;
u1 = u0;
%%%% первый шаг по схеме Кранка -Николсон  %%%%%%%%%%
u2 = D\(Ds*u1);
%%% u1,u2,u3 -- решения на трех слоях %
D2=spdiags([e,  e], [-1,1], N,N);
for t = 2*tau:tau:T
u3 = w*(s*u1+D2*u2);
u1 = u2;
u2 = u3;
end
plot(x,u0,'k:',x,u,'bo-',x,u2,'r.-','LineWidth',1);
xlabel('x'); ylabel('U(x)'); axis([0,2,0,1.1]);
legend('Нач. усл.', 'КН', 'ДФ');
title([' t = ',num2str(t,3), '   \tau = ',num2str(tau,3)])
grid
```

Как можно заметить из результатов численного эксперимента, представленных на рис. 3.2, несмотря на отсутствие гладкости начальных данных, схема Кранка – Николсон обеспечивает хорошую точность решения, в то время как схема Дюфорта Франкела при тех же шагах сетки приводит к существенной погрешности в случае $\tau \simeq h$. Теоретическое объяснение полученных закономерностей в поведении ошибки схемы Дюфорта – Франкела будет дано ниже при рассмотрении вопросов согласованности и устойчивости разностных схем.

Упражнение 3.1. 1. Используя программную реализацию метода Дюфорта – Франкела вычислите приближенное решени рассмотренной в примере задачи при различных значениях шага: $h = 1.e-2$; $1.e-3$; $1.e-4$. Убедитесь, что решение схемы Дюфорта – Франкела не сходится при $\tau = h$.

2. Внося необходимые изменения в программную реализацию примера, замените схему Дюфорта – Франкела явной разностной схемой а. (3.10). Попытайтесь

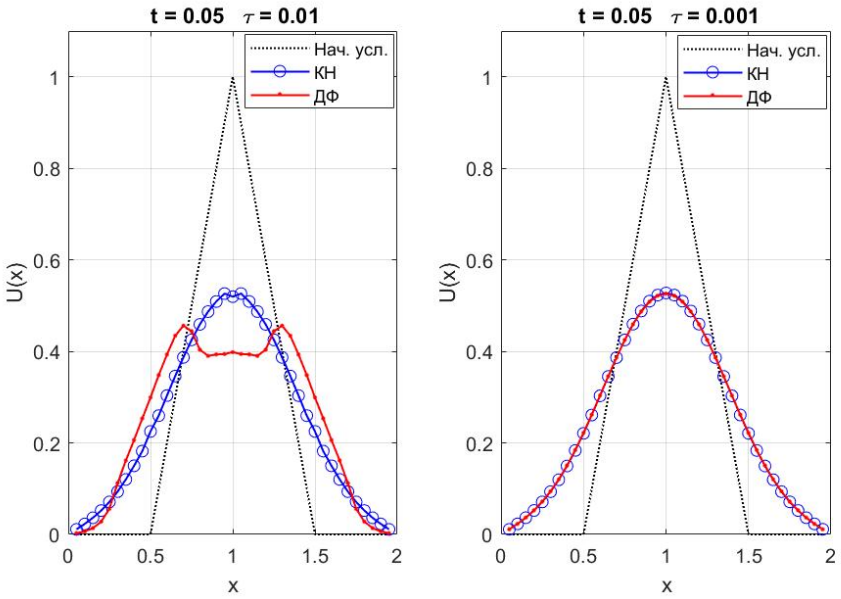


Рис. 3.2. Приближенные решения задачи (3.5), (3.7), (3.7) полученные по схеме Дюфорта–Франкела (ДФ) и Кранка–Николсон (КН) при $h = 0.05$ и различных значениях шага τ

определить при каких соотношениях шагов сетки τ to h решение явной разностной схемы сходится.

3. Внося необходимые изменения в программную реализацию примера, замените схему Дюфорта – Франкела явной трехслойной схемой е. (3.10). Убедитесь, что данная схема не устойчива и не сходится при любых соотношениях шагов τ и h.

4. Вычисляя разность между результатами численного моделирования, полученных с различными значениями шагов сетки τ и h, оцените скорость сходимости схемы с весами при

$$\Theta = 1; \quad \Theta = 0.5; \quad \Theta = 0.5 - \frac{h^2}{12\tau}.$$

Напомним, что **скорость сходимости** определяется постоянными s и q, такими, что при достаточно малых τ и h,

$$\|\delta\| = \|U - u\|_{t=T} = O(h^s + \tau^q).$$

Для оценки скорости сходимости можно воспользоваться апостериорными оценками погрешности (см., например, выражение (1.29)). В случае двух независимых переменных, вначале следует оценить сходимость относительно шага h полагая $\tau << h^2$, а после этого повторяем процедуру для шага τ при $h << \tau^2$.

3.1.2. Линейное уравнение переноса

Уравнение переноса — уравнение в частных производных, описывающее механизм консервативного транспорта материи или энергии. В одномерном случае данное уравнение имеет вид

$$\frac{\partial u}{\partial t} + v\frac{\partial u}{\partial x} = 0, \quad (x,t) \in \Omega = [0,L] \times [0,T], \tag{3.15}$$

и рассматривается совместно с начальными и граничными условиями:

$$u(x, t=0) = u_0(x), \quad u(0,t) = u_b(t). \tag{3.16}$$

Заметим, что **волновое уравнение** второго порядка

$$\frac{\partial u^2}{\partial t^2} - v^2\frac{\partial u^2}{\partial x^2} = F(u,x,t) \tag{3.17}$$

может быть представлено в эквивалентном виде посредством системы двух уравнений первого порядка:

$$\frac{\partial u}{\partial t} + v\frac{\partial u}{\partial x} = w, \tag{3.18}$$

$$\frac{\partial w}{\partial t} - v\frac{\partial w}{\partial x} = F(u,x,t). \tag{3.19}$$

Одно из примечательных свойств уравнения переноса (3.15) состоит в том что его решение остается постоянным вдоль **характеристической прямой**, уравнение которой имеет вид:

$$x = x_0 + vt. \tag{3.20}$$

Как следствие, любое решение уравнения (3.15) описывает движение начального профиля, заданного условием (3.16) со скоростью v:

$$u(x,t) = \begin{cases} u_0(x - vt), & t \leqslant x/v, \\ u_b(t - x/v), & t > x/v, \end{cases} \tag{3.21}$$

Во вногих приложениях механизм переноса входит в уравнения как составная часть более сложной модели. Тем не менее, вычислительные аспекты таких моделей могут быть достаточно полно проанализированы на примере такой упрощенной задачи как (3.15),(3.16).

Для аппроксимации производных в уравнении (3.15) используем следующие формулы разностного дифференцирования:

$$\frac{\partial u}{\partial x} = \frac{U_k^n - U_{k-1}^n}{h} + O(h), \tag{3.22}$$

.

$$\frac{\partial u}{\partial x} = \frac{U_{k+1}^n - U_k^n}{\tau} + O(h), \tag{3.23}$$

.

$$\frac{\partial u}{\partial x} = \frac{U_{k+1}^n - U_{k-1}^n}{\tau} + O(h^2) \tag{3.24}$$

.

$$\frac{\partial u}{\partial t} = \frac{U_k^{n+1} - U_k^n}{\tau} + O(\tau). \tag{3.25}$$

.

Типовые шаблоны для разностной аппроксимации уравнения переноса представлены на рис. 3.3.

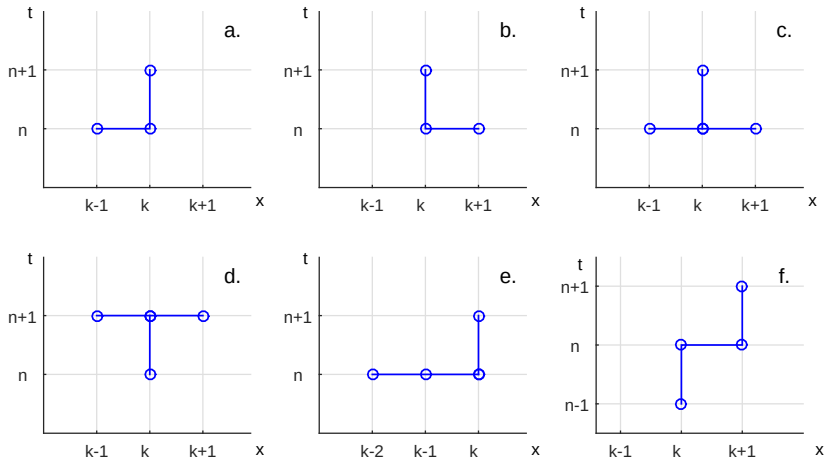


Рис. 3.3. Типовые шаблоны разностных схем для уравнения переноса

Разностные схемы, соответствующие рассмотренным сеточным шаблонам, (рис. 3.3) имеют следующий вид:

Схема с разностями против потока:

$$\begin{aligned} &\frac{U_k^{n+1} - U_k^n}{\tau} + v\frac{U_k^n - U_{k-1}^n}{h} = 0, \qquad v > 0, \\ &\frac{U_k^{n+1} - U_k^n}{\tau} + v\frac{U_{k+1}^n - U_k^n}{h} = 0, \qquad v < 0, \end{aligned} \tag{3.26}$$

Центрированная явная схема:

$$\frac{U_k^{n+1} - U_k^n}{\tau} + v\frac{U_{k+1}^n - U_{k-1}^n}{2h} = 0, \tag{3.27}$$

Центрированная неявная схема:

$$\frac{U_k^{n+1} - U_k^n}{\tau} + v\frac{U_{k+1}^{n+1} - U_{k-1}^{n+1}}{2h} = 0, \tag{3.28}$$

Схема Лакса – Фридрихса :

$$\frac{U_k^{n+1} - 0.5\left(U_{k+1}^n + U_{k-1}^n\right)}{\tau} + v\frac{U_{k+1}^n - U_{k-1}^n}{2h} = 0, \tag{3.29}$$

Схема Лакса – Вендроффа :

$$\frac{U_k^{n+1}-U_k^n}{\tau}+v\frac{U_{k+1}^n-U_{k-1}^n}{2h}-v^2\tau\frac{U_{k-1}^n-2U_k^n+U_{k+1}^n}{2h^2}=0, \tag{3.30}$$

Схема Beam–Warming :

$$\frac{U_k^{n+1}-U_k^n}{\tau}+v\frac{3U_k^n-4U_{k-1}^n+U_{k-2}^n}{2h}-v^2\tau\frac{U_k^n-2U_{k-1}^n+U_{k-2}^n}{2h^2}=0. \tag{3.31}$$

Схема КАБАРЕ:

$$\frac{1}{\tau}\left(\frac{U_{k+1}^{n+1}+U_k^n}{2}-\frac{U_{k+1}^n+U_k^{n-1}}{2}\right)+v\frac{U_{k+1}^n-U_k^n}{h}-=0, \tag{3.32}$$

Здесь $U_k^n=U(x_k,t_n)$, $(x_k,t_n)\in\Omega_{h\tau}$.

Отдельные из приведенных схем имеют весьма замысловатый вид. Например, в схеме Лакса – Вендроффа можно заметить выражение, напоминающее аппроксимацию производной второго порядка, хотя такие производные не входят в исходное дифференциальное уравнение. Смысл такого рода конструкций будет рассмотрен в следующих разделах при обсуждении вопроса согласованности и устойчивости разностных схем.

Для прояснения некоторых ключевых свойств разностных методов для уравнения переноса уместно начать с простого примера.

Пример 3.2. Рассмотрим некоторые из приведенных выше разностных схем применительно к решению задачи (3.15), (3.16) в случае не гладких начальных условий в виде треугольной ударной волны. Сравним результаты численного решения, полученные при использовании схемы с разностью против потока, схем Лакса – Фридрихса и Лакса – Вендроффа. Все рассматриваемые схемы являются явными и реализуются достаточно просто. Для схемы с разностью против потока мы имеем

$$U_k^{n+1}=U_k^n-C(U_k^n-U_{k-1}^n), \tag{3.33}$$

где, $C=\dfrac{\tau}{h}v$, $k=1,2,\ldots,K-1$, $n=1,2,\ldots$. Для схем Лакса – Фридрихса и Лакса – Вендроффа алгоритм решения выглядит во многом аналогично:

$$U_k^{n+1}=\frac{1}{2}\left(U_{k+1}^n+U_{k-1}^n\right)-\frac{C}{2}\left(U_{k+1}^n-U_{k-1}^n\right), \tag{3.34}$$

$$U_k^{n+1}=U_k^{n+1}-\frac{C}{2}\left(U_{k+1}^n-U_{k-1}^n\right)-\frac{C^2}{2}\left(U_{k+1}^n-2U_k^n+U_{k-1}^n\right). \tag{3.35}$$

Реализация алгоритмов (3.33)–(3.35) и результаты численных экспериментов представлены ниже.

```
%% Метод конечных разностей %%%%%%%%%%%%%%%
%% для линейного уравнения переноса %%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
L = 2;  T = 1.0;  v = 1;
```

```
tau = 0.05; h = 0.05;          %step sizes
x =  0:h:L; N = length(x);     %grid
%%%% начальные условия %%%%%%%%%%%%%%%%%%%
u_0 = 2*x(:);
m = find(u_0>1);
u_0(m) = u_0(m)*0;
u_1=u_0; u_10=u_0;
u_2=u_0; u_20=u_0;
u_3=u_0; u_30=u_0;
c = v*tau/h;        % число Куранта
for t=tau:tau:T
%% Схема с разностями против потока %%%%%%%%%%
for k=2:N-1
u_1(k)= u_10(k)-c*(u_10(k)-u_10(k-1));
end
u_10 = u_1;
%% Схема Лакса - Фридрихса %%%%%%%%%%%%%%%%%%
for k=2:N-1
u_2(k) = (u_20(k+1)+u_20(k-1))/2 ...
- c*(u_20(k+1)-u_20(k-1))/2;
end
u_20 = u_2;
%% Схема Лакса Вендроффа %%%%%%%%%%%%%%%%%%%
for k=2:N-1
u_3(k) = u_30(k)-c*(u_30(k+1)-u_30(k-1))/2 ...
+c^2*(u_30(k+1)-2*u_30(k)+u_30(k-1))/2;
end
u_30 = u_3;
end
plot(x,u_0,'k:',x,u_1,'bp-',x,u_2,'ro-',x,u_2,'g.-','LineWidth',1);
xlabel('x'); ylabel('U(x)'); axis([0,2,0,1.5]);
legend('Нач. усл.', 'РПП', 'ЛФ','ЛВ');
title(['Число Куранта: C= v\tau/h=',num2str(v*tau/h,3)])
grid
```

Наиболее примечательное, что бросается в глаза на рис. 3.4, это фантастически точное решение, полученное при параметрах дискретизации, удовлетворяющих условию $C = v\tau/h = 1$, где безразмерная величина C называется **число Куранта** (названная в честь Ричарда Куранта, 1888–1972 гг.). При $C < 1$ точность всех рассмотренных методов существенно хуже. Результаты, полученные с помощью схемы с разностями против потока представляются более корректными по сравнению с остальными схемами.

Упражнение 3.2. 1. Покажите, что разностные схемы с разностями против потока, Лакса – Фридрихса и Лакса – Вендроффа являются точными на решении задачи (3.15),(3.16) при $\tau = h/v$.

2. Пересчитайте представленный выше пример и сравните результаты расчетов при $\tau = h = 0.02$, $v = 0.9$; $v = 1.1$; $v = 1.2$.

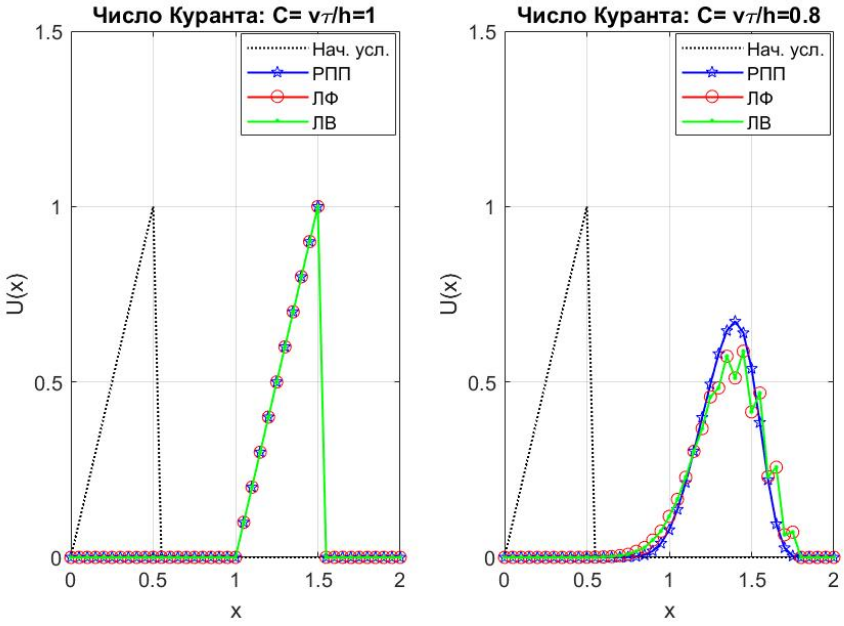


Рис. 3.4. Решение задачи (3.15), (3.16), полученное с помощью схемы с разностями против потока(РПП), Лакса – Фридрихса (ЛФ) и Лакса – Вендроффа (ЛВ) при $h = 0.05$ и различными значениями шага τ.

3. Реализуйте программно схему Beam–Warming и сравните результаты расчетов на основе этой схемы и методов, рассмотренных в примере при $\tau = h = 0.02$, варьируя значение параметра $v = 0.9;\quad v = 1.1;\quad v = 1.2$.

4. Реализуйте программно схему КАБАРЕ (3.32). На основе численных экспериментов определите соотношение шагов τ и h при которых схема является точной на решении задачи (3.15), (3.16).

3.1.3. Согласованность, устойчивость и сходимость. Теорема Лакса

Ключевой вопрос относительно любого приближенного численного метода состоит прежде всего в оценке его точности. Погрешность разностного метода оценивается на основе разности между приближенным и точным решением дифференциальной задачи, причем последнее в общем случае неизвестно. Оценка погрешности может быть получена на основе численных экспериментов или теоретически. Теоретические (априорные) оценки погрешности разностных схем для линейных дифференциальных задач основываются на исследованиях **согласованности (аппроксимации)** и **устойчивости** дискретной модели.

Рассмотрим линейную разностную схему

$$L_{h,\tau}U = f_h, \tag{3.36}$$

$$U = U^n = (U_1^n, U_2^n, \ldots, U_K^n)^T, \quad U^0 = u_0(x_k), \quad k = 1, 2, \ldots, K,$$

для дифференциальной задачи

$$Lu = f, \tag{3.37}$$

$$u = u(x,t), \quad f = f(x,t), \quad x \in [0, X], \quad t \in [0, T], \quad u(x,0) = u_0(x).$$

Здесь L и $L_{h,\tau}$ – линейный дифференциальный оператор и его разностная аппроксимация, соответственно. Напомним, что понятие устойчивости численного метода определяется как непрерывная зависимость его решения от входных данных, что равносильно выполнению следующей оценки:

$$\|U^n\| \leqslant c_1 \|U^0\| + c_2 \max_{t \in [0,T]} \|f\|, \tag{3.38}$$

где c_1 и c_2 некоторые постоянные, не зависящие от n, h и τ.

Мы будем говорить, что разностная схема **сходится**, если норма погрешности, которая определяется разностью между точным и приближенным решениями в узлах сетки, стремится к нулю при $\tau \to 0$, $h \to 0$:

$$\lim_{\tau, h \to 0} \|U - u\| = 0. \tag{3.39}$$

Как правило, норма погрешности связана с шагами дискретизации τ и h оценками вида

$$\|U - u\| = O(h^s + \tau^p), \quad \text{или} \quad \|U - u\| < C_1 h^s + C_2 \tau^p, \tag{3.40}$$

где числа s и p характеризуют **скорость сходимости** или **порядок точности**, C_1 и C_2 — постоянные, зависящие от производных высших порядков от решения.

Понятие **согласованности** (аппроксимации) разностной схемы означает, что для любого достаточно гладкого решения дифференциальной задачи $u(x,t)$

$$L_{h,\tau} u - Lu - f_h + f = L_{h,\tau} u - f_h = \psi(h,\tau) \to 0, \quad \text{при} \quad h, \tau \to 0. \tag{3.41}$$

Величину $\psi(h,\tau)$ обычно называют **погрешностью аппроксимации** или **невязкой** разностной схемы. Если $u(x,t)$ — решение дифференциальной задачи, то погрешность аппроксимации является невязкой, которая получается при подстановке в разностную схему (3.36) решения дифференциальной задачи $u(x_k, t_n)$ вместо разностного решения U_k^n.

Обычно величина $\psi(h,\tau)$ находится из представления решения дифференциальной задачи в точках шаблона сетки в виде степенного ряда по степеням h и τ. Главные члены невязки при подстановке решения в виде отрезков ряда в разностную схему выражает порядок погрешности метода. Как правило, для согласованной разностной задачи погрешность аппроксимации имеет вид:

$$|\psi(h,\tau)| = O(h^g) + O(\tau^q) \leqslant Q_1 h^g + Q_2 \tau^q. \tag{3.42}$$

где Q_1 и Q_2 — постоянные, зависящие от производных высших порядков от решения дифференциальной задачи, числа g и q характеризуют **порядок аппроксимации** разностной схемы.

Следует подчеркнуть, что погрешность аппроксимации разностной схемы $\psi(h,\tau)$ является фактическим источником погрешности приближенного решения и

является ключевым фактором в ее оценке. В частности, как следует из (3.36), (3.37), (3.41) погрешность разностного решения $\delta = U - u$ удовлетворяет разностному уравнению (3.36) с правой частью $\psi(,\hat{\tau})$ вместо f:

$$L_{h,\tau}\delta = -\psi(h,\tau), \quad U^0 = 0. \tag{3.43}$$

Если разностная схема (3.36) устойчива, то, как следует из (3.38),

$$\|\delta^n\| \leqslant c_2 \max_{1\leqslant m<n} \|\psi^m\|. \tag{3.44}$$

Если разностная схема (3.36) является согласованной и для погрешности аппроксимации имеет место (3.42), тогда для погрешности разностного решение выполняется оценка:

$$\|\delta^n\| \leqslant c_2 Q_1 h^s + c_2 Q_2 \tau^q. \tag{3.45}$$

Следовательно, если линейная разностная схема устойчива и согласована с соответствующей дифференциальной задачей, тогда приближенное разностное решение сходится при $\tau, h \to 0$, и скорость сходимости совпадает с порядком аппроксимации схемы. Данный факт известен как **теорема Лакса**.

Таким образом, чтобы оценить скорость сходимости разностной схемы для решения линейной дифференциальной задачи достаточно установить факт устойчивости и оценить порядок аппроксимации разностной схемы.

Пример 3.3. Рассмотрим методику анализа согласованности разностной схемы на примере схемы с разностями против потока (3.26) для линейного уравнения переноса. Для оценки погрешности аппроксимации используем представление решения дифференциальной задачи степенным рядом в узлах шаблона сетки :

$$u(x_{k\pm1}, t_n) = u(x_k, t_n) \pm h\frac{\partial u}{\partial x} + \frac{h^2}{2}\frac{\partial^2 u}{\partial x^2} \pm \frac{h^3}{6}\frac{\partial^3 u}{\partial x^3} + \frac{h^4}{24}\frac{\partial^4 u}{\partial x^4} + O(h^5), \tag{3.46}$$

$$u(x_k, t_{n\pm1}) = u(x_k, t_n) \pm \tau\frac{\partial u}{\partial t} + \frac{\tau^2}{2}\frac{\partial^2 u}{\partial t^2} \pm \frac{\tau^3}{6}\frac{\partial^3 u}{\partial t^3} + \frac{\tau^4}{24}\frac{\partial^4 u}{\partial t^4} + O(\tau^5), \tag{3.47}$$

где производные $\dfrac{\partial^m u}{\partial x^m}$ и $\dfrac{\partial^m u}{\partial t^m}$, $m = 1, 2, \ldots$, вычисляются в точке (x_k, t_n), которая является центральной точкой шаблона.

Подстановка решения дифференциальной задачи, представленного степенными рядами (3.46) и (3.47), в уравнение (3.26) приводит к следующему равенству:

$$\frac{\partial u}{\partial t} + v\frac{\partial u}{\partial x} + \psi(h,\tau) = \psi(h,\tau) = \frac{\tau}{2}\frac{\partial^2 u}{\partial t^2} - v\frac{h}{2}\frac{\partial^2 u}{\partial x^2} + O(h^2 + \tau^2). \tag{3.48}$$

Таким образом, для схемы с разностями против потока (3.26) погрешность аппроксимации имеет первый порядок относительно шагов сетки τ и h. Однако, можно показать, что любое решение уравнение переноса удовлетворяет следующему уравнению:

$$\frac{\partial^2 u}{\partial^2 t} - v^2\frac{\partial^2 u}{\partial x^2} = 0. \tag{3.49}$$

С учетом (3.49) имеем:

$$\psi(h,\tau) = h\frac{v}{2}(C-1)\frac{\partial^2 u}{\partial x^2} + O(h^2+\tau^2) = \tau\frac{v^2}{2}(1-C^{-1})\frac{\partial^2 u}{\partial x^2} + O(h^2+\tau^2), \qquad (3.50)$$

где $C = \dfrac{\tau v}{h}$ — **число Куранта** . Из последнего уравнения следует, что если $C = 1$ то главный член погрешности обращается в нуль. Более того, можно показать, что при $C = 1$ вся невязка разностной схемы обращается в ноль. Это объясняет аномально высокую точность схемы с разностями против потока в случае, когда число Куранта равно единице.

С другой стороны, анализ погрешности аппроксимации позволяет объяснить поведение разностного решения при $C \neq 1$. В частности, если $C < 1$ главный член погрешности аппроксимации равносилен добавлению в оригинальное уравнения дополнительного фактора диффузии, которая приводит к сглаживанию решения. Если $C > 1$, то вносимая погрешностью "отрицательная"диффузия вместо сглаживания порождает неустойчивость схемы.

В силу отмеченных выше обстоятельств, касающихся влияния погрешности аппроксимации на качественное поведение решения, становится понятен смысл последнего слагаемого в схеме Лакса – Вендроффа. Это слагаемое вносит искусственную диффузию, позволяющую стабилизировать явную неустойчивую схему (3.27) .

Для анализа устойчивости разностных схем в случае линейных дифференциальных задач с постоянными коэффициентами рассмотрим одну из наиболее наглядных методик, основанную на разделении переменных и методе Фурье. Проиллюстрируем особенности данной методики на примере схемы с разностями против потока для уравнения переноса. Не нарушая общности, в силу линейности задачи, нам достаточно рассмотреть вопрос устойчивости для отдельной Фурье компоненты разностного решения:

$$U_k^n = q^n \exp(i\omega_m hk), \qquad (3.51)$$

где $i = \sqrt{-1}$, $\omega_k = m\pi/L$, $m = 1, 2, \ldots K$. Условие устойчивости данной компоненты имеет вид

$$|q| \leqslant 1. \qquad (3.52)$$

Подстановка данной пробной моды (3.51) в разностную схему (3.26) приводит к следующему уравнению для величины q:

$$q^n \exp(i\omega_m hk)(q-1) + q^n \exp(i\omega_m hk)\frac{\tau v}{h}\left(1 - \exp(-i\omega_m h)\right). \qquad (3.53)$$

Решение уравнения (3.53) имеет вид:

$$q = 1 - \frac{\tau v}{h}\left(1 - \exp(-i\omega_m h)\right). \qquad (3.54)$$

Несложно заметить, что условие устойчивости (3.52) выполняется тогда и только тогда, когда

$$C = v\tau/h \leqslant 1. \qquad (3.55)$$

Последнее неравенство выражает известное **условия Куранта – Фридрихса – Леви**. Данное условие означает, что для стабильности явной схемы с разностями против потока необходимо и достаточно, чтобы размер шага по времени τ удовлетворял неравенству:

$$\tau \leqslant h/v. \tag{3.56}$$

В случае явных разностных схем наличие ограничений на размер шага τ вида (3.56) для устойчивости схемы является типичным. Разностные схемы, устойчивость которых имеет место при выполнении определенных ограничивающих условий на соотношение шагов сетки, называются **условно устойчивыми**. Для неявных разностных схем типична иная ситуация, когда стабильность имеет место без каких либо условий на соотношение шагов сетки. В некоторых частных случаях безусловная устойчивость может быть достигнута и в рамках явных разностных схем.

Устойчивость разностных схем ассоциируется с асимптотическим поведением их погрешности. Для устойчивой схемы погрешность решения, полученная на одном шаге, не испытывает неограниченного роста на последующих шагах по времени, когда число шагов стремится к бесконечности при $\tau \to \infty$.

Пример 3.4. Рассмотрим вопрос о согласованности и устойчивости схемы **Дюфорта – Франкела** для нестационарного уравнения теплопроводности:

$$\frac{U_k^{n+1} - U_k^{n-1}}{2\tau} - \frac{U_{k-1}^n - (U_k^{n-1} + U_k^{n+1}) + U_{k+1}^n}{h^2} = 0. \tag{3.57}$$

Подстановка пробной функции (3.51) в уравнение (3.57) приводит к следующему уравнению для величины q:

$$q^2 = \frac{1 + 2\tau h^{-2}\cos(\omega_k h)}{1 + 2\tau h^{-2}}. \tag{3.58}$$

Очевидно, что $|q| \leqslant 1$ для любых значений τ и h. Это означает безусловную устойчивость схемы Дюфорта – Франкела, несмотря на то, что она является схемой явного типа.

Подстановка точного решения, представленного разложением в степенной ряд (3.46), (3.47), в разностную схему (3.57) дает следующее выражение для погрешности аппроксимации:.

$$\psi(h,\tau) = \frac{\tau^2}{6}\frac{\partial^3 u}{\partial t^3} - \frac{\tau^2}{2h^2}\frac{\partial^2 u}{\partial t^2} - \frac{h^2}{12}\frac{\partial^4 u}{\partial x^4} + O\left(\frac{\tau^4}{2h^2} + h^4\right) = O\left(\frac{\tau^2}{2h^2} + h^2\right). \tag{3.59}$$

Как следует из выражения (3.59), погрешность аппроксимации схемы Дюфорта – Франкела стремится к нулю вместе с шагами сетки только при выполнении дополнительного ограничения на величину шага τ. В частности, легко видеть, что рассмотренная схема будет согласованной только при выполнении условия

$$\tau = O(h^\alpha), \quad \alpha > 1. \tag{3.60}$$

Для того, чтобы получить порядок аппроксимации $\psi(h,\tau) = O(\tau + h^2)$, мы должны использовать шаг по времени $\tau = O(h^2)$.

Рассмотренный пример показывает, что, несмотря на безусловную устойчивость схемы Дюфорта – Франкела, данная схема обладает **условной согласованностью**. Условная согласованность означает, что погрешность аппроксимации стремится к нулю при условиях, аналогичным условиям устойчивости разностных схем, в виде ограничений на соотношения шагов сетки. Свойство условной согласованности обычно присуще явным, безусловно устойчивым разностным схемам, примером которых является схема Дюфорта – Франкела. См. также результаты численных экспериментов, демонстрирующих проявления свойства условной согласованности, представленных в разделе 3.1.1.

Упражнение 3.3. 1. Исследовать устойчивость и согласоваанность схемы Лакса – Фридрихса. Сравните теоретические оценки и результаты численных экспериментов для скорости сходимости и условия устойчивости данной схемы

2. Исследовать устойчивость схемы с весами (3.10) е. для нестационарного уравнения теплопроводности. Определите минимальное значение Θ для которых данная схема является безусловно устойчивой.

3. Исследовать сходимость явной центрированной схемы для уравнения переноса. $q_m = |q(\omega_m)|$ при $h = 0.01$, $v = 1$, $\tau = h$, $\omega_m \in [0, \pi/h]$. Определите, какая из Фурье компонент (3.51) является наиболее неустойчивой.

4. Докажите, что функция $u = 1 - (x/L)^2$ является решение задачи (3.5), (3.7), при $t \to \infty$, и

$$f(x) = 2, \quad \frac{\partial u}{\partial x}\,|_{x=0} = 0, \quad u(L,t) = 0. \tag{3.61}$$

Решите данную задачу численно, внося соответствующие модификации в программную реализацию примера в разделе 3.1.1, принимая во внимание краевые условия Неймана на левой стороне интервала. Зависит ли точность решения задачи от точности аппроксимации краевого условия Неймана?

5. Оцените и сравните погрешности аппроксимации разностных схем (3.26) – (3.32) для уравнения переноса.

6. Исследуйте устойчивость и согласованность трехслойной разностной схемы (3.10) е. для нестационарного уравнения теплопроводности.

7. Оцените погрешность аппроксимации разностной схемы (3.10) с. для нестационарного уравнения теплопроводности (3.5) – (3.7). Докажите, что схема с весами при $\Theta = 1/2 - h^2/(12\tau)$ является безусловно устойчивой и имеет погрешность аппроксимации $O(h^4 + \tau^2)$.

3.1.4. Эллиптические уравнения в прямоугольной области

Многие актуальные приложения приводят к краевым задачам для **уравнений в частных производных эллиптического типа** , которые в трехмерной постановке имеют вид

$$\sum_{i,j=1}^{3} \frac{\partial}{\partial x_i} \lambda_{ij}(x) \frac{\partial u}{\partial x_j} = f(x), \quad x \in \Omega. \tag{3.62}$$

Рассмотрим пример эллиптического уравнения — **уравнение Пуассона**

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = f(x,y), \quad (x,y) \in \Omega = [0, L_x] \times [0, L_y], \tag{3.63}$$

с **краевыми условиями Дирихле**:

$$u(0,y) = u(L_x,y) = u(x,0) = u(x,L_y) = 0. \tag{3.64}$$

Используя двумерную прямоугольную сетку с шагами дикретизации $h_x = L_x/K$, $h_y = L_y/M$:

$$\Omega_h = \{(x_k, y_m),\ x_k = kh_x,\ \ k = 0,1,\ldots,K,\ y_m = mh_y,\ \ m = 0,1,\ldots,M\}, \tag{3.65}$$

дифференциальная задача (3.63), (3.64) может быть аппроксимирована разностной схемой на пятиточечном шаблоне, показанном на рис. 3.5 a. :

$$\frac{1}{h_k^2}\left(U_{k-1,m} - 2U_{k,m} + U_{k+1,m}\right) + \frac{1}{h_y^2}\left(U_{k,m-1} - 2U_{k,m} + U_{k,m+1}\right) = f(x_k, y_m). \tag{3.66}$$

Шаблон для аппроксимации **смешанных производных** представлен на рис. 3.5 b. Разностная аппроксимация смешанных производных, соответствующая данному шаблону имеет вид

$$\frac{\partial u^2}{\partial y \partial x} \simeq \frac{1}{4h_x h_y}\left(U_{k+1,m+1} - U_{k-1,m+1} - U_{k+1,m-1} + U_{k-1,m-1}\right). \tag{3.67}$$

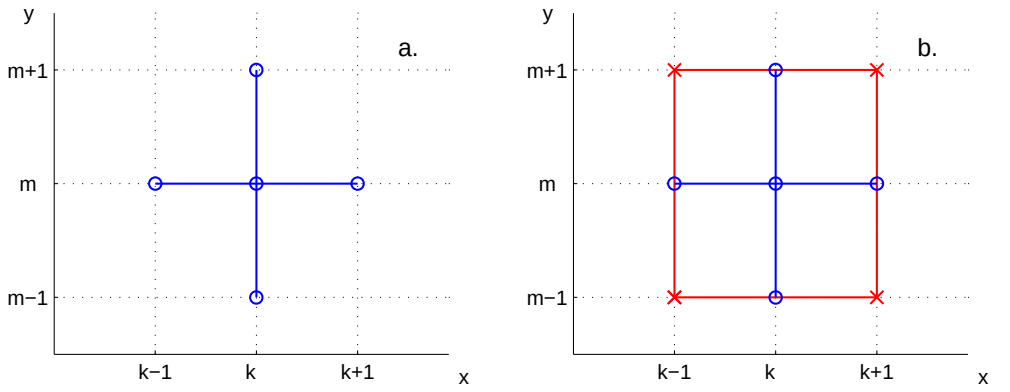


Рис. 3.5. Типовые шаблоны для аппроксимации двумерных эллиптических уравнений в частных производных. Шаблон для смешанных производных помечен x-образным маркером.

Несложно показать, что разностная схема (3.66) имеет второй порядок аппроксимации. В случае переменных коэффициентов дифференциальный оператор задачи (3.62) может быть аппроксимирован аналогично, как и для обыкновенных дифференциальных уравнений (см. уравнения (2.29)–(2.30)).

Разностная схема (3.65) представляет собой систему алгебраических уравнений с пятидиагональной матрицей:

$$AU = F. \tag{3.68}$$

При наличии в уравнении смешанных производных шаблон схемы семиточечный и разностная задача сводится к системе линейных алгебраических уравнений с семидиагональной матрицей. Смешанные производные в эллиптических задачах ассоциируются с моделированием анизотропных сред, а также, например, с нелинейными преобразованиями координат.

Структура матрицы A зависит от способа упорядочения компонент вектора U и F. Обычно используется **естественное упорядочение по строкам**:

$$U = (U_1, U_2, \ldots, U_K)^T, \tag{3.69}$$

$$U_k = (U_{k1}, U_{k2}, \ldots U_{kM})^T, \quad k = 1, 2, \ldots, K. \tag{3.70}$$

В этом случае матрица системы является **матрицей Пуассона**, которая может быть представлена в блочно-трехдиагональном виде:

$$A = \begin{bmatrix} D & E & & & & & \\ E & D & E & & & 0 & \\ & \ddots & \ddots & \ddots & & & \\ & & & & & & \\ & & 0 & & \ddots & \ddots & \ddots \\ & & & & & E & D & E \\ & & & & & & E & D \end{bmatrix}, \tag{3.71}$$

где $E = h_y^{-2} I$, I — единичная матрица размерности $K \times M$, D — трехдиагональная матрица, содержащая компоненты $d_{k,k} = -2h_x^{-2} - 2h_y^{-2}$, $d_{k,k\pm1} = h_x^{-2}$.

Матрица $-A$ является симметричной и положительно определенной. Как следствие, решение данной системы существует и единственно. Число обусловленности матрицы A зависит от параметров дискретизации h_x и h_y: $\mathrm{cond(A)} = \mathrm{O(h_x^{-2} + h_y^{-2})}$. Система (3.68) может быть решена с помощью прямых или итерационных методов. В случае постоянных коэффициентов матрица A является циркулянтной матрицей и система (3.68) может быть решена эффективно с использованием **быстрого преобразования Фурье**. Применимость метода Фурье к решению дискретной краевой задачи (3.68), (3.71) можно объяснить в терминах собственных векторов и собственных значений оператора второй разностной производной:

$$-\frac{1}{h_k^2}\left(\varphi_{i-1,j}^{k,m} - 2\varphi_{i,j}^{k,m} + \varphi_{i+1,j}^{k,m}\right) - \frac{1}{h_y^2}\left(\varphi_{i,j-1}^{k,m} - 2\varphi_{i,j}^{k,m} + \varphi_{i,j+1}^{k,m}\right) = \lambda_{k,m}\varphi_{i,j}^{k,m}, \tag{3.72}$$

$$\lambda_{k,m} = \frac{4}{h_x^2}\sin^2\left(\frac{\pi k h_x}{2L_x}\right) + \frac{4}{h_y^2}\sin^2\left(\frac{\pi m h_y}{2L_y}\right), \tag{3.73}$$

$$\varphi_{i,j}^{k,m} = \varphi^{k,m}(x_i, y_j) = \frac{2}{\sqrt{(L_x L_y)}}\sin\left(\frac{\pi k x_i}{L_x}\right)\sin\left(\frac{\pi m y_j}{L_y}\right), \tag{3.74}$$

где $k = 1, 2, \ldots, K$, $m = 1, 2, \ldots, M$,. Собственные векторы (3.74) образуют ортогональный базис в пространстве сеточных функций $U_{i,j} = U(x_i, y_j)$, удовлетворяющих краевым условиям (3.64). Вследствие этого, любое решение задачи (3.73),(3.74) может быть представлено в следующем виде:

$$U_{i,j} = \sum_{m=1}^{M} \sum_{k=1}^{K} \varphi_{i,j}^{k,m} c_{k,m}. \tag{3.75}$$

Здесь $c_{k,m}$ — коэффициенты Фурье, т.е. коэффициенты разложения приближенного решения в виде отрезка ряда Фурье по синусам.

Принимая во внимание ортогональность собственных векторов $\varphi^{k,m}$, коэффициенты $c_{k,m}$ в разложении (3.75) определяются следующим образом:

$$c_{k,m} = \frac{f_{k,m}}{\lambda_{k,m}}, \quad k = 1, 2, \ldots, K, \quad m = 1, 2, \ldots, M, \tag{3.76}$$

где $f_{k,m}$ — коэффициенты соответствующего разложения вектора правой части системы:

$$F_{i,j} = \sum_{m=1}^{M} \sum_{k=1}^{K} \varphi_{i,j}^{k,m} f_{k,m}. \tag{3.77}$$

Преобразование векторов $f_{i,j}$ в соответствующие коэффициенты Фурье $F_{i,j}$ (равно как и обратное преобразование) может быть эффективно выполнено с помощью алгоритма быстрого дискретного преобразования Фурье (быстрого синус преобразования Фурье).

Отметим, что описанный выше метод Фурье используется здесь исключительно для реализации разностного метода (как способ диагонализации матрицы Пуассона[1]). Вместе с тем, данный метод может быть весьма просто модифицирован с целью получения всех преимуществ **спектрального метода Фурье**. Для этого достаточно заменить собственные значения матрицы Пуассона на собственные значения соответствующего дифференциального оператора (значения собственных функций дифференциального оператора в узлах сетки совпадают со значениям собственных векторов дискретной задачи):

$$-\frac{\partial^2 \varphi}{\partial x^2} - \frac{\partial^2 \varphi}{\partial y^2} = \lambda \varphi, \quad \varphi(0, y) = \varphi(L_x, y) = \varphi(x, 0) = \varphi(x, L_y) = 0. \tag{3.78}$$

$$\lambda_{k,m} = \frac{\pi^2 k^2}{L_x^2} + \frac{\pi^2 m^2}{L_y^2}, \tag{3.79}$$

Отметим, что в силу нулевых краевых условий граничные точки не входят в компоненты вектора U. Для учета ненулевых краевых условий следует либо модифицировать соответствующим образом матрицу A и вектор F, дополнив их уравнениями для решения в граничных точках, либо учесть значения решения в граничных

[1] Любая симметричная матрица A подобна диагональной матрицы D и преобразование подобие $D = P^{-1}AP$ определяется матрицей P, столбцы которой образованы из собственных векторов матрицы A. Диагональные элементы матрицы D являются собственными значениями матрицы A. В нашем случае матрица P является матрицей дискретного Фурье преобразования. См. например, спектральную теорему для симметричных матриц [65].

точках посредством приграничных компонент вектора F. В рассмотренном ниже примере показано каким образом ненулевые условия Дирихле могут быть учтены с помощью вектора правой части без явного присутствия в векторе дискретного решения граничных точек задачи.

Пример 3.5. Рассмотрим задачи Дирихле для двумерного уравнения Пуассона

$$\begin{cases} \dfrac{\partial^2 u}{\partial x^2} + \dfrac{\partial^2 u}{\partial y^2} = 10\sin(2\pi x/L)\sin(2\pi y/L), \\ \\ -L/2 < x < L/2,\ -L/2 < y < L/2,\ L = 2. \end{cases} \tag{3.80}$$

с нулевыми и ненулевыми граничными условиями

$$u(\pm L/2, y) = u(x, \pm L/2) = 0. \tag{3.81}$$

$$u(\pm L/2, y) = y, \quad u(x, \pm L/2) = \pm L/2. \tag{3.82}$$

Программная реализация разностного метода для рассмотренных задач и результаты численного моделирования представлены ниже.

```
%%    Уравнение Пуассона  %%%%%%%%%%%%%%%%%%
%%      с краевыми условиями Дирихле  %%%%%%
L = 2;
n  =29;
%%%%   Построение сетки  %%%%%%%%%%%%%%%%%%%
%% граничные точки не входят в сетку %%%%%%%
h  = L /(n+1);
x =  -L/2+h :h :L /2-h ;
[Y,X] = ndgrid(x,x);
F =  10*sin(2*pi*X/L).*sin(2*pi*Y/L);
A=-gallery('poisson',n )/h^2 % Poisson matrix
%%  Собственные значения матрицы Пуассона  %%
g = - 4/h^2*(sin(pi*(1:n)*h/(2*L))).^2
[g_x,g_y] = ndgrid(g,g);
Lambda=g_x+g_y;
%%  Нулевые краевые условия  %%%%%%%%%%%%%%%
%% Метод конечных разностей %%%%%%%%%%%%%%%%
U0 = A\F(:);
U0 = reshape(U0,n,n);
%%      Фурье метод реализации PC    %%%%%%%%

F0 = reshape(F,n,n);
%% Дискретное sin-Фурье преобразования  dst %
%  применяется дважды (к столбцам и строкам)%
f = dst(F0);            % по столбцам
f = dst(f.').';         % по строкам
UU=f./Lambda
%% обратное  sin-Фурье преобразования  idst %
UU = idst(UU);          % по столбцам
```

```
UU = idst(UU.').';      % по строкам
%%  ненулевые граничные условия    %%%%%%%%%
b1 = -L/2;
F(1,:) =F(1,:) - b1/h^2;          % при x=-L/2
b2 = x;
F(:,1) = F(:,1) - b2(:)/h^2;      % при y=-L/2
b3 =  L/2;
F(end,:) = F(end,:) - b3/h^2;     % при x= L/2
b4 = x;
F(:,end) =F(:,end) - b4(:)/h^2;   % при y= L/2

%%    Метод конечных разностей %%%%%%%%%%%%%%
U = A\F(:);
U = reshape(U,n,n);
%%      Фурье метод реализации PC    %%%%%%%%
F0 = reshape(F,n,n);
f = dst(F0);
f = dst(f.').';
Uf=f./Lambda
Uf = idst(Uf);
Uf = idst(Uf.').';
figure('Position',[403 246 760 300])
subplot(1,2,1), contour(X,Y,U0,17);
subplot(1,2,1), mesh(X,Y,U0);
xlabel('x'); ylabel('y');
title('Нулевые условия Дирихле')
subplot(1,2,2), contour(X,Y,U,17);
subplot(1,2,2), mesh(X,Y,U);
title('Не нулевые условия Дирихле')
xlabel('x'); ylabel('y');
```

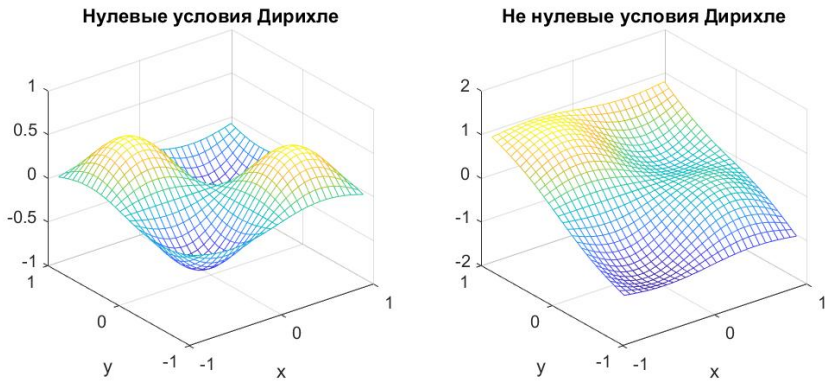


Рис. 3.6. Решение двумерного уравнения Пуассона с нулевыми и ненулевыми условиями Дирихле (3.81), (3.82)

Упражнение 3.4. 1. Используя функцию MATLAB **spy**, изобразите структуру разреженности матрицы Пуассона размерности 25x25, которая может быть сформирована с помошью функции **gallery** :

```
>>A=gallery('posson',5);
```

2. Используя функцию MATLAB **lu**, вычислите компоненты LU – факторизации матрицы Пуассона, $A = LU$. Сравните структуру разреженности исходной матрицы и ее L и U компонент. Оцените число не нулевых элементов матриц L и U ($LU = A$), как функцию размерности матрицы.

3. Оцените погрешность аппроксимации разностной схемы (3.66) на решении краевой задачи (3.63), (3.64).

4. Оцените погрешность аппроксимации смешанных производных (3.67).

5. Постройте разностную схему для трехмерного уравнения Пуассона в прямоугольной области с краевыми условиями Дирихле.

6. Проводя соответствующую модификацию программной реализации разностного метода для уравнения Пуассона в рассмотренном выше примере, вычислить и визуализировать в виде функции двух переменных разность между решениями, полученными с помощью разностного метода и разностного метода с реализацией на основе Фурье преобразования (различия должны быть в пределах вычислительной погрешности).

7. Внося необходимые модификации в программную реализацию рассмотренного выше примера, расширьте функциональные возможности алгоритма решения задачи (3.63), (3.64) для случая $L_x \neq L_y$, $K \neq M$.

3.1.5. Нелинейные уравнения

Область применимости линейных моделей всегда ограничена пределами входных данных, в рамках которых мы можем пренебречь зависимостью коэффициентов модели от решения и его производных. В силу этого, при моделировании прикладных инженерно – физических задач, адекватное описание реальности, как правило, достигается при использовании нелинейных уравнений в частных производных.

Нелинейные дифференциальные задачи весьма нетривиальны с точки зрения классического анализа, и численные методы во многих случаях выступают в качестве единственного универсального подхода, позволяющего получить по крайне мере приближенное решение. Более того, и с точки зрения численного анализа, нелинейные дифференциальные задачи во многих случаях также представляют немалую сложность.

Прежде всего, в отличие от линейных неявных разностных схем, дискретизация нелинейные уравнения в частных производных приводит к системам нелинейных уравнений, анализ которых сам по себе представляет серьезную проблему.

Кроме того, при моделирование нелинейной динамики приходится иметь дело с описанием достаточно сложных режимов, таких как генерация ударных волн, диффузионный хаос, режимы с обострением и т.п. Как правило, большинство таких сложных нелинейных режимов характеризуется деградацией гладкости и уширением Фурье спектра решения, что неизбежно порождает проблемы с обеспечением согласованности и сходимости дискретных моделей.

И, наконец, принципиальным моментом теории численных методов для нелинейных дифференциальных задач является то, что фундаментальное положение теоремы Лакса об эквивалентности согласованности и устойчивости с одной стороны и сходимости дискретной модели с другой стороны, не выполняется для случая нелинейных задач. Поэтому, если согласованность и устойчивость являются достаточными условиями сходимости в линейном случае, то для нелинейных задач это лишь необходимое условие сходимости.

Тем не менее, несмотря на отмеченные выше сложности, ситуация с численным анализом нелинейных задач представляется не столь драматичной, как это может показаться на первый взгляд.

Типичным примером нелинейного уравнения в частных производных является **нелинейное уравнение переноса**, в котором скорость зависит от решения (сравните с линейным случаем (3.15)):

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} = 0, \quad (x,t) \in \Omega = [0,L] \times [0,T], \tag{3.83}$$

Начальные и граничные условия имеют вид:

$$u(x, t=0) = u_0(x) \geqslant 0, \quad u(0,t) = u_b(t) \geqslant 0. \tag{3.84}$$

Уравнение (3.83) может быть представлено в консервативной (дивергентной) форме:

$$\frac{\partial u}{\partial t} + \frac{\partial F}{\partial x} = 0, \tag{3.85}$$

где $F = u^2/2$ — поток величины u.

Интегрирование (3.85) по прямоугольной области $\Omega = [0,L] \times [0,T]$ дает следующее тождество

$$\int\limits_0^L (u(x,T) - u(x,0))\, dx + \frac{1}{2}\int\limits_0^T \left(u^2(L,t) - u^2(0,t)\right) dt = 0. \tag{3.86}$$

Полученное равенство (3.86) представляет собой интегральный закон сохранения величины u в пределах контрольного объема Ω. Очевидно, что данный **закон сохранения** выполняется для произвольного **контрольного объема**:

$$V = [x, x+\Delta x] \times [t, t+\Delta t] \subset [0,L] \times [0,T].$$

Законы сохранения играют важную роль при построении численных методов как для линейных, так и нелинейнных дифференциальных задач. В частности, важным представляется обеспечить **консервативность** нелинейных разностных схем, что подразумевает выполнение соответствующих дискретных аналогов законов сохранения, имеющих место в дифференциальной задаче. Например, дискретный аналог закона сохранения (3.86), может быть получен путем формальной замены интегралов на соотвествующие квадратуры, используя простейшую формулу прямоугольников:

$$\sum_{k=1}^{K} h\left(U_k^N - U_k^0\right) + \frac{1}{2}\sum_{n=1}^{N-1} \tau\left((U_K^n)^2 - (U_0^n)^2\right) = 0. \tag{3.87}$$

Здесь $U_k^n = U(x_k, t_n)$, $(x_k, t_n) \in \Omega_{h\tau}$,

$$\Omega_{h\tau} = \{(x_k, t_n) : \ x_k = x_{k-1} + h_k, \ k = 1, 2, \ldots, K; \ t_n = t_{n-1} + \tau_n, \ n = 0, 1, \ldots, N \}, \tag{3.88}$$

В некоторых случаях отсутствие консервативности может приводить к некорректному поведению искомого решения, ложной сходимости. Такие ситуации, как правило, возникают при отсутствии достаточной гладкости решения. В качестве иллюстрации данного факта рассмотрим следующий пример.

Пример 3.6. Точно также, как и в линейном случае (3.15), (3.21), существует кусочно постоянное решение вида

$$u(x, t) = u_0(x - vt), \tag{3.89}$$

удовлетворяющее уравнениям (3.85) и (3.86) с начальными условиями, имеющими разрыв при $x = x_0$

$$u_0(x) = \begin{cases} a, & x \leqslant x_0, \\ b, & x > x_0, \end{cases} \tag{3.90}$$

Здесь мы полагаем $a > b \geqslant 0$. Можно показать, что скорость ударной волны (3.21) зависит от амплитуды решения:

$$v = (a - b)/2. \tag{3.91}$$

Рассмотрим явную схему с разностями против потока (3.26) для решения уравнения (3.15) с начальными условиями (3.90):

$$\frac{U_k^{n+1} - U_k^n}{\tau} + U_k^n \frac{U_k^n - U_{k-1}^n}{h} = 0, \tag{3.92}$$

Полагаем $U_{k-1}^n = a$, $U_k^n = b$ и $\tau = h/b$. Тогда из уравнения (3.92) мы имеем

$$U_k^{n+1} = b - b\frac{h}{b}\frac{b - a}{h} = a. \tag{3.93}$$

Последнее равенство означает, что ударная волна распространяется со скоростью $v = h/\tau = b$ вместо $v = (a - b)/2$, как это следует из закона сохранения (3.86). Можно показать, что разностная схема (3.92) сходится при $\tau = h/b \to 0$ к ложному разрывному решению, для которого скорость ударной волны не совпадает с истинным значением. Можно также показать, что закон сохранения (3.87) не выполняется для рассмотренной схемы. В самом деле, умножая (3.92) на $h\tau$ и суммируя по сетке $\Omega_{h\tau}$, приходим к следующему равенству

$$\sum_{n=1}^{N-1} \sum_{k=1}^{K} h \left(U_k^{n+1} - U_k^n\right) + \frac{1}{2} \sum_{n=0}^{N-1} \sum_{k=1}^{K} \tau U_k^n \left(U_k^n - U_{k-1}^n\right) = 0, \tag{3.94}$$

и, как следствие,

$$\sum_{k=1}^{K} h \left(U_k^N - U_k^0\right) + \frac{1}{2} \sum_{n=0}^{N-1} \tau \left((U_K^n)^2 - (U_0^n)^2\right) = -\frac{1}{2} \sum_{n=0}^{N-1} \sum_{k=1}^{K} \tau \left(U_k^n - U_{k-1}^n\right)^2. \tag{3.95}$$

В отличие от (3.87), правая часть уравнения (3.95) характеризует величину дисбаланса закона сохранения (3.86).

С другой стороны, можно показать, что консервативность может быть обеспечена при использовании, например, следующей разностной схемы:

$$\frac{U_k^{n+1} - U_k^n}{\tau} + \frac{1}{2}\left(U_k^n + U_{k-1}^n\right)\frac{U_k^n - U_{k-1}^n}{h} = 0. \tag{3.96}$$

Программная реализация рассмотренного примера и результаты численных экспериментов, с использованием консервативных и не консервативных схем представлены ниже.

```
%% методы конечных разностей %%%%%%%%%%%%%%%
%%  для нелинейного уравнения переноса   %%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
L = 2;  T = 1.0;
a =0.1; b = 1.; x_0 = 0.5;        %для начального условия
h = 0.05;    tau = h/b;           %шаг сетки
x =  0:h:L;  N = length(x);       %сетка
%%%%  начальные условия %%%%%%%%%%%%%%%%%%%%%
u_0 = a*ones(size(x));
m = find(x < x_0);
u_0(m) = u_0(m)+(b-a);
u_10 = u_0;
u_20 = u_0;
for t=tau:tau:T
%%   Консервативная схема %%%%%%%%%%%%%%%%%%
for k=2:N
u_1(k)= u_10(k)-0.5*tau/h*(u_10(k)+u_10(k-1))*(u_10(k)-u_10(k-1));
end
u_1(1) = b;
u_10 = u_1;
%%%%     неконсервативная схема %%%%%%%%%%%%%
for k=2:N
u_2(k)= u_20(k)-tau/h*u_20(k)*(u_20(k)-u_20(k-1));
end
u_2(1) = b;
u_20 = u_2;
end
plot(x,u_0,'k-.',x,u_1,'bp-',x,u_2,'ro-','LineWidth',1);
xlabel('x'); ylabel('U(x)'); axis([0,2,0,1.5]);
legend('Нач. усл.', 'Косервативная схема', 'Не консервативная');
title('Распространение ударной волны'); grid
```

Результаты численных экспериментов, представленные на рис. 3.7, показывают, что решение в виде ударной волны в рассмотренной задаче, полученное при использовании неконсервативной схемы, запаздывает по отношению к решению консерва-

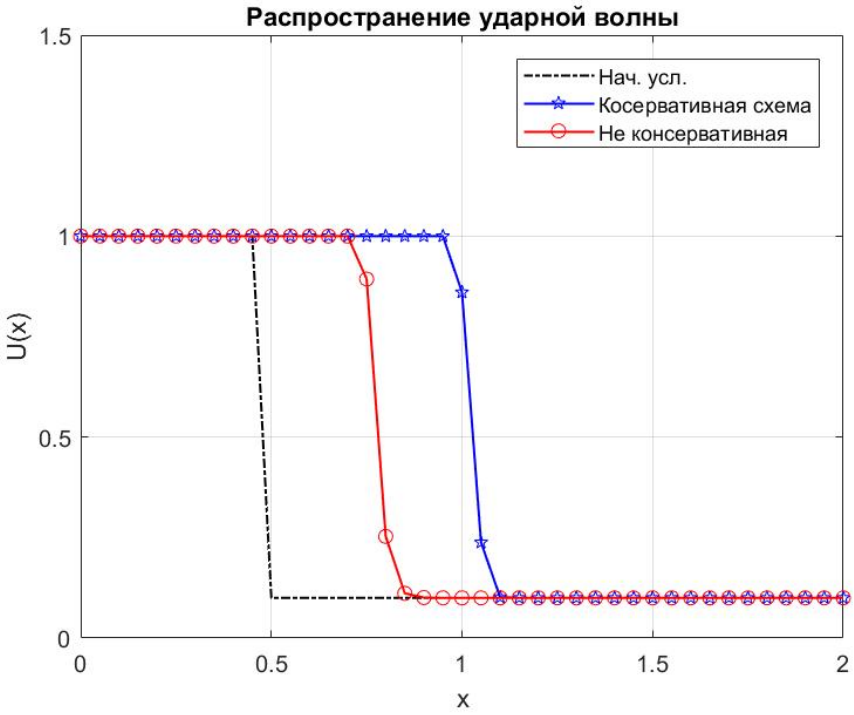


Рис. 3.7. Решение задачи (3.84), (3.84), (3.90) при $a = 0.1$, $b = 1$, $x_0 = 0.5$, $T = 1$, полученная при использовании консервативной схемы (3.96) и неконсервативной схемы (3.92)

тивной схемы. Заметим также, что скорость ударной волны, для решения консервативной схемы совпадает со значением скорости, вытекающейм из аналитической оценки (3.91).

Рассмотренный пример показывает, что неконсервативная схема демонстрирует ложную сходимость, в то время как консервативный метод предотвращает неадекватное поведение решения даже в случае, когда фактическая гладкость решения не обеспечивает согласованность разностной схемы. Если повторить расчеты для случая гладких начальных данных, то консервативная схема будет также показывать лучшие результаты, по сравнению с неконсервативной, однако различия в результатах моделирования будут не столь заметными.

Если для дифференциальной задачи имеют место несколько инвариантов, то разумно попытаться построить **полностью консервативную схему**, для которой выполняется максимальное число дискретных аналогов интегральных инвариантов исходной дифференциальной задачи.

Наряду с консервативной схемой (3.96), для нелинейной задачи (3.83) – (3.84) может быть использована следующая неявная разностная схема:

$$\frac{U_k^{n+1}-U_k^n}{\tau}+\frac{1}{2}\left(U_k^{n+1}+U_{k-1}^{n+1}\right)\frac{U_k^{n+1}-U_{k-1}^{n+1}}{h}=0, \tag{3.97}$$

для которой выполняется следующий инвариант:

$$\sum_{k=1}^{K} h\left(U_k^N-U_k^0\right)+\frac{1}{2}\sum_{n=1}^{N}\tau\left((U_K^n)^2-(U_0^n)^2\right)=0. \tag{3.98}$$

Неявная схема (3.97) существенно нелинейная, однако в рассмотренном случае ее решение может быть выражено явной формулой:

$$U_k^{n+1}=-\frac{h}{\tau}+\sqrt{\frac{h^2}{\tau^2}+\frac{2h}{\tau}U_k^n+(U_{k-1}^{n+1})^2}. \tag{3.99}$$

В отличие от явной схемы (3.96), неявный метод (3.97) является безусловно устойчивым и монотонным. **Монотонность разностного метода** – это свойство, отвечающее за подавление паразитических осцилляций, которые часто можно наблюдать при моделировании разрывных решений нелинейных задач. Условие монотонности явных схем определяется возможностью представления решения в виде:

$$u_k^{n+1}=\sum_m \beta_m u_{k+m}^n, \quad \beta_m \geqslant 0. \tag{3.100}$$

В общем случае нелинейного уравнения переноса, представленного в дивергентной форме (3.85), явная разностная схема может быть представлена в следующем виде:

$$\frac{U_k^{n+1}-U_k^n}{\tau}+V^-(U_{k-1}^n,U_k^n)\frac{U_k^n-U_{k-1}^n}{h}+V^+(U_{k+1}^n,U_k^n)\frac{U_{k+1}^n-U_k^n}{h}=0, \tag{3.101}$$

$$V^-(U_{k-1}^n,U_k^n)=\frac{1}{2}\left[\frac{F(U_k^n)-F(U_{k-1}^n)}{U_k^n-U_{k-1}^n}+\left|\frac{F(U_k^n)-F(U_{k-1}^n)}{U_k^n-U_{k-1}^n}\right|\right]. \tag{3.102}$$

$$V^+(U_{k+1}^n,U_k^n)=\frac{1}{2}\left[\frac{F(U_{k+1}^n)-F(U_k^n)}{U_{k+1}^n-U_k^n}-\left|\frac{F(U_{k+1}^n)-F(U_k^n)}{U_{k+1}^n-U_k^n}\right|\right], \tag{3.103}$$

Полностью неявная схема будет соответственно иметь следующий вид:

$$\frac{U_k^{n+1}-U_k^n}{\tau}+V^-(U_{k-1}^{n+1},U_k^{n+1})\frac{U_k^{n+1}-U_{k-1}^{n+1}}{h}+V^+(U_{k+1}^{n+1},U_k^{n+1})\frac{U_{k+1}^{n+1}-U_k^{n+1}}{h}=0, \tag{3.104}$$

Решение нелинейной разностной схемы (3.104) может быть получено с помощью итерационного метода, например,

$$\frac{\overset{s}{U}{}_k^{n+1}-U_k^n}{\tau}+V^-(U_{k-1}^{n+1},\overset{s-1}{U}{}_k^{n+1})\frac{\overset{s}{U}{}_k^{n+1}-\overset{s}{U}{}_{k-1}^{n+1}}{h}+V^+(U_{k+1}^{n+1},\overset{s-1}{U}{}_k^{n+1})\frac{\overset{s}{U}{}_{k+1}^{n+1}-\overset{s}{U}{}_k^{n+1}}{h}=0, \tag{3.105}$$

где $s = 1, 2, \ldots$, $\overset{0}{U}{}_k^{n+1} = U_k^n$, $k = 2, 3, \ldots, K-1$, U_1^{n+1} или (и) U_K^{n+1} определяются краевыми условиями. Заметим, что краевые условия для уравнения (3.85) задаются на левой границе, $x = 0$, если $\left.\dfrac{\partial F(u)}{\partial x}\right|_{x=0} \geqslant 0$, и (или) на правой границе, $x = L$, если $\left.\dfrac{\partial F(u)}{\partial x}\right|_{x=L} \leqslant 0$.

Для нелинейности степенного вида, $F(u) = cu^m$, уравнения (3.102), (3.103) допускают упрощения:

$$\frac{c(U_{k+1}^n)^m - c(U_k^n)^m}{U_{k+1}^n - U_k^n} =$$

$$= c\left[(U_{k+1}^n)^{m-1} + (U_{k+1}^n)^{m-2}U_k^n + \cdots + U_{k+1}^n(U_k^n)^{m-2} + (U_k^n)^{m-1}\right]. \qquad (3.106)$$

Наиболее популярные численные методы для нелинейных уравнений переноса основаны на использовании явных разностных схем. Примером такой схемы, привлекающей в последнее время большой интерес, является так называемая схема "кабаре"(3.32), которая допускает представление в трехстадийной форме по принципу предиктор – корректор[2]:

$$\text{предиктор} \qquad \frac{U_{k+1/2}^{n+1/2} - U_{k+1/2}^n}{0.5\tau} = \frac{W(U_{k+1}^n) - W(U_k^n)}{h}, \qquad (3.107)$$

$$\text{экстраполяция} \qquad U_k^{n+1} = \begin{cases} 2U_{k-1/2}^{n+1/2} - U_{k-1}^n, & \partial W/\partial U \geqslant 0, \\ 2U_{k+1/2}^{n+1/2} - U_{k+1}^n, & \partial W/\partial U < 0, \end{cases} \qquad (3.108)$$

$$\text{корректор} \qquad \frac{U_{k+1/2}^{n+1} - U_{k+1/2}^{n+1/2}}{0.5\tau} = \frac{W(U_{k+1}^{n+1}) - W(U_k^{n+1})}{h}. \qquad (3.109)$$

На стадии экстраполяции может быть добавлена процедура коррекция решения для получения монотонности схемы. Для этих целей, после выполнения (3.108), полученное значение U_k^{n+1} приводится в границы интервала:

$$\begin{aligned} &\min\left(U_k^n, U_{k-1/2}^n, U_{k-1}^n\right) \leqslant U_k^{n+1} \leqslant \max\left(U_k^n, U_{k-1/2}^n, U_{k-1}^n\right) \quad \partial W/\partial U \geqslant 0, \\ &\min\left(U_k^n, U_{k+1/2}^n, U_{k+1}^n\right) \leqslant U_k^{n+1} \leqslant \max\left(U_k^n, U_{k+1/2}^n, U_{k+1}^n\right), \quad \partial W/\partial U < 0, \end{aligned} \qquad (3.110)$$

По сравнению с другими методами, схема кабаре демонстрирует улучшенные **диссипативные** и **дисперсионные**) свойства, которые определяются главными членами невязки, в частности, соотношением вкладов производных четного и нечетного порядков соответственно.

[2]Более подробно од этой схеме см., например, Karabasov, S. A., Berloff, P. S., and Goloviznin, V. M. (2009). CABARET in the ocean gyres. Ocean Modelling, 30(2), P. 155-168., и другие работы авторов

Аналогично уравнению переноса, методика построения консервативных разностных схем может быть использована для разработки эффективных методов численного анализа **нелинейных уравнений теплопроводности**, с коэффициентом теплопроводности, зависящем от температуры:

$$\frac{\partial u}{\partial t} - \frac{\partial}{\partial x}\left(\lambda(u)\frac{\partial u}{\partial x}\right) = 0. \tag{3.111}$$

Консервативная форма данного уравнения теплопроводности по структуре подобна нелинейному уравнению переноса:

$$\frac{\partial u}{\partial t} + \frac{\partial W}{\partial x} = 0, \quad W = W(u) = -\lambda(u)\frac{\partial u}{\partial x}, \tag{3.112}$$

где W тепловой поток. Интегрируя второе из уравнений (3.112) имеем

$$-\int\limits_{x_k}^{x_{k+1}} \lambda^{-1}(u)W(u)dx = u(x_{k+1},t) - u(x_k,t) \simeq -W(u_{k+1/2}) \int\limits_{x_k}^{x_{k+1}} \lambda^{-1}(u)dx, \tag{3.113}$$

$$-\int\limits_{x_{k-1}}^{x_k} \lambda^{-1}(u)W(u)dx = u(x_k,t) - u(x_{k-1},t) \simeq -W(u_{k-1/2}) \int\limits_{x_{k-1}}^{x_k} \lambda^{-1}(u)dx. \tag{3.114}$$

Приближенное вычисление интегралов в правой части уравнений (3.113) и (3.114), используя формулы трапеций или прямоугольников, приводит к следующим выражениям для теплового потока $W_{k\pm1/2}$:

$$W_{k\pm1/2} \simeq \mp\frac{2\lambda_k\lambda_{k\pm1}}{h(\lambda_k + \lambda_{k\pm1})}\left(u(x_{k\pm1},t) - u(x_k,t)\right), \tag{3.115}$$

$$W_{k\pm1/2} \simeq \mp\frac{\lambda_k + \lambda_{k\pm1}}{2h}\left(u(x_{k\pm1},t) - u(x_k,t)\right). \tag{3.116}$$

Полностью неявная **консервативная схема** для нелинейного уравнения теплопроводности имеет следующий вид:

$$\frac{U_k^{n+1} - U_k^n}{\tau} = \frac{W_{k+1/2}^{n+1} - W_{k-1/2}^{n+1}}{h}, \tag{3.117}$$

где $W_{k\pm1/2}^{n+1}$ определяется одним из выражений (3.115) или (3.116). Разностные схемы (3.117), (3.115) и (3.117), (3.116) имеет порядок аппроксимации $O(h^2+\tau)$. Для получения более высокой точности порядка $O(h^2+\tau^2)$, можно использовать схему Кранка – Николсон:

$$\frac{U_k^{n+1} - U_k^n}{\tau} = \frac{1}{2}\left(\frac{W_{k+1/2}^{n+1} - W_{k-1/2}^{n+1}}{h} + \frac{W_{k+1/2}^{n} - W_{k-1/2}^{n}}{h}\right). \tag{3.118}$$

Пример 3.7. Различия в особенностях поведения решений линейной и нелинейной задач для уравнения теплопроводности проиллюстрируем на примере моделирования автомодельных решений. Для этого рассмотрим линейное однородное уравнение теплопроводности

$$\frac{\partial u}{\partial t} - \frac{\partial^2 u}{\partial x^2} = 0, \quad (x,t) \in \Omega = [-L, L] \times [0, T]. \tag{3.119}$$

с начальными условиями в виде δ-функции

$$u(x,t)|_{t=0} = \delta(x). \tag{3.120}$$

Задача (3.119), (3.120) на достаточно большом пространственном отрезке имеет следующее **автомодельное решение**,

$$u(x,t) = \frac{1}{\sqrt{4\pi t}} \exp\left(-\frac{x^2}{4t}\right), \tag{3.121}$$

которое является точным фундаментальным решением при $L \to \infty$.

Аналогичная задача для нелинейного уравнения теплопроводности

$$\frac{\partial u}{\partial t} - \frac{\partial}{\partial x}\left(u \frac{\partial u}{\partial x}\right) = 0, \tag{3.122}$$

имеет следующее автомодельное решение:

$$u(x,t) = t^{-1/3} f\left(x t^{-1/3}\right), \quad f(\psi) = \begin{cases} C - \psi^2/6C, & \psi^2 \leqslant 6C, \\ 0, & \psi^2 > 6C. \end{cases} \tag{3.123}$$

В обоих случаях для решения линейной и нелинейной задачи имеет место следующий инвариант:

$$I = \int\limits_{-\infty}^{\infty} u(x,t)\, dx = \text{const}. \tag{3.124}$$

Для численного решения рассмотренных задач мы будем использовать неявную разностную схему (3.116), (3.117), для реализации которой в нелинейном случае используем следующий итерационный метод:

$$\frac{\overset{s}{U}{}_k^{n+1} - U_k^n}{\tau} = \frac{1}{h^2}\left[\overset{s-1}{\lambda}{}_{k+1/2}\left(\overset{s}{U}{}_{k+1}^{n+1} - \overset{s}{U}{}_k^{n+1}\right) - \overset{s-1}{\lambda}{}_{k-1/2}\left(\overset{s}{U}{}_{k+1}^{n+1} - \overset{s}{U}{}_k^{n+1}\right)\right]. \tag{3.125}$$

Здесь $\overset{s-1}{\lambda}{}_{k\pm1/2} = \frac{1}{2}\left(\overset{s-1}{U}{}_{k\pm1}^{n+1} + \overset{s-1}{U}{}_k^{n+1}\right)$, $\overset{0}{U}{}_k^{n+1} = U_k^n$, $s = 1, 2, \ldots$. Для упрощения алгоритма мы ограничимся фиксированным числом итераций $s \leqslant 4$. Каждая итерация (3.125) требует решения системы линейных алгебраических уравнений с трехдиагональной матрицей.

Программная реализация рассмотренного примера и результаты численных экспериментов представлены ниже.

```
%% Автомоденлные решения уравнения теплопроводности %
%%        линейный и нелинейный случаи              %
N = 401; L = 4;
h = 2*L/(N-1); tau = 0.001;
x = -L:h:L;
x0 = x(2:end-1);
Nx = length(x0);
%% точные реш. линейной & нелинейной (UL&UN) задач %%
UL = @(x,t) exp(-x.^2/(4*t))/sqrt(4*pi*t);
C = (6)^(1/3)/4;
UN = @(x,t)0.5*t^(-1/3)*((C-(x/t^(1/3)).^2/6) + ...
+abs((C-(x/t^(1/3)).^2/6)));
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%             Initial condition                %%%%%
U0 = zeros(1,Nx);
U0(round(Nx/2)) = 1/h;     %дискретная дельта-функция
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%            Решение линейной задачи       %%%%%%%%%
U = U0(:);
e = ones(Nx,1);
A = eye(Nx)-tau/h^2*spdiags([e,-2*e,e],-1:1,Nx,Nx);
TL = [0.02; 0.1; 0.5]; %% контр. время визуализации
for t = 0:tau:TL(end)
U = A\U;
if (t == TL(1) || t == TL(2) || t == TL(3))
subplot(2,1,1),
hh=plot(x0,U,x(1:5:N),UL(x(1:5:N),t),'o:')
hh(1).LineWidth=1;  hh(2).LineWidth=1;
hh(2).MarkerSize = 3;
hh(1).Color = 'blu'; hh(2).Color = 'blu';
hold on
end
end
hold off
%%%================================================%%%%
title(['Линейное уравнение теплопроводности: t=' mat2str(TL)])
legend('Приближенное','Точное'); axis([-2 2 0 2.2]); grid
% subplot(2,1,1),
% plot(x0,U,x(1:10:end),UL(x(1:10:end),t),'o')
% legend('Numerical','Exact')

%%            Решение нелинейной задачи        %%%%%%
V = U0(:);
V0 = V;
TN = [0.01;0.1; 1.0];    %% контр. время визуализации
for t = 0:tau:TN(end)
```

```
%%   Итерации для решения нелинейной системы      %%%
for k = 1:4
a = [V(1:end-1)+V(2:end);0]/2;
b = [0;V(1:end-1)+V(2:end)]/2;
c = -a -b - h^2/tau;
A = -spdiags([a(:),c(:), b(:)],-1:1,Nx,Nx)*tau/h^2;
V = A\V0;
end
%% ============================================%%%%
if (t == TN(1) ||t == TN(2) || t == TN(3))
subplot(2,1,2),
hh=plot(x0,V,x(1:5:N),UN(x(1:5:N),t),'o:')
hh(1).LineWidth=1;  hh(2).LineWidth=1;
hh(2).MarkerSize = 3;
hh(1).Color = 'blu'; hh(2).Color = 'blu';
hold on
end
V0=V;
end
title(['Нелинейное уравнение теплопроводности:  t=' mat2str(TN)])
legend('Приближенное','Точное'); axis([-2 2 0 2.2]); grid
```

Представленные на рис. 3.8 результаты показывают, что приближенные решения практически не отличимы от точных значений. Решение нелинейной задачи имеет характерную пространственно локализованную структуру с разрывной первой производной на границе движущего фронта тепловой волны. Отметим, что консервативная схема обеспечивает сходимость, несмотря на то, что гладкость решения не достаточна для согласованности дискретной задачи. Вместе с тем, несложно убедиться с помощью численного эксперимента, что разностная схема (3.117) с аппроксимацией коэффициентов в виде (3.115) сохраняет консервативность, но ведет к абсолютно некорректному решению. Таким образом, консервативность не является в строгом смысле достаточным условием для сходимости, но во многих случаях негладких решений данное свойство имеет важное значение для получения адекватных результатов.

Упражнение 3.5. 1. Внося необходимые изменения в программную реализацию методов решения нелинейных уравнений переноса, сравните адекватность результатов численных экспериментов, полученных с использованием явной (3.101) – (3.103) и неявной (3.104) схем при различных значениях шага сетки: $\tau = h$, $\tau = h/1.1$, $\tau = 1.1 \cdot h$. Какая из рассмотренных схем представляется более предпочтительной для моделирования распространения ударных волн?

2. Замените разрывные начальные условия в примере решения нелинейного уравнения переноса на Гауссову функцию с центром в точке $x = 0.5$, $u_0(x) = \exp\left(-25(x-0.5)^2\right)$. Что происходит с решением задачи с этими гладкими начальными условиями при $t \geqslant 0.2$?

3. Докажите, что условия монотонности (3.100) для явной схемы (3.26) совпадает с условием устойчивости (3.55).

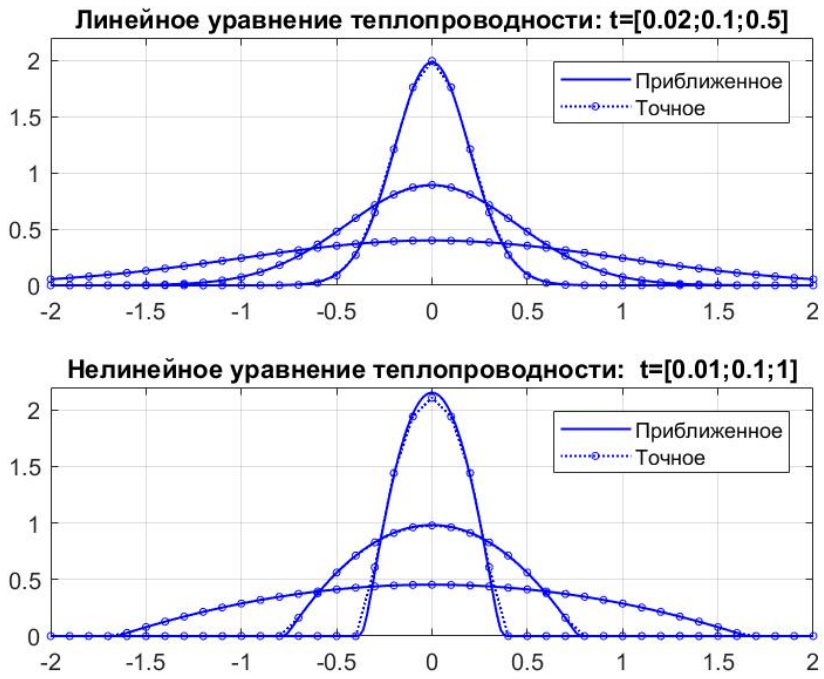


Рис. 3.8. Решение линейного и нелинейного уравнений теплопроводности (3.119), (3.122), с начальными условиями в виде δ-функции (3.120).

4. Покажите, что схема Лакса – Фридрихса (3.29) не является монотонной (см. также результаты численных экспериментов для данной схемы представленные на рис. 3.4).

5. Убедитесь, что для решения консервативной разностной схемы (3.96) выполняется следующий инвариант:

$$I = \sum_{k=1}^{N-1} hU_k^n = \text{const}. \tag{3.126}$$

Сравните (3.126) и (3.124).

6. Сравните результаты численных экспериментов для нелинейного уравнения теплопроводности, (3.122) полученные при использовании консервативной разностной схемы (3.117), (3.116), и схемы (3.117), когда поток $W_{k\pm1/2}$ вычисляется согласно следующему выражению:

$$W_{k\pm1/2}^{n+1} = \mp\frac{U_k^{n+1}}{h}\left(U_{k\pm1}^{n+1} - U_k^{n+1}\right). \tag{3.127}$$

Что можно сказать о консервативности разностной схемы (3.117), (3.127)?

3.1.6. Методы переменных направлений и дробных шагов

Многие дифференциальные задачи могут быть сформулированы в виде операторных уравнений

$$\frac{\partial u}{\partial t} = Au, \tag{3.128}$$

с оператором A, допускающим аддитивное представление,

$$A = A_1 + A_2, \tag{3.129}$$

где компоненты A_1 и A_2 являются дифференциальными операторами или их дискретными приближениями. Кроме того, в большинстве практически значимых случаев вычислительная сложность задачи с полным оператором (3.129) представляется существенно более высокой по сравнению с аналогичной задачей, в которой оператор представлен только одной из компонент $A = A_1$, или $A = A_2$. Например, рассмотрим двумерное уравнение теплопроводности

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}, \tag{3.130}$$

и его одномерные аналоги

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2}, \quad \frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial y^2}. \tag{3.131}$$

Вычислительная сложность явных разностных схем для уравнений (3.130) и (3.131) вполне сравнима и может рассматриваться как оптимальная, линейно зависящая от числа узлов сетки. Однако, в случае неявных численных методов ситуация совершенно иная. Для уравнения (3.130) вычислительная сложность неявных разностных схем становится несравнимо выше по сравнению с одномерными задачами (3.131). В итоге приходится выбирать, либо использовать простые в использовании явные методы, которые однако условно устойчивы и для них требуется $\tau = O(h^2)$, либо отдать предпочтение неявным схемам, безусловно устойчивым, но не обеспечивающими оптимальной вычислительной сложности.

Существуют компромиссные решения, позволяющее совместить преимущества явных методов (простоту реализации и оптимальную вычислительную сложность) и неявных схем (безусловную устойчивость) в одной вычислительной технике. Один из таких подходов получил название **метод переменных направлений (МПН)**. В англоязычной литературе он известен под аббревиатурой ADI метод (alternative direction implicit method). Применительно к двухкомпонентной задаче (3.128),(3.129), схема МПН имеет следующий вид:

$$\frac{U^{n+1/2} - U^n}{0.5\tau} = A_1 U^{n+1/2} + A_2 U^n, \tag{3.132}$$

$$\frac{U^{n+1} - U^{n+1/2}}{0.5\tau} = A_1 U^{n+1/2} + A_2 U^{n+1}. \tag{3.133}$$

Для случая двумерного уравнения теплопроводности в качестве операторов A_1 и A_2 могут выступать матрицы конечно-разностных аппроксимаций частных производных второго порядка по x и y:

$$A_1U = \frac{U_{k-1,m} - 2U_{k,m} + U_{k+1,m}}{h_x^2}, \quad A_2U = \frac{U_{k,m-1} - 2U_{k,m} + U_{k,m+1}}{h_y^2}, \tag{3.134}$$

где $U_{k,m} = U(x_k, y_m)$, $(x_k, y_m) \in \omega_h$,

$$\omega_h = \{(x_k, y_m),\ x_k = kh_x,\ \ k = 0, 1, \ldots, K,\ y_m = mh_y,\ \ m = 0, 1, \ldots, M\}. \tag{3.135}$$

Несложно заметить, что каждое из уравнений (3.132) и (3.133) представляют собой локально неявную схему по одному из пространственных направлений. Идея, положенная в основу двухкомпонентного МПН, состоит в чередовании двух альтернативных схем, каждая из которых явная по одному пространственному направлению и неявная по другому. Чередование этих схем приводит к удивительным результатам. Данный метод обладает безусловной устойчивостью и его вычислительная сложность сравнима с методами решения одномерных задач, т.е. не выходит за рамки оптимальной вычислительной сложности. Кроме того, симметричность алгоритма на четном числе шагов позволяет получить второй порядок точности как по пространству, так и по времени.

В самом деле, исключая промежуточное значение $U^{n+1/2}$ из уравнений (3.132), (3.133), мы приходим к эквивалентному представлению схемы МПН:

$$(I - 0.5\tau A_1)(I - 0.5\tau A_2)U^{n+1} = (I + 0.5\tau A_1)(I + 0.5\tau A_2)U^n. \tag{3.136}$$

Несложно заметить, что полученная схема (3.136) с точностью до слагаемого

$$0.25\tau^2 A_1 A_2 \left(U^{n+1} - U^n\right) = O(\tau^3), \tag{3.137}$$

является приближенной факторизацией схемы Кранка – Николсон:

$$(I - 0.5\tau(A_1 + A_2))U^{n+1} = (I + 0.5\tau(A_1 + A_2))U^n \tag{3.138}$$

Таким образом, различие между схемами МПН и Кранка – Николсон не выходят за пределы погрешности последней и, следовательно, порядок аппроксимации данных схем одинаков и составляет $O(h^2 + \tau^2)$.

Пример 3.8. В качестве примера сравним вычислительные качества схем Кранка – Николсон и МПН, применительно к двумерному нестационарному уравнению теплопроводности, стационарный аналог которого рассмотрен в разделе 3.1.4:

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + f(x, y), \quad -1 < x < 1, \quad -1 < y < 1, \tag{3.139}$$

где $f(x, y) = -10\sin(2\pi x/L)\sin(2\pi y/L)$. Рассмотрим задачу Дирихле с нулевыми краевыми условиями:

$$u(\pm 1, y, t) = u(x, \pm 1, t) = 0, \quad u(x, y, t = 0) = 0. \tag{3.140}$$

Схема Кранка – Николсон для уравнения (3.139) имеет вид

$$(I - 0.5\tau A)U^{n+1} = (I + 0.5\tau A)U^n + \tau F, \tag{3.141}$$

где A — пятидиагональная матрица Пуассона, умноженная на h^{-2}, U и F — вектор - столбцы приближенного решения и функции правой части уравнения, упорядоченные по столбцам двумерного массива прямоугольной сетки: $U = U_{k,m} = U(x_k, y_m)$, $F = F_{k,m} = f(x_k, y_m)$, $(x_k, y_m) \in \omega_h$. Для простоты, полагаем, что количество узлов сетки по координатным направлениям x и y одинаково.

Схему МПН (3.139) представим в виде следующего операторного уравнения:

$$(I - 0.5\tau A_1)(I - 0.5\tau A_2)U^{n+1} = (I + 0.5\tau A_1)(I + 0.5\tau A_2)U^n + 0.5\tau F, \qquad (3.142)$$

где U и F — двумерные массивы, которые содержат значения приближенного решения и правой части в узлах сетки: $U = U_{k,m} = U(x_k, y_m)$, $F = F_{k,m} = f(x_k, y_m)$, $(x_k, y_m) \in \omega_h$, A_1 и A_2 — операторы второй разностной производной в виде трехдиагональных матриц (2.40), которые при умножении на столбцы двумерных массивов U и U^T дают приближенные значения производных по x и y соотвественно. Произведение A_1U реализуется в виде стандартного алгоритма умножения матриц. Для вычисления A_2U, мы вначале выполняем транспонирование матрицы (двумерного массива) U, а после вычисления произведения снова транспонируем полученный результат: $A_2U = \left(A_1U^T\right)^T$.

Программная реализация рассмотренного примера и результаты численных экспериментов представлены ниже. Как видно из рис. 3.9, по времени решения задачи схема МПН превосходит схему Кранка – Николсон более чем на два порядка, в то время как относительные различия между полученными решениями в пределах одного – двух процентов.

```
%% Двумерное уравнение теплопроводности %%%%%
%% Схемы Кранка -- Николсон и МПН %%%%%%%%%%
clear
L = 2;
n  =45; Nt = 3;
tau = 0.05;
%%%%   построение сетки  %%%%%%%%%%%%%%%%%%%
h  = L /(n+1);
x =  -L/2+h :h :L /2-h ;
[Y,X] = ndgrid(x,x);      %граница не включена
F =  10*sin(2*pi*X/L).*sin(2*pi*Y/L); %  AU=F
U = zeros(n,n);  % Initial Condition
A = -gallery('poisson',n )/h^2; % Матрица Пуассона
%% Неявная схема Кранка--Николсон  %%%%%%%%%%
E = eye(size(A));
Ap = E+0.5*tau*A;
Am = E-0.5*tau*A;
U1 = U;
tic
for k = 1:Nt
f=Ap*U1(:)+tau*F(:);
```

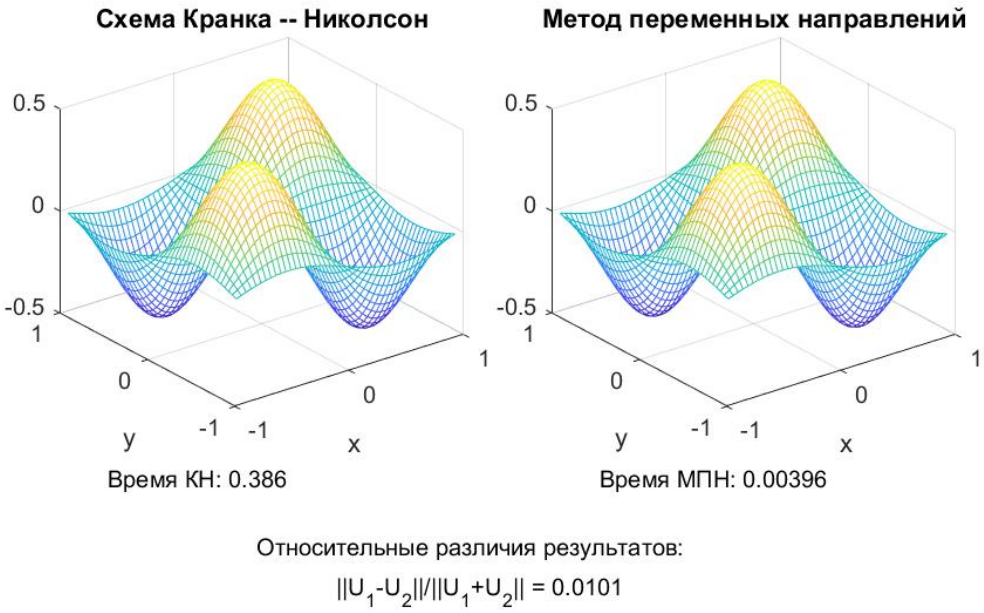


Рис. 3.9. Сравнение эффективности схем Кранка – Николсон и МПН.

```
U1=Am\f(:);
end
CN=toc
%% Схема МПН %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
e = ones(n,1);
A = spdiags([e,-2*e e], -1:1, n,n)/h^2; %
E = eye(n);
Ap = E+0.5*tau*A;
Am = E-0.5*tau*A;
U2 = U;
tic
for k=1:Nt
f = (Ap*(U2.'))'+tau*F/2;
U2 = Am\f;                % МПН x-направление
f = Ap*U2+tau*F/2;
U2 = (Am\f.').';          % МПН y-направление
end
ADI=toc
U1=reshape(U1,n,n);
```

```
D=2*norm(U2(:)-U1(:))/norm(U2(:)+U1(:));
subplot('position',[0.06,0.4, 0.41 0.5]), mesh(X,Y,U1);
title('Crank-Nicolson Scheme');
xlabel('x'); ylabel('y');
text(-3.5,-2.8,['Computational time CN: ', num2str(CN,3)]);
subplot('position',[0.56,0.4, 0.41 0.5]), mesh(X,Y,U2);
title('ADI Scheme');
xlabel('x'); ylabel('y');
text(-3.5,-2.8,['Computational time ADI: ', num2str(ADI,3)]);
subplot('position',[0.3,0.15, 0.41 0.2]),
text( -0.10,  0.3,'Relative difference between the solutions:');
text( 0.1, -0.0,['||U_1-U_2||/||U_1+U_2|| = ', num2str(D,3)]);
axis off
```

Рассмотренная схема МПН является частным случаем класса экономичных методов, вычислительная сложность которых пропорциональна числу неизвестных. Еще одним примером данного класса приближенных методов являются **методы расщепления** или **методы дробных шагов**. Общая идея данных методов состоит в том, что эволюционный операторов задачи (3.128) представляется в виде суперпозиции более простых эволюционных операторов (простых в смысле вычислительной сложности). Другими словами, исходная задача для уравнения (3.130), разбивается на совокупность задач для уравнений боле простой структуры (3.131), которые решаются последовательно, циклически, и решение каждой из задач выступают в качестве начальных условий для следующей задачи данной последовательности.

Пусть $U(x_k, y_m, t_0)$ приближенное решение уравнения (3.130) при $t = t_0$ полученное с помощью полудискретной схемы (3.128) с конечно разностной аппроксимацией дифференциальных операторов (3.134). Сеточную функцию $U(x, y, t_0)$ можно рассматривать как начальные условия задачи. Решение задачи в точке $t = t_0 + \tau$ может быть формально выражено с помощь оператора **матричной экспоненты** :

$$U(x_k, y_m, t_0 + \tau) = \exp(\tau A)\, U(x_k, y_m, t_0). \tag{3.143}$$

С другой стороны, приближенное решение этой же задачи, полученное на основе методом расщепления может быть выражено аналогичным образом через решение соответствующих подзадач (3.131):

$$\begin{aligned} \tilde{U}_1(x_k, y_m, t_0 + \tau) &= \exp(\tau A_1)\, U(x_k, y_m, t_0), \\ \tilde{U}_2(x_k, y_m, t_0 + \tau) &= \exp(\tau A_2)\, \tilde{U}_1(x_k, y_m, t_0 + \tau), \end{aligned} \tag{3.144}$$

$$\tilde{U}(x_k, y_m, t_0 + \tau) = \exp(\tau A_2) \exp(\tau A_1)\, U(x_k, y_m, t_0), \tag{3.145}$$

Норма относительной разности решений, полученных с помощью прямого метода (3.143) и метода расщепления (3.145) определяется соотвествующей разностью матричных экспонент:

$$\frac{\|\tilde{U}(x_k, y_m, t_0+\tau) - U(x_k, y_m, t_0+\tau)\|}{\|U(x_k, y_m, t_0)\|} \leqslant \|\exp(\tau A_2)\exp(\tau A_1) - \exp(\tau(A_1+A_2))\|. \tag{3.146}$$

Матричные экспоненты могут быть приближенно вычислены с помощью разложения в степенной ряд:

$$\exp(\tau A) \simeq 1 + \tau A + \frac{\tau^2}{2!}A^2 + \cdots + \frac{\tau^n}{n!}A^n. \tag{3.147}$$

Используя разложение (3.147) находим

$$\exp(\tau A_1)\exp(\tau A_2) - \exp(\tau(A_1+A_2)) = \frac{\tau^2}{2}(A_1A_2 - A_2A_1) + O(\tau^3). \tag{3.148}$$

Таким образом, рассмотренная схема расщепления (3.145) имеет первый порядок точности, поскольку локальная погрешность на каждом шаге имеет второй порядок. Как следует из уравнения (3.148), погрешность схемы расщепления (3.145) определяется отсутствием перестановочности компонент оператора: $A_1A_2 \neq A_2A_1$. В случае перестановочности компонент, $A_1A_2 = A_2A_1$, схема расщепления является точной. Кроме того, точность схемы расщепления может быть повышена при использовании схем более высокого порядка. Например метод второго порядка точности может быть получен для общего случая не перестановочных компонент, $A_1A_2 \neq A_2A_1$, при использовании **симметричной схемы расщепления**:

$$\exp(\tau(A_1+A_2)) = \exp(0.5\tau A_1)\exp(\tau A_2)\exp(0.5\tau A_1) + O(\tau^3). \tag{3.149}$$

Примечательно, что схема второго порядка точности не требует дополнительных вычислительных затрат по сравнению со схемой первого порядка (3.145) в силу следующего тождества

$$\exp(0.5\tau A_1)\exp(0.5\tau A_1) = \exp(\tau A_1), \tag{3.150}$$

вследствие чего

$$\exp(0.5\tau A_1)\exp(\tau A_2)\exp(0.5\tau A_1)\exp(0.5\tau A_1)\exp(\tau A_2)\exp(0.5\tau A_1) =$$
$$\exp(0.5\tau A_1)\exp(\tau A_2)\exp(\tau A_1)\exp(\tau A_2)\exp(0.5\tau A_1). \tag{3.151}$$

Метод расщепление может эффективно использоваться для решения сложных задач. В рамках данного подхода достигается максимальная гибкость в возможности модификации решаемой задачи без существенной переделки алгоритма (добавление и удаление дополнительных членов в уравнении во многих случаях производится естественным образом в виде дополнительных однотипных подзадач). Этот метод позволяет также комбинировать различные численные и аналитические методики, наиболее подходящие для каждой подзадачи, которые обрабатываются независимо друг от друга.

В отличие от МПН, метод расщепления может быть использован для произвольного числа компонент в аддитивном представлении оператора, включая случай

неограниченных операторов. Схема МПН является безусловно устойчивой в случае двух компонент и с небольшими изменениями может быть обобщена на трехмерный случай.

Метод расщепления наиболее эффективен при решении линейных и нелинейных задач, допускающих аддитивное представление оператора, в котором компоненты слабо связаны друг с другом. Связанность компонент определяется свойством их перестановочности и количественно может быть оценена разностью:

$$\delta = \delta(\tau) = \| \exp(\tau A_1) \exp(\tau A_2) - \exp(\tau A_2) \exp(\tau A_1) \| = O(\tau^2). \tag{3.152}$$

Крайне нежелательная ситуация при использовании метода расщепления, когда компоненты оператора (или один из них) являются сингулярными. Например, метод расщепления может приводить к большой погрешности результатов применительно к оператору Лапласа в цилиндрической (сферической) системе координат:

$$\Delta = \frac{1}{r}\frac{\partial}{\partial r} r \frac{\partial}{\partial r} + \frac{1}{r^2}\frac{\partial^2}{\partial \varphi^2}, \tag{3.153}$$

компоненты которого являются сингулярными при $r \to 0$.

Пример 3.9. Одним из наиболее известных примеров успешного использования метода расщепления является так называемый **метод дробных шагов с использованием преобразования Фурье** (**split-step Fourier method**) для решения **нелинейного уравнения Шредингера**, которое широко использования в теории нелинейных волн, в частности, для моделирования распространения оптических пучков и импульсов в нелинейных средах. Простейший пример — одномерное уравнение Шредингера с кубической нелинейностью:

$$\frac{\partial u}{\partial z} = i\frac{\partial^2 u}{\partial t^2} + i2|u|^2 u, \quad t \in [-T, T], \quad z \in [0, Z], \tag{3.154}$$

$$u(0, t) = u_0(t). \tag{3.155}$$

Задача обычно дополняется **периодическими краевыми условиями**:

$$u(z, -T) = u(z, T), \quad \frac{\partial}{\partial t} u(z, -T) = \frac{\partial}{\partial t} u(z, T). \tag{3.156}$$

Здесь $i = \sqrt{-1}$, $u = u(t, z)$ — комплексная огибающая амплитуды поля волны в зависимости от времени t при распространении вдоль оси z.

Уравнение (3.154) может быть разбито на линейную и нелинейную части:

$$\frac{\partial u}{\partial z} = i\frac{\partial^2 u}{\partial t^2} = A_1 u, \tag{3.157}$$

$$\frac{\partial u}{\partial z} = i2|u|^2 u = A_2(|u|)u. \tag{3.158}$$

В отличие от исходного уравнения (3.154), решение уравнений (3.157) и (3.158) может быть выражено аналитически. Для решения линейной части задачи наиболее эффективным является спектральный метод Фурье с использованием алгоритма

БДПФ. В данном случае, согласно общей формуле (3.144), (3.145), решение уравнения (3.157) может быть выражено посредством матричной экспоненты с показателем в виде матрицы спектрального дифференцирования второго порядка (2.79). Заметим, что при переходе в частотную область матрица спектрального дифференцирования Фурье приобретает диагональный вид. Переход в частотную область (вычисление коэффициентов Фурье) и обратно может быть эффективно выполнен с использованием БДПФ. Схема решения линейной части задачи методом Фурье имеет вид:

$$U(z+\Delta z) = F^{-1}\left[\exp\left(-i\Delta z\Omega\right) F\left[U(z)\right]\right]. \tag{3.159}$$

Здесь $U(z)$ — вектор решения на равномерной сетке,

$$U(z) = \left(U(z,t_1),\, U(z,t_2), \ldots U(z,t_N)\right)^T,$$

$$t_k = -T + (k-1)\Delta t, \quad k = 1, 2, \ldots N,$$

Δt и Δz — шаги сетки в направлении переменных t и z, $\Delta t = 2T/N$.

С учетом упорядоченности компонент вектора коэффициентов Фурье, получаемого при использовании стандартных функций БДПФ, диагональная матрица Ω должна иметь следующий вид:

$$\Omega_{k,k} = \begin{cases} \dfrac{\pi^2(k-1)^2}{T^2}, & 1 \leqslant k \leqslant N/2, \\ \dfrac{\pi^2(N-k+1)^2}{T^2}, & N/2+1 \leqslant k \leqslant N. \end{cases} \tag{3.160}$$

Заметим, что решение линейной части задачи, вычисленное согласно (3.159), является практически точным (в пределах вычислительной погрешности), когда спектр задачи имеет компактный носитель[3] в пределах частотного диапазона: $\pi/\Delta t \leqslant \omega \leqslant \pi/\Delta t$.

Решение нелинейной части задачи может быть получено аналогичным образом:

$$U(z+\Delta z) = \exp\left(i2\Delta z \cdot \operatorname{diag}\left(|U(z))\,|^2\right)\right) U(z). \tag{3.161}$$

Решение (3.161) уравнения (3.156) является точным, поскольку $|U(z)| = \text{const}$ в рамках нелинейной части задачи.

Таким образом, решение линейной и нелинейной частей задачи вычисляется практически точно. В итоге, погрешность схемы расщепления с использованием метода Фурье определяется исключительно отсутствием перестановочности линейной и нелинейной компонент оператора.

Программная реализация симметричной схемы дробных шагов (3.149) с использованием метода Фурье и результаты численных экспериментов даны ниже. Приве-

[3] Функция имеет компактный носитель если она тождественна нулю за пределами некоторого финитного подмножества области определения

денный пример демонстрирует процесс взаимодействия двух **солитонов**[4] с начальными условиями

$$u_0(t) = \frac{\exp(-ivt/2)}{\cosh(t-t_0)} + \frac{\exp(ivt/2)}{\cosh(t+t_0)}, \tag{3.162}$$

где постоянные $t_0 = 8$ и $v = 1$ определяют начальное положение ($t = \pm t_0$) и скорости солитонов соответственно. Результаты моделирования, представленные на рис. 3.10, демонстрирует одно из фундаментальных свойст данного класса нелинейных волн: форма солитонов остается неизменной при распространении и взаимодействии.

```
%%                Схема дробных шагов              %%
%%             с использованием метода Фурье  %%%%
%%        Нелинейное уравнение Шредингера       %%
clear
%%               геометрия задачи           %%%%%%%%
%%                 праметры сетки            %%%%%%%
T = 20; Z = 8;
N = 256;
dt = 2*T/N;
dz = 0.01;
gam = 2i*dz;
t = -T:dt:T-dt; z = dz:dz:Z; [TT,ZZ] = ndgrid(t,z);
UU = zeros(size(TT)); % выходной массив данных
Omega = (pi/T)^2*fftshift(-N/2:N/2-1).^2;
L = exp(- 1i*dz*Omega);
%%  начальные условия   %%%%%%%%%%%%%%%%%%%%%%%%
t_0 = T/2.5; v=2; % начальное положение и скорость
U=exp(-i*v*t/2)./cosh(t-t_0) + exp(i*v*t/2)./cosh(t+t_0);
U=U.*exp(0.5*gam*U.*conj(U)); % для симметрии схемы
for m = 1:length(z)
%%% Линейный шаг %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
U=fft(U);
U=U.*L;
U=ifft(U);
%%%%  нелинейный шаг %%%%%%%%%%%%%%%%%%%%%%%%%%%%
U=U.*exp(gam*U.*conj(U));
UU(:,m)=abs(U(:));
end
U = U.*exp(-0.5*gam*U.*conj(U));%%  для симметрии схемы
mesh(TT,ZZ,UU)
xlabel('t');ylabel('z'), zlabel('|U(z,t)|');
Um=max(UU(:));
axis([min(t),max(t),min(z),max(z), -Um/2, 2*Um])
```

[4]Солитон — это уединенная волна, распространяющаяся и взаимодействующая с себе подобными в нелинейной среде, сохраняя свою форму, подобно частицам. Солитоноподобные решения присущи целому ряду нелинейных волновых уравнений. Подробнее см. *R.K.Dodd, J.C.Eilbeck, J.D.Gibbon, and H.C.Morris. Solitons and nonlinear wave equations.(1982).*

```
title('Взаимодействие солитонов')
```

Упражнение 3.6. 1. Оцените относительную локальную погрешность симметричной схемы дробных шагов используя представление матричной экспоненты в степенной ряд.

2. Реализуйте метод дробных шагов для решения задачи Дирихле с нулевыми граничными условиями в прямоугольной области:

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}, \quad -1 < x < 1, \quad -1 < y < 1,$$

$$u(x, y, 0) = u_0(x, y) = -10 \sin(4\pi x/L) \sin(2\pi y/L).$$

Вычислите разность между результатами, полученными при использовании схемы расщепления

$$U_{12}(t+\tau) = \exp(\tau A_1) \exp(\tau A_2)\, U(t), \quad \text{и} \quad U_{21}(t+\tau) = \exp(\tau A_2) \exp(\tau A_1)\, U(t),$$

и прямым методом

$$U(t+\tau) = \exp(\tau A)\, U(t).$$

Объясните полученный результат. Допускается приближенное вычисление матричной экспоненты с использованием схемы Эйлера

$$\exp(\tau A) \simeq E + \tau A,$$

где E — единичная матрица.

3. Для задачи (3.154) – (3.156) выполняется следующий закон сохранения:

$$\frac{d}{dz} \int_{-T}^{T} |u(t, z)|^2 dt = 0. \tag{3.163}$$

Проверьте, что дискретный аналог данного инварианта,

$$P = \sum_{k=1}^{N} |U(t_k, z_n)|^2 dt = \text{const}, \quad n = 0, 1, 2, \dots \ ,$$

выполняется для схемы дробных шагов с использованием метода Фурье в рассмотренном выше примере.

4. Внося необходимые изменения в приведенную выше программную реализацию схемы дробных шагов с использованием метода Фурье, адаптируйте ее для решения задачи:

$$\frac{\partial u}{\partial z} = i \frac{\partial^2 u}{\partial t^2} - \frac{1}{6} \frac{\partial^3 u}{\partial t^3} + i2|u|^2 u, \quad t \in [-T, T], \tag{3.164}$$

$$u(0, t) = u_0(t), \tag{3.165}$$

$$u(z, -T) = u(z, T), \quad \frac{\partial}{\partial t} u(z, -T) = \frac{\partial}{\partial t} u(z, T). \tag{3.166}$$

Сравните качественное отличие в поведении решения задачи (3.154)–(3.156) и (3.164)–(3.166) при начальных условиях (3.162).

5. Используя разложение (3.147), докажите оценку (3.149).

6. Оцените погрешность схемы дробных шагов (3.128), (3.129) при использовании следующе приближенной факторизации матричной экспоненты:

$$\exp(\tau A) \simeq \frac{1}{2}\left[\exp(\tau A_1)\exp(\tau A_2) + \exp(\tau A_2)\exp(\tau A_1)\right]. \tag{3.167}$$

3.1.7. Многосеточные методы

При реализации неявных разностных схем для многомерных задач математической физики часто возникает необходимость использования итерационных методов. Рассмотрим вначале **метод простой итерации** для решения системы линейных алгебраических уравнений

$$A_h U_h = f_h, \tag{3.168}$$

с большой разреженной матрицей A_h (смысл индекса h будет пояснен ниже):

$$U_h^{(k+1)} = U_h^{(k)} - \tau\left(A_h U_h^{(k)} - f_h\right), \quad U_h^{(0)} = 0, \quad k = 0, 2, \ldots K. \tag{3.169}$$

Здесь $\tau > 0$ — итерационный параметр. Несложно заметить, что метод простой итерации (МПИ) по сути идентичен методу Эйлера, применительно к решению системы дифференциальных уравнений

$$\frac{dU_h}{dt} = -A_h U_h + f_h, \quad U(0) = 0. \tag{3.170}$$

где итерационный параметр $\tau > 0$ играет смысл шага по времени в методе Эйлера, начальные условие (3.170) имеют тот же смысл, что и начальное приближение метода простой итерации (3.169). Фактически, решение системы (3.168) есть ни что иное, как стационарное решение системы (3.170), и итерационный метод (3.169) выступает в качестве способа вычисления решения алгебраической задачи, как предела последовательности $U_h^{(k)}$,

$$U_h = \lim_{k\to\infty} U_h^{(k)},$$

не прибегая к непосредственному обращению матрицы A_h.

Предположим, что мы решаем двумерное уравнение Пуассона в прямоугольной области с нулевыми граничными условиями, и система (3.168) является системой соответствующих разностных уравнений, возникающих при аппроксимации дифференциальной задачи на равномерной сетке:

$$\omega_h = \{(x_n, y_m),\ x_k = nh_x,\ \ n = 1, 2, \ldots, K,\ y_m = mh_y,\ \ m = 1, 2, \ldots, M\}, \tag{3.171}$$

где $h_x = h_y = h$ — шаг сетки. В этом случае матрица $-A_h$ — симметричная, положительно определенная матрица Пуассона, а индекс h указывает, что соответствующая матрица соотносится с сеткой ω_h, размер шага которой равен h.

Собственные числа матрицы Пуассона определяются следующим выражением:

$$\lambda_{nm} = \frac{4}{h_x^2}\sin^2\left(\frac{n\pi}{2N_x}\right) + \frac{4}{h_y^2}\sin^2\left(\frac{m\pi}{2N_y}\right), \tag{3.172}$$
$$n = 1, 2, \dots N_x - 1, \quad m = 1, 2, \dots N_y - 1,$$

$h_x = L_x/N_x$, $h_y = L_y/N_y$. Собственные векторы матрицы A_h совпадают со значениями собственных функций дифференциального оператора Лапласа в узлах сетки:

$$\varphi_{nm}(x_i, y_j) = \sin(n\pi x_i)\cdot\sin(m\pi y_j),$$
$$(x_i, y_j) \in \omega_h,\ n = 1, 2, \dots N_x - 1,\ m = 1, 2, \dots N_y - 1. \tag{3.173}$$

Несложно показать, что

$$\lambda_{\min} = \lambda_{11} \geqslant \frac{9}{L_x^2} + \frac{9}{L_y^2}, \quad \lambda_{\max} = \lambda_{(N_x-1)(N_y-1)} \leqslant 8h^{-2}. \tag{3.174}$$

Отсюда видно, что при достаточно больших N_x и (или) N_y система дифференциальных уравнений (3.170) становится жесткой. Как следствие, шаг τ в методе Эйлера должен быть достаточо малым, в соответствии с условием устойчивости (1.54). С другой стороны, чтобы достичь стационарного решения дифференциальной системы мы должны интегрировать (3.170) на достаточно большом временном отрезке $0 \leqslant t \leqslant T$, где $T \gg \lambda_{\min}^{-1}$. Принимая во внимание аналогию между методом Эйлера и методом простых итераций, мы можем объяснить рост числа итераций в МПИ с ростом размерности матрицы A_h. В самом деле, число итераций в методе (3.169) — это то же самое, что и число шагов в методе Эйлера на отрезке $t \in [0, T]$ для достижения стационарного решения задачи (3.170). Поскольку итерационный параметр (и шаг τ) согласно условию устойчивости (см., например, (1.17)),

$$\tau \leqslant 2/\lambda_{\max} = O(h^2),$$

тогда число итераций можно оценить следующим образом:

$$K = T/\tau >> \lambda_{\max}\lambda_{\min}^{-1}/2.$$

Здесь $\lambda_{\max}\lambda_{\min}^{-1} = K_A$ — число обусловленности матрицы Пуассона A_h.

Таким образом, для рассмотренной задачи, даже при использовании оптимального значения итерационного параметра, число итераций в МПИ (3.174) возрастает пропорционально **числу обусловленности матрицы Пуассона**:

$$K = O(K_A) = O(h^{-2}). \tag{3.175}$$

Более того, для других явных итерационных методов скорость сходимости также зависит от числа обусловленности. В лучшем случае (например, для таких методов, как метод сопряженных градиентов) число итераций для достижения заданной точности имеет порядок $O(\sqrt{(K_A)})$.

Для ускорения сходимости итерационных методов в случае систем с плохо обусловленной матрицей следует использовать неявные итерационные методы с преобуславливанием:

$$B\frac{U_h^{(k+1)}-U_h^{(k)}}{\tau}=A_hU_h^{(k)}-f_h,\quad U_h^{(0)}=0,\quad k=0,2,\dots K. \tag{3.176}$$

Здесь B — невырожденная матрица той же размерности, что и матрица A_h, причем наиболее существенным при выборе матрицы B является то, что число обусловленности матрицы $B^{-1}A$ должно быть существенно меньше, чем число обусловленности матрицы A_h: $K_{B^{-1}A}\ll K_A$.

Матрицу B принято называть **переобуславливатель** итерационного метода. В самом деле, при умножении уравнения (3.176) на матрицу B^{-1}, имеем:

$$\frac{U_h^{(k+1)}-U_h^{(k)}}{\tau}=B^{-1}A_hU_h^{(k)}-B^{-1}f_h,\quad U_h^{(0)}=0,\quad k=0,2,\dots K. \tag{3.177}$$

Легко заметить, что неявный итерационный метод (3.176) эквивалентен явному методу (3.169) с *перобусловленной* матрицей $A_p=B^{-1}A$.

Заметим, что при реализации неявного метода (3.176), на каждой итерации требуется решение системы линейных уравнений с ватрицей B:

$$BU_h^{(k+1)}=F,\quad F=BU_h^{(k)}-\tau\left(A_hU_h^{(k)}-f_h\right). \tag{3.178}$$

В силу этого, вычислительная сложность решение системы с матрицей B должна быть существенно меньше чем с матицей A_h. Как следствие, переобуславливатель B выбирается обычно в виде диагональной или треугольной матрицы, элементы которой, определяются элементами матрицы A_h.

Итерационный метод (3.176) с диагональным переобуславлимателем $B=D$,

$$d_{km}=\begin{cases}a_{km}, & m=k,\\ 0, & m\neq k.\end{cases}, \tag{3.179}$$

называется **методом Якоби**. Метод Якоби привносит наибольший вклад в ускорение сходимости в случае диагонально доминирующей матрицы, когда

$$|a_{kk}|<\sum_{m\neq k}|a_{km}|,\quad \text{и}\quad \frac{\max\limits_k(|a_{kk}|)}{\min\limits_k(|a_{kk}|)}\gg 1. \tag{3.180}$$

Более существенное ускорение сходимости может быть получено при использовании **метода последовательной верхней релаксации** (в англоязычной литературе известен под аббревиатурой SOR — sequential over-relaxation) с переобуславливателем

$$B=D+\omega R,\quad 0<\omega<2,\qquad r_{km}=\begin{cases}a_{km}, & m>k,\\ 0, & m\leqslant k.\end{cases} \tag{3.181}$$

Здесь D — диагональная матрица, построенная из соответствующих элементов матрицы A_h, R — верхняя треугольная часть матрицы A_h, не включая главную диагональ.

Важно отметить, что при решении систем алгебраических уравнений с разреженной матрицей, возникающих, например, при дискретизации многомерных задач для уравнений с частными производными, для неявных итерационных методов с переобуславливателями типа SOR и т.п., скорость сходимости итераций, как правило, деградирует при увеличении пространственного разрешения сетки. В силу отмеченных обстоятельств особую актуальность представляет разработка таких итерационных методов для систем с матрицей типа матрицы Пуассона, в которых бы оказалось возможным устранить зависимость скорости сходимости итераций от размерности матрицы . Один из способов решения данной проблемы состоит в использовании итерационных многосеточных методов, на рассмотрении которых мы остановимся подробнее.

Прежде всего заметим, что погрешность МПИ на k-й итерации, $\delta_h^{(k)} = U_h^{(k)} - U_h$, может быть представлена рекурсивной зависимостью

$$\delta_h^{(k)} = \delta_h^{(k-1)} + \tau r_h^{(k-1)} = (I - \tau A_h)\, \delta_h^{(k-1)}, \tag{3.182}$$

где $r_h^{(k)} = f_h - A_h U_h^{(k)} = -A_h \delta_h^{(k)}$ невязка приближенного решения $U_h^{(k)}$. Как следствие, для погрешности МПИ выполняется следующая оценка:

$$\|\delta_h^{(k)}\| \leqslant \|I - \tau A_h\|\, \|\delta_h^{(k-1)}\|. \tag{3.183}$$

Очевидно, что максимальная скорость сходимостипри МПИ достигается при оптимальном значении итерационного параметра τ, при котором $\|I - \tau A_h\|$ принимает минимальное значение. Если посмотреть на итерационную последовательность (3.182) через призму спектрального метода Фурье, несложно заметить, что каждая собственная мода решения имеет свое собственное оптимальное значение итерационного параметра. А именно, для Фурье компоненты $\delta_{nm}^{(k)}$, соответствующей собственному значению λ_{nm} матрицы A_h, мы имеем

$$|\delta_{nm}^{(k)}| = q_{nm} \cdot |\delta_{nm}^{(k-1)}|, \quad q_{nm} = |1 - \tau\lambda_{nm}|, \tag{3.184}$$

и оптимальные значения итерационных параметров соотносятся со спектром матрицы следующим образом:

$$\tau_{\text{opt}} = \tau_{\text{opt}}(\lambda_{nm}) = \lambda_{nm}^{-1}. \tag{3.185}$$

Однако, если $\lambda_{\max}/\lambda_{\min} >> 1$ то $\tau_{\text{opt}}(\lambda_{\min}) >> \tau_{\text{opt}}(\lambda_{\max})$, и более того,

$$\|I - \tau_{\text{opt}}(\lambda_{\min})A\| > 1.$$

Как следствие, МПИ с параметром $\tau = \tau_{\text{opt}}(\lambda_{min})$ будет расходится. С другой стороны, скорость сходимости медленных (низкочастотных) мод в итерационном методе с параметром $\tau = \tau_{\text{opt}}(\lambda_{\max})$ будет чрезвычайно низкой.

Оптимальное значение итерационного параметра для МПИ определяется соотношением:

$$\tau_{\text{opt}} = \frac{2}{\lambda_{\min} + \lambda_{\max}}. \tag{3.186}$$

Скорость сходимости при $\tau = \tau_{\text{opt}}$ определяется оценкой

$$\|\delta^{(k)}\| \leqslant \frac{1 - \lambda_{\min}\lambda_{\max}^{-1}}{1 + \lambda_{\min}\lambda_{\max}^{-1}} \|\delta^{(k-1)}\| \leqslant \left(\frac{1 - K_A^{-1}}{1 + K_A^{-1}}\right)^k \|\delta^{(0)}\|, \tag{3.187}$$

Можно заметить, что скорость сходимости падает, когда число обусловленности матрицы Пуассона, определенное выражением (3.175), становится слишком большим при $h \to 0$.

Идея **многосеточного метода** состоит в отделении низкочастотных, медленно сходящихся составляющих решения с целью их независимой обработки на **грубой сетке**. Как работает многосеточный метод легко понять на простом примере двухсеточного метода.

Пусть нам необходимо решить задачу Дирихле для двумерного уравнения Пуассона в прямоугольной области методом конечных разностей (3.66) с использованием итерационного метода (3.169) для решения соответствующей алгебраической задачи (3.168). Требуется найти решение на достаточно **мелкой сетке** ω_h с шагом h. Используем грубую сетку ω_{2h} с шагом $2h$, $\omega_{2h} \subset \omega_h$, для ускорения сходимости низкочастотных составляющих приближенного решения.

Суть двухсеточного метода в том, что вначале мы делаем несколько итераций (3.169) с итерационным параметром $\tau = (3/2)/\lambda_{\max}$. Эта стадия называется **сглаживание**, поскольку каждая итерация подобна низкочастотному фильтру, который эффективно подавляет высокочастотную составляющую погрешности решения. Фактически, в диапазоне собственных значений

$$\frac{\lambda_{\max}}{4} \leqslant \lambda_{nm} \leqslant \lambda_{\max}$$

Фурье-компоненты погрешности подавляются согласно (3.184) с коэффициентом q_{nm}:

$$q_{nm} = \left|1 - \frac{3\lambda_{nm}}{2\lambda_{\max}}\right| \leqslant \frac{5}{8}. \tag{3.188}$$

После 3-5 *сглаживающих* итераций мы получаем приближенное решение со сглаженной погрешностью, в которой подавлены высокочастотные составляющие, а значит данная погрешность может быть достаточно точно представлена на грубой сетке. Для этого мы вычисляем невязку $r_h = f_h - A_h U_h$ и проектируем ее на грубую сетку ω_{2h}:

$$r_{2h} = R r_h. \tag{3.189}$$

Здесь R — **оператор проекции на грубую сетку**, который, например, может состоять в прореживании вектора r_h путем выборки его компонент в четных узлах и формировании вектора вдвое меньшей размерности r_{2h}.

Следующий шаг состоит в вычислении погрешности приближенного решения на грубой сетке в соответствии с полученным вектором невязки r_{2h}:

$$A_{2h}\delta_{2h} = r_{2h}, \quad \delta_{2h} = A_{2h}^{-1} r_{2h}. \tag{3.190}$$

Поскольку размерность задачи и число обусловленности матрицы на грубой сетке существенно меньше, чем на мелкой, то система (3.190) может быть решена с

помощью прямых или итерационных методов с существенно меньшими вычислительными затратами по сравнению с исходной задачей (3.168). Имея приближенное решение на мелкой сетке и хорошее приближение для его погрешности на грубой, мы можем отобразить (продолжить) полученную погрешность на мелкую сетку и вычесть ее из приближенного решения. Данная процедура называется **коррекцией с грубой сетки**:

$$U_h = U_h - P\delta_{2h}. \tag{3.191}$$

Здесь P — **оператор продолжения**, который дополняет вектор на грубой сетке до вектора на мелкой сетке, например, с посредством интерполяции. В результате коррекции решения на мелкой сетке мы получаем приближенное решение со сравнительно низкой погрешность в низкочастотной части его Фурье спектра. Остаточная погрешность присутствует преимущественно в высокочастотных составляющих решения.

В завершение двухсеточной процедуры разумно выполнить еще несколько сглаживающих итераций, для подавления остаточной погрешности в высокочастотной части спектра после коррекции с грубой сетки. Это могут быть 3-5 итераций (3.169) с итерационным параметром $\tau = (3/2)/\lambda_{\max}$.

Подводя итог, напомним основные шаги двухсеточного итерационного метода:

- 3-5 сглаживающих итераций (3.169);
- проекция невязки на грубую сетку согласно (3.189);
- вычисление погрешности решения на грубой сетке (3.190);
- коррекция решения с грубой сетки (3.191);
- 3-5 сглаживающих итераций (3.169);

На стадиях сглаживания могут использоваться не только МПИ, но и другие итерационные методы, такие как метод Якоби или последовательной верхней релаксации, с помощью которых может быть достигнут лучший сглаживающий эффект.

Описанный выше алгоритм является примером двухсеточного **v-цикла**. Использование такого цикла из 6-10 итераций может подавлять погрешность в фиксированное число раз, независимо от того, какой шаг h использован в мелкой сетке. Обычно, один v-цикл с пятью сглаживающими итерациями вначале и в конце позволяет получит снижения погрешности на фактор $\rho \simeq 1/10$.

Данный алгоритм может быть применен рекурсивно с использованием нескольких грубых сеток различного пространственного разрешения: $\omega_h,\ \omega_{2h}, \ldots \omega_{nh}$. Последовательность таких вложенных v-циклов называется **V-циклом**:

$$V = [\omega_h; \omega_{2h}; \omega_{4h}; \omega_{8h}; \omega_{4h}; \omega_{2h}; \omega_h], \tag{3.192}$$

В данном случае V-цикл включает четыре уровня пространственного разрешения. Схематически данный V-цикл представлен на рис. 3.11.

Большинство многосеточных алгоритмов строится на основе комбинации V-циклов различной глубины. Например, так называемый **W-цикл** имеет следующую структуру сеток:

$$W = [\omega_h; \omega_{2h}; \omega_{4h}; \omega_{8h}; \omega_{4h}; \omega_{8h}; \omega_{4h}; \omega_{2h}; \omega_{4h}; \omega_{8h}; \omega_{4h}; \omega_{2h}; \omega_h]. \tag{3.193}$$

Еще один пример многосеточного метода — **полный многосеточный цикл**, который использует следующую последовательность сеток:

$$FMG = [\omega_{8h}; \omega_{4h}; \omega_{8h}; \omega_{4h}; \omega_{2h}; \omega_{4h}; \omega_{8h}; \omega_{4h}; \omega_{2h}; \omega_h; \omega_{2h}; \omega_{4h}; \omega_{8h}; \omega_{4h}; \omega_{2h}; \omega_h], \tag{3.194}$$

Численные эксперименты показывают, что эффективность одного W-цикла, или полного многосеточного цикла, превосходит эффективность двух V-циклов.

Отметим, что в двумерном случае одна итерация на грубой сетке ω_{kh} в 4^{k-1} раз быстрее чем чем одна итерация на мелкой сетке ω_h. Более того, все вычисления на последовательности грубых сеток в многосеточном цикле произвольной глубины по вычислительным затратам не превосходят вычислений на мелкой сетке. Поскольку число итераций на каждой сетке многосеточного метода полагается фиксированным, то общее число итераций практически не зависит от размерности матрицы и вычислительная сложность метода в целом находится в пределах оптимального значения $O(N)$, где N число узлов мелкой сетки. Важно подчеркнуть, что во многих приложениях многосеточные итерационные методы являются практически единственной универсальной техникой, число итераций в которой не зависит от размерности сетки.

Пример 3.10. Продемонстрируем преимущество многосеточного метода на примере реализации разностной схемы для решения двумерного уравнения Пуассона:

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = \cos\left(\frac{\pi x}{L}\right)\cos\left(\frac{\pi y}{L}\right), \tag{3.195}$$

$$-L/2 < x < L/2, \quad -L/2 < y < L/2, \quad L = 2,$$

с нулевыми краевыми условиями

$$u(\pm L/2, y) = u(x, \pm L/2) = 0, \tag{3.196}$$

$$u(\pm L/2, y) = y, \quad u(x, \pm L/2) = x. \tag{3.197}$$

Аппроксимируем задачу разностной схемой (3.66). Далее, воспользуемся двухсеточным v-циклом, применяя МПИ для итераций сглаживания. В качестве операторов проекции и продолжения для перехода между грубой и мелкой сетками воспользуемся процедурами прореживания и линейной интерполяции соответственно. Программная реализация описанного многосеточного алгоритма и некоторые иллюстрации результатов численных экспериментов представлены ниже. В частности, для сравнения на рис. 3.12 показана динамика сходимости МПИ с оптимальным значением итерационного параметра при решении рассмотренной задачи на грубой и мелкой сетках. Как видно из представленных результатов, относительная погрешность 10^{-4} достигается на грубой сетке примерно за 700, в то время как на грубой сетке число итераций не превосходит 200.

В случае двухсеточного метода с тремя итерациями сглаживания, относительная погрешность 10^{-4} достигается после одного v-цикла независимо от размера сетки (см. рис. 3.13).

```
%% Многосеточный метод для уравнения Пуассона %%%
%%  с краевыми условиями Дирихле  %%%%
clear
L = 2;
n = 21;  n2 = 11;
n=27;  n2 = 14;
% n=51;  n2 = 26;
%%%% генерация сетток  %%%%%%%%%%%%%%%%%%%%
h = L/(n-1);   % шаг мелкой сетки
H = L/(n2-1);  % шаг грубой сетки
x  = -L/2 :h :L/2;   [X,Y] = meshgrid(x,x);   %   мелкая сетка
x2 = -L/2 :H :L/2; [X2,Y2] = meshgrid(x2,x2); %   грубая сетка
mask = ones(size(X));
mask(1:2:end,1:2:end) = mask(1:2:end,1:2:end)*0;
down=find(mask(2:n-1,2:n-1) == 0);
%%%  характеризация мелкой сетки       %%%%%%%%%%%%%%%%%%%%%%%%
Ah = -gallery('poisson',n-2 )/h^2;      % матрица Пуассона
L_Ah = 4/h^2*(sin(pi*h*(1:n)/4)).^2+...
4/h^2*(sin(pi*h*(1:n)/4)).^2;  % собственные значения матрицы
tau_0 = 2/(L_Ah(1)+L_Ah(end));  % опт. итерационный параметр
%%%  характеризация грубой сетки       %%%%%%%%%%%%%%%%%%%%%%%%
AH = -gallery('poisson',n2-2 )/H^2;  % матрица Пуассона
L_AH = 4/H^2*(sin(pi*(1:n2-1)/n2/2)).^2+...
4/H^2*(sin(pi*(1:n2-1)/n2/2)).^2;%собственные значения матрицы
Tau_0 = 2/(L_AH(1)+L_AH(end))
tau = tau_0*3/4;  Tau = Tau_0*3/4;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Fh = (cos(pi*X/2).*cos(pi*Y/2)).^1;
Err0 = zeros(size(Fh));
Err1 = Err0;
Err  = Err0;
U = zeros(size(Fh)); U0 = Fh(2:end-1,2:end-1);
FH = reshape(Fh(down),n2-2,n2-2); Fh = Fh(2:end-1,2:end-1);
%%                точное решение     %%%%%%%%%%%%%%%%%%%%%%
UU=Ah\Fh(:);  U0 = reshape(UU,n-2,n-2);
U(2:end-1,2:end-1)=U0;
UT=AH\FH(:);
% subplot(2,2,1), mesh(X,Y,reshape(U,n,n));
% title('Exact Solution')
%%     Метод простой итерации     %%%%%%%%%%%%%%
KK=800;
Uh = zeros(size(Fh(:)));UH = zeros(size(FH(:)));
Err_SIM=zeros(KK,1);
for k=1:KK
Uh=Uh-tau_0*(Fh(:)-Ah*Uh);
```

```
Err_SIM(k)=norm(Uh-U0(:))/norm(U(:));
UH=UH-Tau_0*(FH(:)-AH*UH);
Err_SIM_H(k)=norm(UH-UT(:))/norm(UT(:));
end
%%    операции сглаживания    %%%%%%%%%%%%%%%%%%
Uh = zeros(size(Fh(:)));  % начальное приближение
K = 5; % 3-5 итераций сглаживания в нвчале и конце
for k=1:K
Uh = Uh - tau*(Fh(:) - Ah*Uh);
end
Err0(2:end-1,2:end-1) = reshape(Uh(:),n-2,n-2) -U0;
subplot('position',[0.05 0.53 0.42 0.40]), mesh(X,Y,(Err0));
title('Pre-smoothing Error')
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% расчет невязки на грубой сетке %%%%%%%%%%%%%
rh = -Fh(:)+Ah*Uh; %невязка на мелкой сетке
rH = rh(down);     %проекция на грубую сетку
eH = AH\rH;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%  Коррекция с грубой сетки   %%%%%%%%%%%%%%%%
eH = reshape(eH,n2-2,n2-2);
EE = zeros(size(X2));
EE(2:end-1,2:end-1) = eH;   % продолжение на мелкую сетку
EE = interp2(X2,Y2,EE,X,Y); % интерполяция
eh = EE(2:end-1,2:end-1);
eh = eh(:);
Uh = Uh-eh(:);
Err1(2:end-1,2:end-1)=reshape(Uh,n-2,n-2)-U0;
subplot('position',[0.30 0.03 0.42 0.40 ]), mesh(X,Y,Err1);
limits=axis;
title('Coarse-Grid Correction Error')
%% итерации сглаживания  %%%%%%%%%%%%%%%%%%%%%%%%%%
tau = 3/4*2/(L_Ah(1)+L_Ah(end))
for k = 1:K
Uh = Uh-tau*(Fh(:)-Ah*Uh);
end
Err(2:end-1,2:end-1) = reshape(Uh,n-2,n-2)-U0;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
subplot('position',[0.53 0.53 0.42 0.40 ]), mesh(X,Y,Err);
title('Post-Smoothing Error')
axis(limits);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
figure
out = 1:20:KK;
semilogy(out,Err_SIM(out),'.-',out,Err_SIM_H(out),'o-')
xlabel('Iterations');
```

```
ylabel('Relative Accuracy')
legend('\omega_h','\omega_{2h}')
title('Convergence of SIM'); grid
```

Упражнение 3.7.

1. В примере раздела 3.1.4, замените прямой метод решения системы линейных алгебраических уравнений на метод простой итерации. Оцените число итераций для достижения относительной нормой невязки значения 10^{-5} при различных значениях n: $n = 31; 61; 121$.

2. Проверьте как результаты, представленные на рис. 3.12 согласуются с теоретической оценкой (3.175) .

3. В рассмотренном выше примере, замените МПИ на метод сопряженных градиентов (СГ), используя для этой цели функцию matlab **pcg**. Сравните зависимость числа итераций от размерности сетки n в случае МПИ и СГ.

4. В рассмотренном выше примере, замените метод простой итерации на метод последовательной верхней релаксации (SOR) (3.178), (3.181), используя $1.6 < \omega \leqslant 1.8$. Сравните число итераций для достижения заданной точности в зависимости от n для МПИ и SOR.

5. По аналогии с рассмотренным выше примером, реализуйте четырехсеточный V-цикл. Сравните погрешность приближенного решения после двух- и четырехсеточного V-цикла с пятью сглаживающими итерациями.

6. Реализуйте четырехсеточный W-цикл (3.193) используя в качестве отправной точки соответствующий V-цикл. Сравните погрешность решения после W-цикла и двух последовательных V-циклов.

3.2. Введение в метод конечных элементов

3.2.1. Слабая формулировка дифференциальной задачи.

Дифференциальные уравнения играют важную роль не только в математике, но и в многочисленных приложениях, поскольку дифференциальная форма математических моделей зарекомендовала себя как наиболее адекватная и продуктивная для решения широкого круга задач науки и техники. Можно сказать, что каждая прикладная дифференциальная задача имеет в своей основе соотвествующее представление закономерностей, отражающих сущность описываемого объекта или явления. Как правило, это наиболее фундаментальные и общие законы, такие как законы сохранения энергии, массы, момента движения и т.п. Преимущества дифференциальных моделей ассоциируются прежде всего с их гибкостью и подробностью в непрерывной передаче закономерностей в каждой точке пространства и времени.

Когда мы имеем дело с численными методами решения дифференциальных задач, мы заменяем дифференциальную модель на соответствующую ей дискретную. Эффективность численных методов определяется близостью (согласованностью) дискретной модели и ее дифференциального прототипа. Мерой согласованности дискретных моделей является невязка. Напомним, что под невязкой понимается

дисбаланс, возникающий при подстановке точного решения в приближенную дискретную модель. Для того, чтобы максимально согласовать дискретную модель с ее дифференциальным оригиналом необходимо минимизировать ее невязку.

Глядя через призму основных принципов численного анализа, легко понять основные идеи метода конечных элементов: Зачем нужна слабая формулировка задачи? Какое место занимает метод Галеркина в формировании конечно-элементных дискретных моделей?

В этом разделе мы рассмотрим подробно особенности метода конечных элементов применительно к решению задачи Дирихле для двумерного эллиптического уравнения в прямоугольной области Ω с границей $\partial\Omega$:

$$\frac{\partial}{\partial x}\lambda(x,y)\frac{\partial u}{\partial x}+\frac{\partial}{\partial y}\lambda(x,y)\frac{\partial u}{\partial y}=f(x,y),\quad (x,y)\in\Omega, \tag{3.198}$$

$$u(x,y)=0,\quad (x,y)\in\partial\Omega. \tag{3.199}$$

Первый шаг в построении метода конечных элементов состоит в переходе к слабой формулировке дифференциальной задачи. Слабая формулировка для (3.198) получается путем умножения данного уравнения на достаточно гладкую пробную функцию $v(x,y)$: $v(x,y)=0$, $(x,y)\in\partial\Omega$, с последующим интегрированием полученного равенства по области Ω:

$$\iint\limits_{\Omega}\left[\frac{\partial}{\partial x}\lambda(x,y)\frac{\partial u}{\partial x}v+\frac{\partial}{\partial x}\lambda(x,y)\frac{\partial u}{\partial y}v\right]dxdy=\iint\limits_{\Omega}f(x,y)v\,dxdy. \tag{3.200}$$

Используя формулы интегрирования по частям будем иметь:

$$\iint\limits_{\Omega}\lambda(x,y)\left[\frac{\partial u}{\partial x}\frac{\partial v}{\partial x}+\frac{\partial u}{\partial y}\frac{\partial v}{\partial y}\right]dxdy+\iint\limits_{\Omega}f(x,y)v\,dxdy=0. \tag{3.201}$$

Уравнение (3.201) называют **слабая формулировка** краевой задачи (3.198)–(3.199), а u — **слабое решение**. В терминах скалярного произведения $(\cdot,\cdot)$ слабая формулировка может быть представлена в виде:

$$b(u,v)=(f,v), \tag{3.202}$$

где скалярное произведение выражается следующим образом:

$$(u,v)=\iint\limits_{\Omega}uv\,dxdy, \tag{3.203}$$

где $u,v\in H_0^1(\Omega)$, H_0^1 — пространство Соболева функций, интегрируемых вместе с первыми производными в области Ω, $b(u,v)$ — билинейная форма в пространстве H_0^1.

Предпочтительность слабой формулировки (3.201) состоит в том, что данная интегральная форма снижает требования к дифференцируемости решения (требуется наличие только первых производных решения). Кроме того, слабая формулировка задачи упрощает дальнейшие шаги в построении конечно-элементной модели.

3.2.2. Конечно-элементная дискретизация

Следующий шаг состоит в дискретизации области и выборе базисных функций. В двумерном случае область Ω разбивается обычно на треугольные подобласти. Однако, вообще говоря, для этих целей могут использоваться семейства произвольных многоугольников Ω_n (включая криволинейные многоугольники): $\overline{\Omega} = \cup\Omega_n$. Слабая формулировка задачи позволяет естественным образом провести декомпозицию области в систему подобластей, поскольку интегрирование в (3.201) может быть выполнено путем суммирования интегралов для каждой подобласти Ω_n в отдельности.

Следующая идея, лежащая в основе метода конечных элементов, состоит в замене бесконечномерной задачи (3.202) конечномерным аналогом. Полагаем, что решение задачи (3.202) может быть представлено точно (или приближенно) бесконечной (или конечной) комбинацией некоторых базисных функций $\varphi_k(x, y)$:

$$u(x, y) = \sum_{k=1}^{\infty} \alpha_k \varphi_k(x, y) \simeq \sum_{k=1}^{N} \alpha_k \varphi_k(x, y). \tag{3.204}$$

Приближенное решение и аппроксимирующее подпространство может быть получено путем перехода к конечномерному базису в представлении (3.204).

В качестве аппроксимирующего конечномерного подпространства H_0^1 мы будем использовать множество кусочно полиномиальных функций, удовлетворяющих краевым условиям и имеющих компактный носитель в пределах отдельной ячейки сетки. Простейший пример базисных функций — кусочно-линейные функции. Мы уже описали пример такого подпространства в одномерном случае (см. раздел 2.2).

Полагаем, что в двумерном случае решение задачи (3.202) аппроксимируется в пределах треугольной ячейки сетки линейной функцией:

$$u(x, y) = \alpha_1 + \alpha_2 x + \alpha_3 y. \tag{3.205}$$

Сравнивая (3.205) и (3.204), находим, что функции φ_k в данном представлении имеют вид: $\varphi_1(x, y) = 1$, $\varphi_2(x, y) = x$, $\varphi_3(x, y) = y$. Отметим, что данные функции формы обращаются в ноль вне заданного треугольника. Коэффициенты α_k, $k = 1, 2, 3$, могут быть определены однозначно, как решение системы линейных алгебраических уравнений

$$\begin{bmatrix} 1 & x_1 & y_1 \\ 1 & x_2 & y_2 \\ 1 & x_3 & y_3 \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \end{bmatrix} = \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix}, \tag{3.206}$$

где $u_k = u(x_k, y_k)$, $k = 1, 2, 3$, (x_k, y_k) — координаты вершин треугольника.

Что же такое **конечный элемент**? Под конечными элементами понимают ячейки сетки с определенными на них **функциями формы** — базисные функции $\psi_k(x, y)$: $\psi_k(x_n, y_n) = \delta_{k,m}$. В рассмотренном случае мы имеем дело с линейными треугольными конечными элементами. В случае квадратичных или кубических функций формы мы имеем дело с **конечными элементами высшего порядка**.

Среди возможных вариантов базисных функций, функции формы наиболее

удобны тем, в этом случае коэффициенты в разложении (3.204) совпадают со значениями решения в вершинах ячейки сетки:

$$u(x,y)=\sum_{k=1}^{3}u_k\psi_k(x,y). \tag{3.207}$$

Преобразование функций формы к такому удобному базису можно осуществить на основе следующего соотношения:

$$u(x,y)=[1\ x\ y]\begin{bmatrix}1 & x_1 & y_1\\ 1 & x_2 & y_2\\ 1 & x_3 & y_3\end{bmatrix}^{-1}\begin{bmatrix}u_1\\ u_2\\ u_3\end{bmatrix}=\sum_{k=1}^{3}u_k\psi_k(x,y), \tag{3.208}$$

где

$$\psi_1(x,y)=\frac{(x_2y_3-x_3y_2)+(y_2-y_3)x+(x_3-x_2)y}{D}, \tag{3.209}$$

$$\psi_2(x,y)=\frac{(x_3y_1-x_1y_3)+(y_3-y_1)x+(x_1-x_3)y}{D}, \tag{3.210}$$

$$\psi_3(x,y)=\frac{(x_1y_2-x_2y_1)+(y_1-y_2)x+(x_2-x_1)y}{D}, \tag{3.211}$$

$$D=x_2y_3-x_3y_2+y_1x_3-x_1y_3+x_1y_2-y_1x_2. \tag{3.212}$$

Мы видим, что новые базисные функции $\psi_k(x,y)$ также кусочно линейны и для вершин треугольника (x_m,y_m) имеем:

$$\psi_k(x_m,y_m)=\begin{cases}1, & k=m,\\ 0, & k\neq m.\end{cases} \tag{3.213}$$

Представление (3.207) имеет интерполяционный характер, и коэффициенты u_k совпадают с соответствующими значения интерполируемой функции в узлах интерполяции ($u(x_k,y_k)=u_k$). Преимущество использования базисных функций (3.209)–(3.208) состоит в упрощении процедуры последующей обработки полученного дискретного решения.

В простейшем случае линейных конечных элементов градиент приближенного решения является постоянным в пределах фиксированного элемента. Как следует из (3.209)–(3.212),

$$\frac{\partial\psi_1}{\partial x}=\frac{y_2-y_3}{D},\quad \frac{\partial\psi_2}{\partial x}=\frac{y_3-y_1}{D},\quad \frac{\partial\psi_3}{\partial x}=\frac{y_1-y_2}{D}, \tag{3.214}$$

$$\frac{\partial\psi_1}{\partial y}=\frac{x_3-x_2}{D},\quad \frac{\partial\psi_2}{\partial y}=\frac{x_1-x_3}{D},\quad \frac{\partial\psi_3}{\partial y}=\frac{x_2-x_1}{D}. \tag{3.215}$$

Пример 3.11. Рассмотрим линейный случай и способы повышения порядка полиномиальных функций формы. Конечные элементы высших порядков представляют интерес для получения более точных дискретных моделей. Отметим, что с

ростом порядка функций формы возрастает и их количество. Например, базис линейных полиномов определяется $N = n + 1$ функциями формы, где n — число независимых переменных. Вместе с тем мы должны использовать $n + 1$ контрольные точки, для того, чтобы определить произвольный линейный полином в пространстве n переменных однозначно. В частности, в двумерном случае для определения произвольной кусочно–линейной функции требуется три контрольных точки, и вершины треугольника являются наиболее естественными контрольными точками для построения линейных конечных элементов на плоскости. В общем случае число контрольных точек совпадает с числом функций формы.

1. Для линейного конечного элемента с треугольной геометрией, представленной на рис. 3.14 (слева): $x_1 = 0$, $x_2 = h$, $x_3 = 0$, $y_1 = 0$, $y_2 = 0$, $y_3 = h$, согласно (3.209)–(3.211), $D = h^2$:

$$\psi_1(x, y) = 1 - x/h - y/h, \quad \psi_2(x, y) = x/h, \quad \psi_3(x, y) = y/h. \tag{3.216}$$

2. Квадратичные полиномы двух переменных представимы с помощью следующих функций формы:

$$\begin{gathered} \varphi_1(x, y) = 1,\ \varphi_2(x, y) = x,\ \varphi_3(x, y) = y, \\ \varphi_4(x, y) = xy,\ \varphi_5(x, y) = x^2,\ \varphi_6(x, y) = y^2. \end{gathered} \tag{3.217}$$

Для определения коэффициентов квадратичного конечного элемента с треугольной геометрией нам потребуется три дополнительные контрольные точки. Для этих целей, например, могут использоваться средние точки сторон треугольника (см. рис. 3.14, (справа)):

$$\begin{gathered} x_1 = 0, \quad x_2 = h/2, \quad x_3 = h, \quad x_4 = h/2, \quad x_5 = 0, \quad x_6 = 0, \\ y_1 = 0, \quad y_2 = 0, \quad y_3 = 0, \quad y_4 = h/2, \quad y_5 = h, \quad y_6 = h/2. \end{gathered}$$

Дальнейшие вычисления для квадратичного элемента аналогичны линейному случаю с той лишь разницей, что количество функций формы удваивается. Согласно (3.208), используя функции символьных вычислений Matlab (Symbolic Math ToolBox), имеем:

$$\Phi = \begin{bmatrix} \varphi_1(x_1, y_1) & \varphi_2(x_1, y_1) & \varphi_3(x_1, y_1) & \varphi_4(x_1, y_1) & \varphi_5(x_1, y_1) & \varphi_6(x_1, y_1) \\ \varphi_1(x_2, y_2) & \varphi_2(x_2, y_2) & \varphi_3(x_2, y_2) & \varphi_4(x_2, y_2) & \varphi_5(x_2, y_2) & \varphi_6(x_2, y_2) \\ \varphi_1(x_3, y_3) & \varphi_2(x_3, y_3) & \varphi_3(x_3, y_3) & \varphi_4(x_3, y_3) & \varphi_5(x_3, y_3) & \varphi_6(x_3, y_3) \\ \varphi_1(x_4, y_4) & \varphi_2(x_4, y_4) & \varphi_3(x_4, y_4) & \varphi_4(x_4, y_4) & \varphi_5(x_4, y_4) & \varphi_6(x_4, y_4) \\ \varphi_1(x_5, y_5) & \varphi_2(x_5, y_5) & \varphi_3(x_5, y_5) & \varphi_4(x_5, y_5) & \varphi_5(x_5, y_5) & \varphi_6(x_5, y_5) \\ \varphi_1(x_6, y_6) & \varphi_2(x_6, y_6) & \varphi_3(x_6, y_6) & \varphi_4(x_6, y_6) & \varphi_5(x_6, y_6) & \varphi_6(x_6, y_6) \end{bmatrix}^{-1} =$$

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ -3/h & 4/h & -1/h & 0 & 0 & 0 \\ -3/h & 0 & 0 & 0 & -1/h & 4/h \\ 4/h^2 & -4/h^2 & 0 & 4/h^2 & 0 & -4/h^2 \\ 2/h^2 & -4/h^2 & 2/h^2 & 0 & 0 & 0 \\ 2/h^2 & 0 & 0 & 0 & 2/h^2 & -4/h^2 \end{bmatrix}, \tag{3.218}$$

В итоге, новые базисные функции ψ_m имеют вид:

$$[\psi_1, \psi_2, \ldots, \psi_6] = [\varphi_1, \varphi_2, \ldots, \varphi_6]\Phi \tag{3.219}$$

$$\begin{aligned}
&\psi_1 = (2x^2)/h^2 - (3y)/h - (3x)/h + (2y^2)/h^2 + (4xy)/h^2 + 1\\
&\psi_2 = 4x/h - 4x^2/h^2 - 4xy/h^2,\\
&\psi_3 = 2x^2/h^2 - x/h,\\
&\psi_4 = 4xy/h^2,\\
&\psi_5 = 2y^2/h^2 - y/h,\\
&\psi_6 = 4y/h - 4y^2/h^2 - 4xy/h^2
\end{aligned} \tag{3.220}$$

Билинейный базис может быть получен на основе следующих функций формы:

$$\varphi_1(x,y) = 1,\ \varphi_2(x,y) = x,\ \varphi_3(x,y) = y,\ \varphi_4(x,y) = xy. \tag{3.221}$$

Такой базис может быть использован для построения конечных элементов с четырехугольной геометрией или треугольниками с дополнительной четвертой точкой в центре.

Упражнение 3.8. 1. Построить билинейный базис $\psi_k(x_m, y_m) = \delta_{km}$, $k = 1, 2, 3, 4$, используя функции формы (3.221) для треугольной геометрии

$$x_1 = 0, \quad x_2 = h, \quad x_3 = 0, \quad x_4 = h/4,$$

$$y_1 = 0, \quad y_2 = 0, \quad y_3 = h, \quad y_4 = h/2.$$

2. Построить билинейный базис $\psi_k(x_m, y_m) = \delta_{km}$, $k = 1, 2, 3, 4$, используя функции формы (3.221) для прямоугольной геометрии, заданной четырьмя точками:

$$x_1 = 0, \quad x_2 = h, \quad x_3 = h, \quad x_4 = 0,$$

$$y_1 = 0, \quad y_2 = 0, \quad y_3 = h, \quad y_4 = h.$$

Визуализируйте построенный базис, используя функцию **ezmesh**.

3.2.3. Метод Галеркина

В любой методике построения дискретных моделей дифференциальных задач центральное место занимает способ минимизации невязки (локально или в среднем по области $\overline{\Omega}$). Один из наиболее продуктивных способов минимизации невязки, эффективно использующий слабую формулировку задачи, был предложен Б.Галеркиным в 1915 году и сейчас известен как **метод Галеркина**. Данный метод играет ключевую роль в построении конечно-элементных дискретных моделей.

Напомним, что приближенное решение дифференциальной задачи представляется в виде линейной комбинации базисных функций $\psi_k(x, y)$ с коэффициентами u_k:

$$u(x,y) \simeq U(x,y) = \sum_{k=1}^{N} u_k \psi_k(x,y). \tag{3.222}$$

Подстановка $U(x,y)$ в уравнение для слабой формулировки задачи (3.201), используя в качестве базисных функций $\psi_m(x,y)$, приводит к следующему уравнению:

$$\iint\limits_{\Omega} \lambda(x,y) \sum_{k=1}^{N} u_k \left[\frac{\partial \psi_k}{\partial x} \frac{\partial \psi_m}{\partial x} + \frac{\partial \psi_k}{\partial y} \frac{\partial \psi_m}{\partial y} \right] dx\, dy + \iint\limits_{\Omega} f(x,y)\psi_m \, dx\, dy = \iint\limits_{\Omega} R(x,y)\psi_m \, dx\, dy, \tag{3.223}$$

где $k = 1, 2, \ldots, N$, $R(x,y)$ — невязка, возникающая в силу использования конечного базиса в представлении приближенного решения (3.222). Идея **метода Галеркина** состоит в поиске приближенного решения в виде (3.222), для которого *невязка* $R(x,y)$ *ортогональна всем базисным функциям* ψ_k:

$$\iint\limits_{\Omega} R(x,y)\psi_k \, dxdy = 0, \quad k = 1, 2, 3. \tag{3.224}$$

Из последнего равенства и уравнения (3.222) мы получаем систему уравнений для однозначного определения коэффициентов u_k для каждого конечного элемента:

$$M\mathbf{u} = \mathbf{b}, \tag{3.225}$$

где M — матрица $N \times N$, $\mathbf{b} = [b_1, b_2, \ldots b_N]^T$, $\mathbf{u}, = [u_1, u_2, \ldots u_N]^T$

$$M_{km} = M_{mk} = \iint\limits_{\overline{\Omega}} \lambda(x,y) \left[\frac{\partial \psi_k}{\partial x} \frac{\partial \psi_m}{\partial x} + \frac{\partial \psi_k}{\partial y} \frac{\partial \psi_m}{\partial y} \right] dxdy, \tag{3.226}$$

$$b_k = -\iint\limits_{\overline{\Omega}} f(x,y)\psi_k \, dxdy. \tag{3.227}$$

Матрицу M в уравнении (3.225) называют **матрицей жесткости**. Интегралы в (3.226), (3.227) в общем случае могут быть вычислены численно или частично аналитически (если возможно) для каждого конечного элемента Ω_n в отдельности. Например,

$$\iint\limits_{\Omega_n} \lambda(x,y) \left[\frac{\partial \psi_k}{\partial x} \frac{\partial \psi_m}{\partial x} + \frac{\partial \psi_k}{\partial y} \frac{\partial \psi_m}{\partial y} \right] dxdy \simeq \overline{\lambda} \iint\limits_{\Omega_n} \left[\frac{\partial \psi_k}{\partial x} \frac{\partial \psi_m}{\partial x} + \frac{\partial \psi_k}{\partial y} \frac{\partial \psi_m}{\partial y} \right] dxdy, \tag{3.228}$$

$$\iint\limits_{\Omega_n} f(x,y)\psi_k \, dxdy \simeq \overline{f} \iint\limits_{\Omega_n} \psi_k \, dxdy, \tag{3.229}$$

где $\cup\Omega_n = \Omega$, $\overline{\lambda}$ и $\overline{f}$ — средние значения соотвествующих коэффициентов на заданном объеме (площади) конечного элемента. Интегралы в (3.228)–(3.229) легко вычисляются для полиномиальных функций ψ_m. С другой стороны, мы можем представить любую функцию $f(x,y)$, $(x,y) \in \Omega_n$ приближенно в виде

$$f(x,y) \simeq \sum_{n=1}^{N} f(x_n, y_n)\psi_n(x,y). \tag{3.230}$$

В этом случае мы получаем для вычисления компонент матрицы жесткости и вектора правой части следующие формулы:

$$M_{km} = M_{mk} = \left[\frac{\partial\psi_k}{\partial x}\frac{\partial\psi_m}{\partial x} + \frac{\partial\psi_k}{\partial y}\frac{\partial\psi_m}{\partial y}\right]\sum_{n=1}^{3}\lambda(x_n, y_n)\iint\limits_{\Omega}\psi_n(x,y)dxdy, \tag{3.231}$$

$$b_k = -\sum_{n=1}^{3} f(x_n, y_n)\iint\limits_{\Omega}\psi_n(x,y)\psi_k(x,y)dxdy. \tag{3.232}$$

Вычисление интегралов в расчетах матрицы жесткости и вектора правой части можно упростить при использовании локальной системы координат. В частности, наиболее удобно для этих целей использовать **барицентрическую систему координат**, которая определяется вершинами треугольника, $P_1(x_1, y_2)$, $P_2(x_2, y_2)$, $P_3(x_3, y_3)$. Барицентрические координаты λ_1, λ_2, λ_3 произвольной точки $P = P(x, y)$ внутри треугольника $P_1P_2P_3$ определяются следующим образом:

$$\lambda_1(P) = \frac{S_{P_2P_3P}}{S_{P_1P_2P_3}}, \quad \lambda_2(P) = \frac{S_{P_1P_3P}}{S_{P_1P_2P_3}}, \quad \lambda_3(P) = \frac{S_{P_1P_2P}}{S_{P_1P_2P_3}}, \tag{3.233}$$

где $S_{P_jP_kP}$ — площадь треугольника P_jP_kP.

Примечательно, что барицентрические координаты (3.233) любой точки $P(x, y)$ совпадают со значениями соотвествующей функции формы (3.209)–(3.211) в этой точке:

$$\lambda_1(P) = \varphi_1(x,y), \quad \lambda_2(P) = \varphi_2(x,y), \quad \lambda_3(P) = \varphi_3(x,y). \tag{3.234}$$

Для барицентрических координат любой внутренней точки P треугольника имеем:

$$\lambda_m(P_k) = \delta_{mk}, \tag{3.235}$$

$$\lambda_1(P) + \lambda_2(P) + \lambda_3(P) = 1. \tag{3.236}$$

Последнее равенство означает, что три барицентрических координаты не являются независимыми, и можно ограничиться двумя координатами для однозначного определения любой точки треугольника. Преобразование барицентрических координат в декартовы можно представить в виде:

$$\begin{aligned} x &= \lambda_1x_1 + \lambda_2x_2 + (1 - \lambda_1 - \lambda_2)x_3, \\ y &= \lambda_1y_1 + \lambda_2y_2 + (1 - \lambda_1 - \lambda_2)y_3. \end{aligned} \tag{3.237}$$

Одним из важнейших свойств барицентрических координат является их инвариантность относительно аффинных преобразований. Пусть F — аффинное отображение $F(P') = P$, тогда любой контрольный треугольник Ω'_n с вершинами P'_1, P'_2, P'_3 может быть однозначно отображен на произвольный треугольник Ω с вершинами P_1, P_2, P_3 и обратно. Из инвариантности барицентрических координат по отношению к аффинным преобразованиям следует

$$\begin{aligned} x &= F_1(x', y') = \psi_1(x', y')x_1 + \psi_2(x', y')x_2 + \psi_3(x', y')x_3, \\ y &= F_2(x', y') = \psi_1(x', y')y_1 + \psi_2(x', y')y_2 + \psi_3(x', y')y_3, \end{aligned} \tag{3.238}$$

где $\psi_k = \psi_k(x', y')$, $k = 1, 2, 3$ — барицентрические координаты контрольного треугольника. Для вычисления матрицы жесткости (3.226) удобно использовать некоторый контрольный треугольник Ω', например, представленный на рис. 3.15, где $x'_1 = 0$, $y'_1 = 0$, $x'_2 = 1$, $y'_2 = 0$, $x'_3 = 0$, $y'_3 = 1$, — декартовы координаты вершин, барицентрические координаты которых определяются соотношениями (3.216) при $h \equiv 1$. Пример использования барицентрических координат для расчета матрицы жесткости будет рассмотрен ниже.

Имея результаты интегрирования для некоторого контрольного треугольного Ω' легко получить значения данных интегралов для произвольного треугольника Ω используя следующие соотношения

$$\iint\limits_{\Omega} \psi_k(x, y)\, dxdy = \iint\limits_{\Omega'} \psi_k(x', y')|\det(J_F)|\, dx'dy', \tag{3.239}$$

$$\iint\limits_{\Omega} \psi_k(x, y)\psi_n(x, y)\, dxdy = \iint\limits_{\Omega'} \psi_k(x', y')\psi_n(x', y')|\det(J_F)|\, dx'dy', \tag{3.240}$$

где

$$J_F = \begin{bmatrix} \dfrac{\partial F_1}{\partial x'} & \dfrac{\partial F_1}{\partial y'} \\ \dfrac{\partial F_2}{\partial x'} & \dfrac{\partial F_2}{\partial y'} \end{bmatrix}. \tag{3.241}$$

Несложно показать, что определитель матрицы Якоби J_F не зависит от x', и y'. Кроме того, $|\det J_F| = 2S_\Omega$, где S_{Ω_n} — площадь треугольника Ω. Для случая контрольного треугольника Ω' и линейных функций формы

$$\iint\limits_{\Omega'} \psi_k^\alpha \psi_n^\beta\, d\Omega' = \frac{\alpha!\beta!}{(\alpha + \beta + 2)!} \tag{3.242}$$

Пример 3.12. Рассмотрим метод Галеркина применительно к двумерной задаче (3.198)–(3.199) для прямоугольной области с однородной треугольной сеткой. В этом случае дискретизация области представлена одинаковыми треугольниками, аналогичными, как на рис. 3.14. В силу этого достаточно вычислить матрицу жесткости для одного треугольника и использовать ее для каждого элемента $\Omega_n \in \Omega$. Для этих целей может быть использована локальная система координат с началом в точке P_1. Подстановка (3.216) в (3.226) с учетом (3.228) приводит к следующему результату:

$$M_{11} = \overline{\lambda} \iint\limits_{\Omega_n} \frac{2}{h^2} dxdy = \overline{\lambda} \int_0^h \int_0^{h-x} \frac{2}{h^2} dydx = \overline{\lambda}, \tag{3.243}$$

$$M_{22} = M_{33} = \overline{\lambda} \iint\limits_{\Omega_n} \frac{1}{h^2} dxdy = \overline{\lambda}\frac{1}{2}, \tag{3.244}$$

$$M_{12} = M_{21} = M_{31} = M_{13} = -\overline{\lambda} \iint\limits_{\Omega_n} \frac{1}{h^2} dxdy = -\overline{\lambda}\frac{1}{2}, \tag{3.245}$$

$$M_{32} = M_{23} = 0. \tag{3.246}$$

Для вычисления компонент вектора правой части мы имеем:

$$\iint\limits_{\Omega_n} \psi_1 \, dxdy = \int_0^h \int_0^{h-x} 1 - \frac{x}{h} - \frac{y}{h} dydx = \frac{h^2}{6}, \tag{3.247}$$

$$\iint\limits_{\Omega_n} \psi_2 \, dxdy = \iint\limits_{\Omega_n} \psi_3 \, dxdy = \frac{h^2}{6}. \tag{3.248}$$

Упражнение 3.9. 1. Рассчитать матрицу жесткости для билинейного конечного элемента с функциями формы (3.221) в треугольной области с точками:

$$x_1 = 0, \quad x_2 = h, \quad x_3 = 0, \quad x_4 = h/4,$$

$$y_1 = 0, \quad y_2 = 0, \quad y_3 = h, \quad y_4 = h/2.$$

Убедитесь, что полученная матрица вырожденная.

2. Рассчитать матрицу жесткости для билинейного конечного элемента с функциями формы (3.221) в прямоугольнике с вершинами:

$$x_1 = 0, \quad x_2 = h_x, \quad x_3 = h_x, \quad x_4 = 0, \tag{3.249}$$

$$y_1 = 0, \quad y_2 = 0, \quad y_3 = h_y, \quad y_4 = h_y. \tag{3.250}$$

Убедитесь, что полученная матрица вырожденная.

3.2.4. Смешанные производные и граничные условия

Есть два класса стандартных граничных условий — краевые условия Дирихле и Неймана. В случае **краевого условия Дирихле** на границе области (или на некотором участке границы $\partial\Omega_1 \subset \partial\Omega$) задается явно значение искомого решения:

$$u(x, y) = u_g(x, y), \quad (x, y) \in \partial\Omega_1 \tag{3.251}$$

В случае **краевых условий Неймана** на границе области (или ее части $\partial\Omega_2 \subset \partial\Omega$) определяется градиент искомого решения в виде:

$$\lambda \mathbf{n} \cdot \nabla u(x, y) = -qu + \omega, \quad (x, y) \in \partial\Omega_2, \tag{3.252}$$

где $\mathbf{n}$ — единичный вектор внешней нормали к границе области $\partial\Omega$.

Простейший пример граничных условий Дирихле $u_g \equiv 0$ был рассмотрен выше. Рассмотрим теперь краевую задачу общего вида для уравнения

$$\frac{\partial}{\partial x}\lambda_{11}\frac{\partial u}{\partial x} + \frac{\partial}{\partial y}\lambda_{22}\frac{\partial u}{\partial y} + \frac{\partial}{\partial x}\lambda_{12}\frac{\partial u}{\partial y} + \frac{\partial}{\partial y}\lambda_{21}\frac{\partial u}{\partial x} = f, \quad (x, y) \in \Omega. \tag{3.253}$$

Здесь $f = f(x,y)$, $\lambda_{ij} = \lambda_{ij}(x,y)$, $i,j = 1,2$, $\lambda_{12} = \lambda_{21}$. Для некоторой части границы $\partial\Omega_1$ мы полагаем заданными краевые условия Дирихле (3.251), а для другой части границы $\partial\Omega_2$ — условия Неймана (3.252), $\partial\Omega_1 \cup \partial\Omega_2 = \partial\Omega$.

Слабая формулировка краевой задачи получается при умножении уравнения (3.253) на пробную функцию $\psi = \psi(x,y)$ и интегрировании полученного равенства по области Ω:

$$\iint\limits_{\Omega} \left[\frac{\partial}{\partial x}\lambda_{11}\frac{\partial u}{\partial x}\psi + \frac{\partial}{\partial y}\lambda_{22}\frac{\partial u}{\partial y}\psi + \frac{\partial}{\partial x}\lambda_{12}\frac{\partial u}{\partial y}\psi + \frac{\partial}{\partial y}\lambda_{21}\frac{\partial u}{\partial x}\psi \right] dxdy = \iint\limits_{\Omega} f\psi\, dxdy, \tag{3.254}$$

Используя формулы интегрирования по частям и формулу Остроградского – Гаусса для преобразования интеграла по площади в контурный интеграл по границе, имеем:

$$\oint\limits_{\partial\Omega} \mathbf{n}\cdot\lambda\nabla u\psi d\gamma - \iint\limits_{\Omega} \lambda_{11}\frac{\partial u}{\partial x}\frac{\partial\psi}{\partial x} + \lambda_{22}\frac{\partial u}{\partial y}\frac{\partial\psi}{\partial y} + \lambda_{12}\frac{\partial u}{\partial y}\frac{\partial\psi}{\partial x} + \lambda_{21}\frac{\partial u}{\partial x}\frac{\partial\psi}{\partial y} dxdy = \iint\limits_{\Omega} f\psi\, dxdy. \tag{3.255}$$

Если положить, что пробная функция имеет нулевое значение на границе $\partial\Omega_1$ с учетом краевых условий контурный интеграл в (3.255) принимает вид

$$\oint\limits_{\partial\Omega} \mathbf{n}\cdot\lambda\nabla u\psi d\gamma = q\oint\limits_{\partial\Omega_2} u\psi d\gamma + \oint\limits_{\partial\Omega_2} \omega\psi d\gamma. \tag{3.256}$$

Таким образом, мы видим, что ненулевые условия Неймана приводят к дополнительному интегралу в слабой формулировке задачи. Используя метод Галеркина мы можем естественным образом учесть краевые условия произвольного вида. Для вычисления контурного интеграла полезно использовать следующее соотношение:

$$\int\limits_{P_k}^{P_m} \psi_k^{\alpha}\psi_n^{\beta}\, ds = \frac{\alpha!\beta!}{(\alpha+\beta+1)!}|P_m, P_k|, \quad n = k, m, \tag{3.257}$$

$$\int\limits_{P_k}^{P_m} \psi_k^{\alpha}\psi_n^{\beta}\, ds = 0, \quad n \neq k,\ n \neq m, \tag{3.258}$$

где $|P_m, P_k|$ — расстояние между вершинами треугольника P_m и P_k. Принимая во внимание (3.257) для случая линейных функций формы мы имеем:

$$\int\limits_{P_k}^{P_m} \psi_k\psi_n\, ds = \frac{1}{6}|P_k, P_m|, \quad \int\limits_{P_k}^{P_m} \psi_k\psi_k\, ds = \frac{1}{3}|P_k, P_m|. \tag{3.259}$$

Пример 3.13. Рассмотрим краевую задачу (3.253), (3.251), (3.252) в треугольной области Ω, представленной на рис. 3.15, где $P_1(0.5, 1.0)$, $P_2(0.0, 0.0)$, $P_3(1.0, 0.0)$.

Для построения конечно-элементной модели вида (3.225) нам требуется рассчитать матрицу жесткости M и вектор правой части b. Для упрощения задачи

мы воспользуемся вычислениями для контрольного треугольника Ω': $P_1'(0.0, 0.0)$, $P_2(1.0, 0.0)$, $P_3(0.0, 1.0)$ (см. рис. 3.15). После этого необходимые вычисления могут быть перенесены на любой треугольник Ω, используя соотношения (3.239)–(3.240).

Для контрольного треугольника Ω' и функций формы $\psi_1 = 1 - x' - y'$, $\psi_2 = x'$, $\psi_3 = y'$, имеем:

$$\{M\}_{11} = -\sum_{k,m=1,2} \frac{\partial\psi_1}{\partial x_k}\frac{\partial\psi_1}{\partial x_m}\sum_{n=1}^{3}\lambda_{km}(P_n')\iint_{\Omega'}\psi_n\psi_1 dy'dx' - q\oint_{\partial\Omega_2}\psi_1^2 ds = \tag{3.260}$$

$$-\frac{2S_{P_1'P_2'P_3'}}{6}\sum_{n=1,m=1,2}[2\lambda_{km}(P_1') + \lambda_{km}(P_2') + \lambda_{km}(P_3')] - \frac{q}{3}(|P_1', P_2'| + |P_1', P_3'|),$$

$$\{M\}_{12} = -\sum_{k,m=1,2} \frac{\partial\psi_1}{\partial x_k}\frac{\partial\psi_2}{\partial x_m}\sum_{n=1}^{3}\lambda_{km}(P_n')\iint_{\Omega'}\psi_n\psi_2 dy'dx' - q\oint_{\partial\Omega_2}\psi_1\psi_2 ds = \tag{3.261}$$

$$\frac{2S_{P_1'P_2'P_3'}}{6}\sum_{k=1,2}[\lambda_{k1}(P_1') + 2\lambda_{k1}(P_2') + \lambda_{k1}(P_3')] - \frac{q}{6}(|P_1', P_2'| + |P_1', P_3'|),$$

$$\{M\}_{13} = -\sum_{k,m=1,2} \frac{\partial\psi_1}{\partial x_k}\frac{\partial\psi_3}{\partial x_m}\sum_{n=1}^{3}\lambda_{km}(P_n')\iint_{\Omega'}\psi_n\psi_3 dy'dx' - q\oint_{\partial\Omega_2}\psi_1\psi_3 ds = \tag{3.262}$$

$$\frac{2S_{P_1'P_2'P_3'}}{6}\sum_{k=1,2}[\lambda_{k2}(P_1') + \lambda_{k2}(P_2') + 2\lambda_{k2}(P_3')] - \frac{q}{6}(|P_1', P_2'| + |P_1', P_3'|),$$

здесь $x_1 = x$, $x_2 = y$, $S_{P_1'P_2'P_3'} = 1/2$ — площадь треугольника (P_1', P_2', P_3'). Остальные коэффициенты определяются краевыми условиями (3.262):

$$\{M\}_{22} = 1, \quad \{M\}_{21} = \{M\}_{23} = 0, \tag{3.263}$$

$$\{M\}_{33} = 1, \quad \{M\}_{31} = \{M\}_{32} = 0. \tag{3.264}$$

Для вектора правой части $\mathbf{b}$ имеем:

$$b_2 = u_g(P_2'), \quad b_3 = u_g(P_3'), \tag{3.265}$$

$$b_1 = \sum_{n=1}^{3} f(P_n')\iint_{\Omega'}\psi_n\psi_1 dy'dx' - \sum_{n=1}^{3}\omega(P_n')\oint_{\partial\Omega_2}\psi_1\psi_n ds = \tag{3.266}$$

$$\frac{2S_{P_1'P_2'P_3'}}{6}[2f(P_1') + f(P_2') + f(P_3')] - \frac{|P_1', P_2'| + |P_1', P_3'|}{6}[2\omega(P_1') + \omega(P_2') + \omega(P_3')].$$

Для применения полученных результатов (3.260)–(3.266) к конечному элементу на произвольном треугольнике Ω нам следует использовать P_1, P_2, P_3 вместо P_1', P_2', P_3', и учесть фактические значения градиентов $\dfrac{\partial\psi_k}{\partial x}$ $\dfrac{\partial\psi_m}{\partial y}$, $k, m = 1, 2, 3$ согласно (3.214)–(3.215).

Упражнение 3.10. 1. Рассчитать матрицу жесткости билинейного элемента с функциями формы (3.221) в прямоугольной области с вершинами (3.249), (3.250) и нулевыми краевыми условиями в точке (x_1, y_1). Убедитесь, что полученная матрица жесткости не вырожденная.

2. Построить слабую формулировку задачи и матрицу жесткости для уравнения (3.254), используя линейные функции формы для треугольных областей, представленных на рис. 3.15 с нулевыми краевыми условия Дирихле в точках P_1, и P_1' соответственно.

3.2.5. Дискретная конечно-элементная модель. Сборка

После того как матрицы жесткости и векторы правых частей вычислены для каждого элемента в отдельности, следующим шагом является сборка дискретной модели. Данная модель представляет собой систему алгебраических уравнений, размерность которой совпадает с числом точек сетки.

Для простоты будем полагать, что все конечные элементы линейные и имеют однородную геометрию. Пример такой однородной триангуляции для случая прямоугольной области представлен на рис. 3.16.

Вначале пронумеруем внутренние узлы сетки и внутренние треугольники. Для определенности используем нумерацию по строкам. Граничные элементы можно пронумеровать отдельно (см. рис. 3.16).

Процедура сборки не столь тривиальна, как может показаться на первый взгляд. Это связано с тем, что каждый узел сетки, за исключением угловых граничных точек, принадлежит одновременно нескольким соседним элементам. Например, каждый внутренний узел принадлежит одновременно шести элементам и вклад каждого должен быть адекватно учтен в итоговой дискретной модели.

Один из возможных **алгоритмов сборки** может быть представлен в виде двух стадий. На первом этапе мы собираем вместе все локальные матрицы жесткости в блок-диагональную черновую матрицу жесткости, согласно порядку нумерации элементов сетки. Размерность черновой матрицы жесткости определяется общим числом элементов и размерностью локальных матриц жесткости. Для нулевых краевых условий размерность локальной матрицы жесткости может быть понижена с учетом нулевого вклада граничных узлов.

Следующий этап состоит в объединении строк черновой матрицы, которые связаны с одним и тем же узлом сетки. Рассмотри сборку глобальной матрицы для случая квадратной области из двух треугольных элементов (см. рис. 3.17). Вначале мы вычисляем **локальные матрицы жесткости** для каждого элемента в отдельности. Для простоты полагаем $\lambda \equiv 1$:

$$S^{(1)} = \frac{1}{2}\begin{bmatrix} 2 & -1 & -1 \\ -1 & 1 & 0 \\ -1 & 0 & 1 \end{bmatrix}, \quad S^{(2)} = \frac{1}{2}\begin{bmatrix} 2 & -1 & -1 \\ -1 & 1 & 0 \\ -1 & 0 & 1 \end{bmatrix} \tag{3.267}$$

Далее, собираем черновую **глобальную матрицу жесткости** в виде:

$$M_d = \begin{bmatrix} S^{(1)} & 0 \\ 0 & S^{(2)} \end{bmatrix}. \tag{3.268}$$

Окончательно, нам необходимо преобразовать черновую матрицу жесткости 6×6 в глобальную матрицу жесткости 4×4, путем объединения общих узлов смежных треугольников. Данная процедура может быть формализована с использованием соответствующей матрицы преобразования, которая строится следующим образом. За основу берется нулевая матрица размера (4×6) и в каждой строке задается один единичный элемент $s_{km} = 1$. Здесь индексы k и m определяются в соответствии с порядковым номером k-й строки черновой матрицы и соответствующим ей m-м номером точки сетки в глобальной нумерации узлов. В рассмотренном случае сетки, представленной на рис. 3.17, матрицы преобразования сборки будет иметь следующий вид.

$$Q = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}. \tag{3.269}$$

Таким образом, для формирования глобальной матрицы жесткости мы имеем

$$S = Q^T M_d Q = \begin{bmatrix} 2 & -1 & -1 & 0 \\ -1 & 2 & 0 & -1 \\ -1 & 0 & 2 & -1 \\ 0 & -1 & -1 & 2 \end{bmatrix}. \tag{3.270}$$

Граничные элементы могут быть включены в процесс сборки в отдельности, либо, в случае нулевых условий Дирихле, исключены из рассмотрения, в силу нулевого вклада граничных точек в глобальную модель.

Пример 3.14. Рассмотрим простой пример реализации метода конечных элементом для решения двумерного уравнения Пуассона в прямоугольной области с нулевыми условиями Дирихле на границе:

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = 1, \quad 0 < x < 1, \quad 0 < y < 1, \tag{3.271}$$

$$u(0, y) = u(1, y) = u(x, 0) = u(x, 1) = 0. \tag{3.272}$$

```
clear
N = 32;
h = 1/(N+1); x = h:h:1-h;
%% матрица жесткости внутренних элементов   %%
A0 =  0.5*[2,-1,-1;-1, 1, 0;-1, 0, 1];
%% Матрицы жесткости граничных элементов  %%
A12 = 0.5*[2,-1, 0;-1, 1, 0; 0, 0, 1];
A13 = 0.5*[2, 0,-1; 0, 1, 0;-1, 0, 1];
A00 = 0.5*[2, 0, 0; 0, 1, 0; 0, 0, 1];
Nt = 2*(N-1)^2; % Число внутренних элементов
Nb = (N-1)*8+6; %   Число граничных элементов
```

```
NN=(N+1)^2;     %   Общее число точек сетки
 %% массив координат вершин  треугольников %%%%%%
t = zeros(Nt,3); %   внутренних
tb = zeros(Nb,3);%   граничных


%% Построение массивов, содержащих индексы
%% вершин треугольников
%% Индексы вершин внутренних треугольников
n=1;
for m=1:N-1
for s=1:N-1
k = s + (m-1)*N;
t(n,:) = [k,  k+1, k+N];
n =  n+1;
t(n,:) = [k+N+1,k+N, k+1];
n = n+1;
end
end
%%           Черновая матрица жесткости       %%%
A = kron(speye(Nt),A0);
%% Рассматриваются только внутренние элементы  %%%%%
%% Граничные элементы будут рассмотрены ниже %%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Граница I. y=0   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
n = 1;
N1 = N^2+4*(N+1);
tb(n,:) = [1, N1, N^2+2];
A = blkdiag(A,A00);
n = n+1;
for m = 1:N-1
k = N^2+m+1;
tb(n,:) = [m+1, m, k];
n = n+1;
tb(n,:) = [k-1, k, m];
A = blkdiag(A,A12,A00);
n = n+1;
end
N1 = N^2+N+1;
tb(n,:) = [N1,  N1+1, N];
n = n+1;
tb(n,:) = [N1+2,  N, N1+1];
A = blkdiag(A,A00,A00);
n = n+1;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Граница II. x=L  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
N1 = N^2+N+2;
for m = 1:N-1
k = N*m;
tb(n,:) = [k,  N1+m, k+N];
n = n+1;
tb(n,:) = [N1+m+1, k+N, N1+m];
A = blkdiag(A,A13,A00);
n = n+1;
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Граница III. y=L  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
N1 = N^2+2*N+2;
tb(n,:) = [N^2, N1, N1+2];
A = blkdiag(A,A00);
n=n+1
for m = 1:N-1
N1 = N^2+2*(N+2);
tb(n,:) = [N^2-m, N^2-m+1, N1+m];
n = n+1;
tb(n,:) = [N1+m-1, N1+m, N^2-m+1];
A = blkdiag(A,A12,A00);
n = n+1;
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Граница IV. x=0  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
N1 = N^2+3*(N+1);
tb(n,:) = [N1, N1+1, N*(N-1)+1];
n = n+1;
tb(n,:) = [N1+2, N*(N-1)+1, N1+1];
A = blkdiag(A,A00,A00);
n = n+1;
N1 = N^2+3*(N+1)+1;
for m = 1:N-1
tb(n,:) = [N*(N-m)+1, N1+m, N*(N-m-1)+1];
n = n+1;
tb(n,:) = [ N1+m+1, N*(N-m-1)+1, N1+m];
A = blkdiag(A,A13,A00);
n = n+1;
end
T = [t;tb]';
%% Построение матрицы преобразования сборки %%%%%%
P = sparse([],[],[],3*(Nt+Nb),NN,3*(Nt+Nb));
for m = 1:3*(Nt+Nb)
P(m,T(m)) = 1;
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%% Сборка матрицы жесткости     %%%%%%%%%%%
S = P'*A*P;
W = S;
S = S(1:N^2,1:N^2); % Удаление граничной части
f = h^2*ones(N,N);
u = S\f(:);
u = reshape(u,N,N); [X,Y] = meshgrid(x,x);
if N < 10
figure('Position',[403 246 860 410])
subplot(1,2,1), surfc(X,Y,u);
else
subplot(1,2,2), surfc(X,Y,u)
end
xlabel('x'); ylabel('y'); axis([0,1,0,1,0,0.075]);
title(['Grid Size ',num2str(N),'x',num2str(N)]);
```

Упражнение 3.11. 1. Используя функцию Matlab **det** убедитесь, что определитель матрицы жесткости S, заданной уравнением (3.270), равен нулю. Объясните почему?

2. Рассчитать матрицу жесткости для билинейного конечного элемента с функциями формы (3.221) в прямоугольной области с вершинами P_1, P_2, P_3 и P_4 (см. рис. 3.17). Сравните полученную матрицу с матрицей S, заданной уравнением (3.270).

3. Построить конечно-элементную модель задачи, рассмотренной в примере выше, используя билинейные функции формы (3.221) на прямоугольной сетке.

3.2.6. Примеры программного обеспечения для метода конечных элементов.

Метод конечных элементов является одним из наиболее эффективных инструментов численного анализа уравнений с частными производными. В то же время, данный метод является достаточно сложным и требует серьезной подготовки для реализации его возможностей применительно к тому многообразию дифференциальных задач, которые возникают в современных инженерно – физических приложениях. В связи с этим в инженерной практике разумно использовать профессионально разработанные программные средства, позволяющие упростить и сделать доступной данную технику для широкого круга пользователей, не имеющих опыта разработки подобных приложений, но хорошо представляющих суть решаемых задач.

Среди программных продуктов, использующих метод конечных элементов, можно найти десятки интересных вариантов, начиная от простейших библиотек с отрытым кодом (Elmer, FreeFem++, OOFEM, Code Aster, OpenFOAM, Z88Aurora и т.п.) до продвинутых коммерческих проектов (COMSOL, ANSYS, SOLIDWORKS, Range Software и др.).

В качестве первого опыта работы с методом конечных элементов можно рекомендовать одно из наиболее простых, широко доступных и одновременно весьма

эффективных приложений MATLAB Partial Differential Equations Toolbox^{TM} (PDE app). Данное приложение включает набор функций, рассчитанных на решения основных типов дифференциальных уравнений (эллиптических, параболических, гиперболических и проблему собственных функций, собственных значений) в двумерной постановке для области произвольной формы. Благодаря наличию весьма адекватного графического интерфейса работа с приложением достаточно удобна и доступна широкому кругу пользователей, причем глубокие знания и опыт программирования в среде MATLAB не являются обязательными условиями успешного использования приложения.

Рассмотрим порядок использования приложения на примере задачи о нагреве силового электрического кабеля при повышенной токовой нагрузке. Пусть двужильный кабель с радиусом медных проводников $R_c = 1$ мм и расстоянием между ними 5 мм имеет полихлорвиниловую изоляцию толщиной $d = 2$ мм. Нас интересует распределение температурных полей в поперечном сечении кабеля, которое удовлетворяет стационарному уравнению теплопроводности:

$$-\mathrm{div}\left(k \cdot \mathrm{grad}(T)\right) = Q, \tag{3.273}$$

где k — коэффициент теплопроводности, Q — плотность мощности тепловых источников,

$$Q = \frac{I^2 \rho}{(\pi R_c^2)^2}. \tag{3.274}$$

Здесь I — сила тока, ρ — удельное сопротивление проводника, которое, вообще говоря, зависит от температуры,

$$\rho = \rho(T) = \rho_0 \left(1 + \alpha_\rho (T - T_0)\right). \tag{3.275}$$

Полагаем, что кабель окружен воздушной средой и условие теплообмена на внешней поверхности кабеля определяются уравнением

$$\mathbf{n} \cdot c \cdot \mathrm{grad}(T) = \alpha(T)\left(T - T_0\right), \tag{3.276}$$

где $\mathbf{n}$ — единичный вектор внешней нормали к поверхности кабеля, T_0 — температура окружающей среды, $\alpha(T)$ — нелинейный коэффициент конвективного и радиационного теплообмена с окружающей средой. Типичные значения параметров задачи приведены в таблице ниже.

	c [Вт(м · К)]	ρ_0 [Ом · м]	α_ρ[1/К]
Проводник (медь)	390	1.75e-8	0.004
Изолятор (ПВХ)	1.7	-	-
Окружающая среда (воздух)	0.027	-	-

Таблица 3.1 Материальные параметры кабеля при температуре $T = 300K$.

При умеренном нагреве кабеля теплообмен с окружающей средой можно полагать чисто конвекционным. В этом случае коэффициент теплообмена α имеет вид:

$$\alpha = \frac{c_a \cdot Nu}{D}, \tag{3.277}$$

где c_a — коэффициент теплопроводности воздуха, Nu — число Нуссельта, D — характеристический размер. В нашем случае D — диаметр кабеля с изоляцией. Типичные значения числа Нуссельта для горизонтального цилиндра в воздушном пространстве находится в интервале $Nu = 3 \div 12$. Будем использовать среднее значение коэффициента теплообмена $\alpha = 25$ Вт(м $\cdot$ К).

Приведенные данные достаточны для построения математической модели и численного моделирования температурных полей в поперечном сечении кабеля с применением **PDETool**. Для запуска данного приложения используем команду

```
>>pdetool
```

Управление приложением осуществляется посредством графического интерфейса с использованием соответствующих пунктов меню. Начать удобно с общих установок, предлагаемых в разделе **Options**. Первое, необходимо установить границы осей в соответствие с геометрией задачи с помощью пункта **axes limits**. Например, **X-axis range**: [−0.005, 0.007] , **Y-axis range**: [−0.004, 0.004]. Заметим, что геометрические размеры и другие физические параметры должны быть заданы в единой системе единиц измерения, например СИ (килограмм, метр секунда...). Для построения геометрии модели в разделе **Options** полезно активировать опции **Grid** and **Snap** (to grid), а также в подразделе **Application** установить режим **Heat Transfer** вместо **Generic Scalar**, который используется по умолчанию. Вид графического интерфейса представлен на рис. 3.2.6.

Следующий шаг состоит в построении геометрии модели. Наиболее простой способ состоит в построении области в виде комбинации геометрических примитивов (кругов, эллипсов, прямоугольников или многоугольников), определяя их положения, размеры и ориентацию на координатной плоскости.

Для определения геометрия поперечного сечения кабеля пользуемся пунктами раздела меню **Draw**, переходя в режим **Draw Mode**, или соответствующими иконками с изображениями геометрических примитивов в верхней части интерфейса. Например, для определения кругового сечения проводника активируем соответствующую фигу посредством меню (**Draw** $\Rightarrow$ **Ellipse circle(centerd)**), или нажимая на икону с соответствующим изображением. Затем, помещаем курсор в точку предполагаемого центра проводника и, удерживая нажатой левую кнопку мыши, перемещаем курсор от центра к периферии круга. Завершение рисования происходит когда мы отпускаем кнопку мыши. Заметим, что каждая фигура декомпозиционной геометрии имеет оригинальное имя (по умолчанию C1, C2, ... для кругов, E1, E2, ... для эллипсов и т.д.). Имя, положение, размеры и ориентацию фигуры можно изменять двигая выделенный объект мышкой или уравляя параметрами объектов

посредством диалогового окна, которое появляется при двойном щелчке левой кнопки мыши, когда курсор находится в поле фигуры (в случае наложения фигур, курсор подводится к надписи имени примитива). В нашем случае расчетная область задачи задается четырьмя кругами, определяющими границы проводников и изоляции кабеля, как показано на рис. 3.20.

Окончательно, площади фигур декомпозиционной геометрии могут быть включены или исключены из расчетной области задачи, используя соответствующие знаки *плюс* или *минус* в редактируемом текстовом окне **Set Formula**, расположенного непосредственно под главным меню графического интерфейса приложения.

Для задания граничных условий следует включить режим **Boundary Mode** нажатием на кнопку с символом $\partial\Omega$, либо посредством опций меню: (**Boundary** $\Rightarrow$ **Boundary Mode**).

Граница области сегментирована. Граничные условия могут определяться для всей границы (**Boundary** $\Rightarrow$ **Specify Boundary Conditions...**) или для отдельных участков, путем двойного щелчка левой клавишей мыши по соответствующему сегменту. В открывающемся при этом диалоговом окне мы можем определить тип краевых условий (Дигихле или Неймана) и величины соответствующих параметров.

Коэффициенты в граничных условиях могут быть определены постоянными параметрами или выражениями, описывающими их зависимость от координат и времени. Для рассматриваемой задачи нам необходимо определить краевые условия Неймана в соответствие с уравнением (3.277) и уравнением краевых условий, приводимом в верхней части диалогового окна (см. Рис. 3.21). Внутренние границы между подобластями могут быть удалены с помощью соответствующих опций меню **Boundary**.

Процедура определение коэффициентов уравнения задачи аналогична заданию краевых условий. Для этого переходим в режим PDE, используя опцию **PDE Mode** в пункте меню. Далее, путем двойного щелчка левой клавишей мыши по метке соответствующей подобласти активирует диалоговое окно, посредством которого определяются тип и входные данные уравнения задачи в активной подобласти. В отличие от краевых условий, коэффициенты уравнения могут быть нелинейными и описываться в виде функций, зависящих не только от координат и времени, но и от решения задачи.

В рассмотренном нами случае коэффициенты эллиптического уравнения задаются для каждой подобласти в отдельности согласно Таблице 3.1 и уравнению (3.274). Пример диалогового окна для определения параметров проводника представлен на Рис. 3.22.

Определение коэффициентов уравнения и граничных условий завершает построение математической модели, и далее можно переходить к численным экспериментам. Предварительные результаты могут быть получены нажатием на кнопку с символом $=$. В этом случае задача решается с использованием параметров численной модели, заданных по умолчанию. Дополнительная настройка параметров позволяет повысить эффективность и гарантии корректности результатов моделирования.

При первом пробном эксперименте полезно использовать грубую сетку, которая генерируется посредством опций меню **Mesh** $\Rightarrow$ **Initialize Mesh**, или нажатием на кнопку с символом в виде правильного треугольника. В нашем случае грубая

триангуляция области имеет вид, представленный на Рис. 3.23 (Coarse Mesh). Такая триангуляция используется по умолчанию и обычно не обеспечивает желаемую точность результатов моделирования. Переход к более мелкой сетке осуществляется посредством опций меню **Mesh** ⇒ **Refine Mesh**, или нажатием на кнопку с символом в виде правильного треугольника, разбитого на четыре равные части. Пример улучшенной сетки, с линейными размерами ячейки, уменьшенными в два раза по сравнению с начальными значениями, представлен на Рис. 3.23, (Refinement Mesh). Отметим, что триангуляция области отображается в режиме **Mesh Mode**, который инициируется посредством соответствующей опцией меню **Mesh**, или автоматически при построении и дроблении сетки.

Перерасчет решения на сетке с уменьшенным размером ячеек сетки и сравнение результатов моделирования с использованием последовательности сеток с улучшенным пространственным разрешением является одним из способов убедиться в корректности результатов моделирования и получить представление о величине погрешности решения. Если при однократном дроблении сетки полученные результаты визуально отличаются от результатов на грубой сетке, тогда следует повторять дробление сетки до тех пор, пока различия в результатах на сетках с различным пространственным разрешением станет несущественными. Метод конечных элементов с линейными функциями формы имеет второй порядок аппроксимации и разность решений, полученных на сетках с размерами ячейки h и $h/2$, сравнимы по порядку величины с погрешностью решения на мелкой сетке.

Другой способ выбора подходящего пространственного разрешения дискретной модели состоит в использовании процедуры адаптации сетки, доступной в PDEtool приложении для решения краевых задач эллиптического типа. Для активации данной функции следует воспользоваться соответствующей опцией **Adaptive mesh** в диалоговом окне настройки параметров: **Solve⇒Parameters...**.

При использовании функции адаптации сетки задача решается несколько раз на сетках с различным пространственным разрешением для оценки локальной погрешности и согласования размеров ячеек сетки с требуемой точностью решения задачи. Пример адаптивной сетки, которая отвечает оптимальному размеру ячеек при решении рассмотренной задачи, представлена на Рис. 3.23.

При успешном завершении расчетов полученное решение в виде функции двух переменных отображается графически с помощью цветовой палитры. Используя меню **Plot** ⇒ **Parameters...** мы можем настроить содержание и форму представления результатов моделирования. В частности, доступно изменение цветовой палитры, отображение градиента решения, триангуляции и т.д. (смотри подробнее опции меню **Plot**).

Используя меню **File** ⇒ **Export image** (доступно в поздних версиях MATLAB, начиная с 2013 г.) результаты могут быть сохранены в форме рисунка (доступны ргафические форматы .bmp, .jpg, .eps, .tif, и др.). Пример изображения с результатами моделирования рассмотренной задачи, включающего поле температур (T), и тепловые потоки (векторное поле q) в поперечном сечении кабеля представлен на Рис. 3.24.

Кроме того, любые компоненты модели, такие как решение, сетка, геометрия задачи, коэффициенты и др. могут быть экспортированы в цифровом формате и использованы другими приложениями, в том числе и функциями MATLAB. Например,

изображения сеток, представленные на Рис. 3.23 были получены с использованием данных, экспортированных из приложения PDEtool и обработанных с помощью функции Matlab **pdeplot**.

Построенная математическая модель может быть сохранена в виде стандартного m-файла, доступного для дальнейшей модификации и использования другими пользователями в среде приложения PDEtool. Для сохранения и загрузки модели используется меню приложения: **File** $\Rightarrow$ **Saveas...** и **File** $\Rightarrow$ **Open...**.

В качестве примера дальнейшего усовершенствования модели рассмотрим возможности решения нелинейных задач. Ранее, при построении модели, мы полагали, что материальные параметры являются постоянными, несмотря на то, что в действительности они являются функциями температуры.

Рассмотрим возможность учета зависимости удельного сопротивления проводника от температуры и влияние данного эффекта на результаты моделирования. Вместо линейной зависимости (3.275), используем более общую формулу:

$$\rho(T) = \rho_0 \left(1 + \alpha_\rho (T - T_0) + \beta_\rho (T - T_0)^2\right), \tag{3.278}$$

где $\rho_0 = 1.75e-8$ [Ом$\cdot$ м], $\alpha_\rho = 4e-3$ $[1/K]$, $\beta_\rho = 6e-7$ $[1/K^2]$. Используем усовершенствованную модель для исследования эффекта нагрева проводника током короткого замыкания для силового кабеля длинной 200 м при напряжении $U = 110$ Вольт. Ток короткого замыкания определяется согласно закону Ома

$$I = \frac{\pi R_c^2 U}{\rho L}. \tag{3.279}$$

Подстановка (3.279) в уравнение (3.274) приводит к следующему выражению для плотности тепловых источников:

$$f = \frac{I^2 \rho}{(\pi R_c^2)^2} = \frac{U^2}{\rho L^2} = \frac{U^2}{L^2 \rho_0 \left(1 + \alpha_\rho (T - T_0) + \beta_\rho (T - T_0)^2\right)} \tag{3.280}$$

Наиболее существенное отличие между уравнением теплопроводности (3.273) с функциями правой части (3.274) и (3.280) состоит в том, что в последнем случае мы имеем дело с нелинейной задачей. Для иллюстрации существенности эффекта температурной зависимости удельного сопротивления, определим тепловые источники для одного проводника согласно уравнению (3.280),

```
110^2/200^2./(175e-10*(1+0.004*(u-300)+6e-7*(u-300).^2)),
```

а для другого проводника используем выражение (3.274):

```
110^2/200^2/175e-10
```

В последнем случае мы пренебрегли температурной зависимостью удельного сопротивления проводника. Используя построенную модель удобно проследить влияние температурной зависимости сопротивления проводника на тепловой эффект короткого замыкания.

Для численного решения нелинейной задачи следует активировать соответствующую опцию **Use nonlinear solver** в настройках параметров, используя меню **Solve** $\Rightarrow$ **Parameters...**.

Результаты моделирования, демонстрирующие эффект температурной зависимости удельного сопротивления при нагреве проводника током короткого замыкания, представлены на Рис. 3.25. Легко видеть, что пренебрежение температурной зависимостью сопротивления приводит к существенной переоценке температуры нагрева кабеля.

Рассмотренный пример иллюстрирует основные функциональные возможности приложения MATLAB PDEtool. Наряду с эллиптическими уравнения, данное приложение может быть использовано для решения нестационарных уравнений и систем параболического и гиперболического типов, применительно к задачам строительной механики, диффузии, электро- и магнитостатики, электрических машин постоянного и переменного тока. К сожалению, возможности приложения ограничены случаем двумерной геометрии.

Упражнение 3.12. 1. Усовершенствуйте рассмотренную модель для исследования динамики тепловых полей в поперечном сечении кабеля при коротком замыкании. При определении типа уравнения следует использовать опцию **Parabolic** (см. Рис.3.22):

$$\rho_m C \frac{\partial u}{\partial t} - \operatorname{div}\left(k \cdot \operatorname{grad}(T)\right) = Q, \tag{3.281}$$

где ρ_m и C — плотность [кг/м^3] и теплоемкость [Дж/(кг· К)] соответственно. Параметры нестационарной модели приведены в Таблице 3.2.

Таблица 3.2 Материальные параметры кабеля для нестационарной модели.

	ρ_m [кг/м^3]	C [Дж/(кг· К]
Проводник (медь)	8960	390
Изоляция (ПВХ)	1400	1100

2. Используя нестационарную модель, построенную согласно упражнению 1, найти решение задачи при $t = 400$ сек. и сравнить с решением соответствующей стационарной задачи, рассмотренной в примере выше.

3. Используя нестационарную модель (см. упражнение 1) оценить время, при котором температура проводника достигает уровня 99% от максимальной температуры при $t \to \infty$.

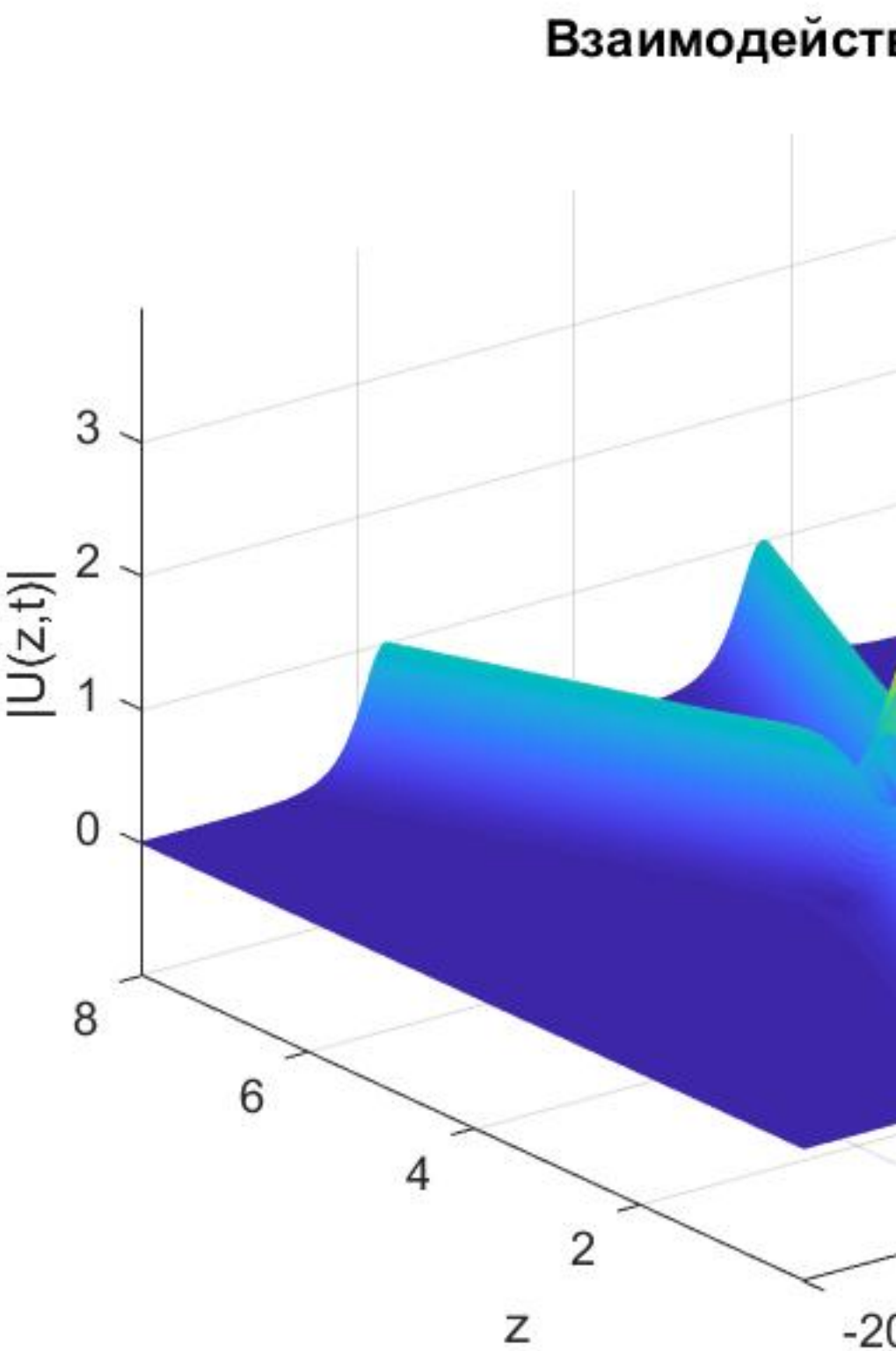
3.3. Дополнительная литература

Детальное изложении основных подходок к численному анализу уравнений в частных производных на основе методов конечных элементов, конечных разностей, конечных объемов, спектральных методов и методов декомпозиции области, можно

найти во многих книгах [45], [47]. Ряд источников содержит также алгоритмические и компьютерные аспекты численных методик и предоставляет примеры программной реализации [54]. Единый подход к построению численных методик для анализа обыкновенных дифференциальных уравнений и уравнений в частных производных представлен в [47]. Специфические проблемы вычислительной гидродинамики обсуждаются в одной из наиболее цитируемых книг [64].

Доступное введение в метод конечных элементов можно найти в [41] – [30]. Обсуждения практических вопросов построения и программной реализации метода конечных элементов в среде MATLAB приведены в [30].

Итерационные методы для больших разреженных линейных систем, возникающих при построении дискретных моделей на основе методов конечных разностей и конечных элементов, обсуждаются в работах [57] – [19]. Современное состояние многосеточных методов можно найти в [36].


Взаимодейст
|U(z,t)|
3
2
1
0
8
6
4
2
z
-20

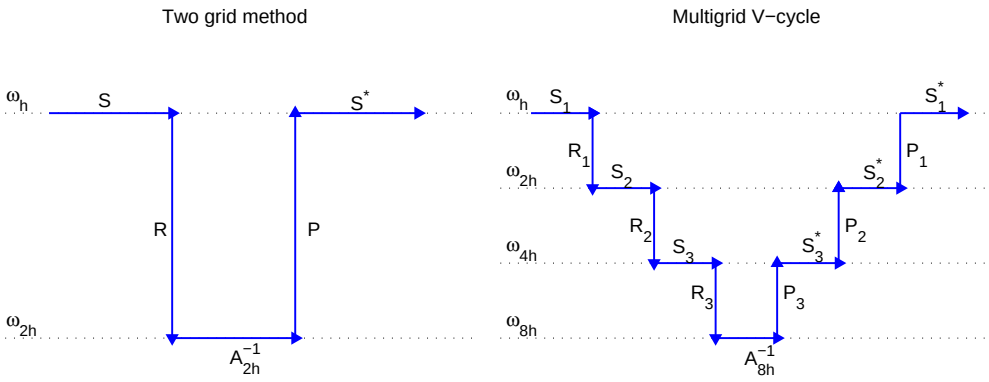


Рис. 3.11. Схема многосеточного метода: S и S* — сглаживания, R — проекция на грубую сетку, A_{2h}^{-1} — вычисление погрешности на грубой сетке, P — коррекция с грубой сетки.

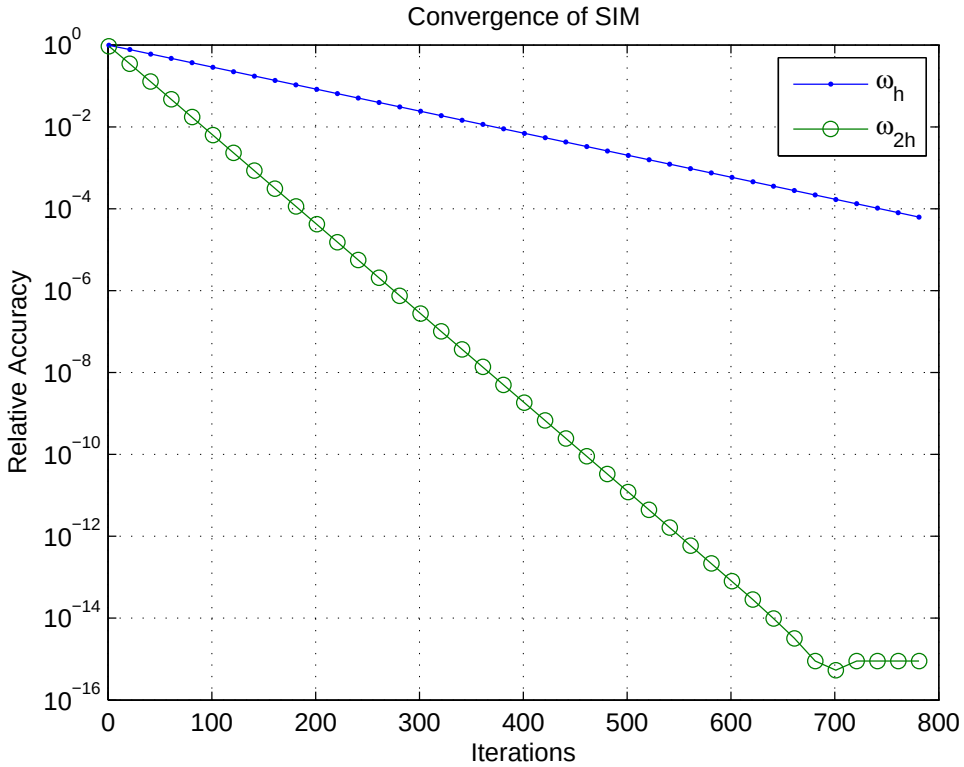


Рис. 3.12. Сходимость МПИ на мелкой ($\$\omega_h$) и грубой ($\omega_{2h}$) сетках .

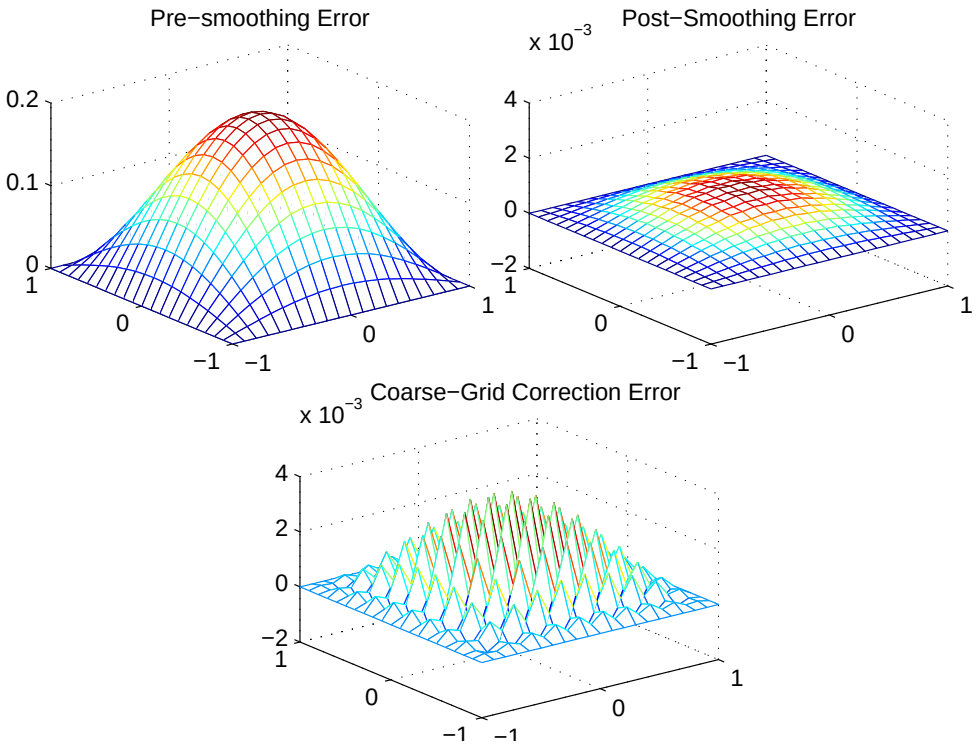


Рис. 3.13. Динамика погрешности в процессе двухсеточного v-цикла.

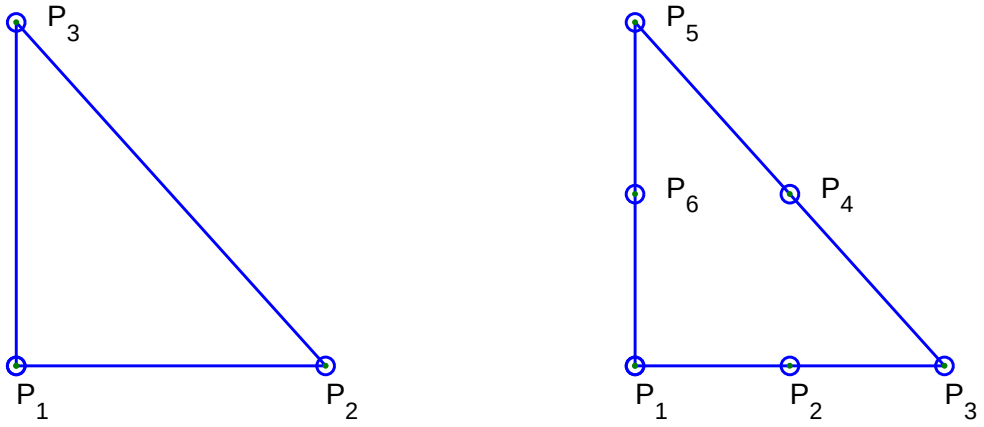


Рис. 3.14. Геометрия линейного (слева) и квадратичного (справа) треугольных элементов

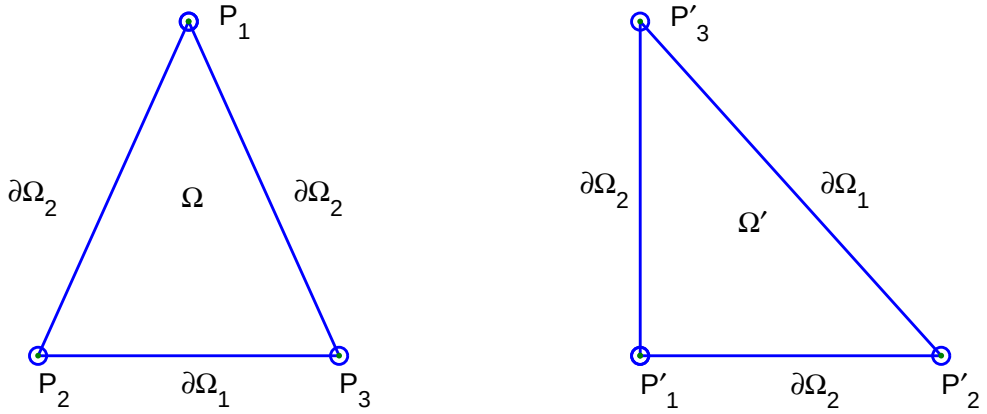


Рис. 3.15. геометрия треугольных области Ω и контрольный треугольник Ω'

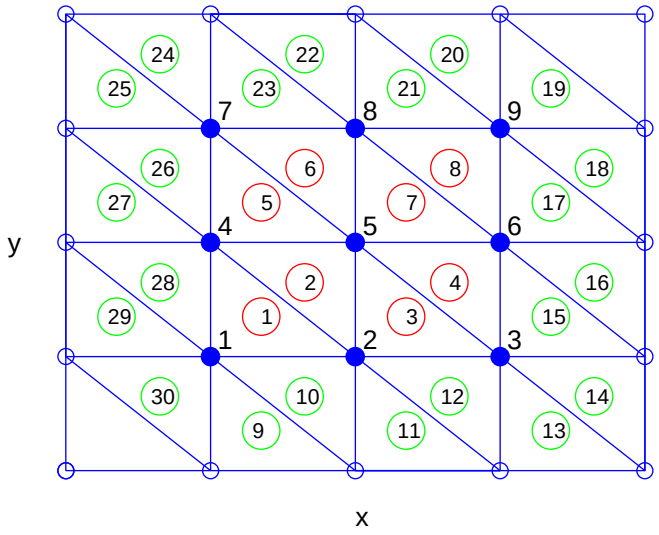


Рис. 3.16. Однородная треугольная сетка в прямоугольной области.

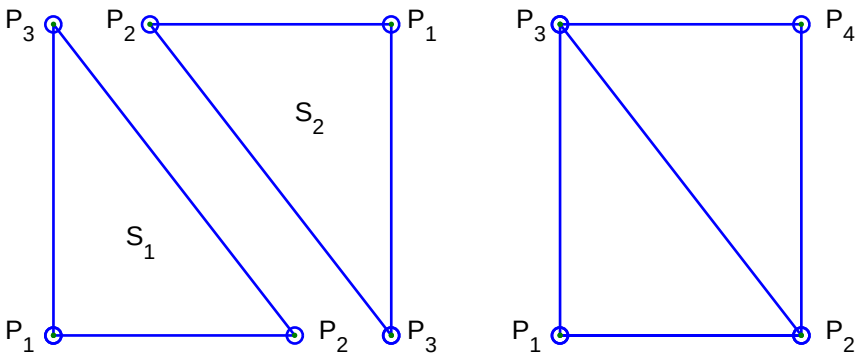


Рис. 3.17. Сборка двух конечных элементов.

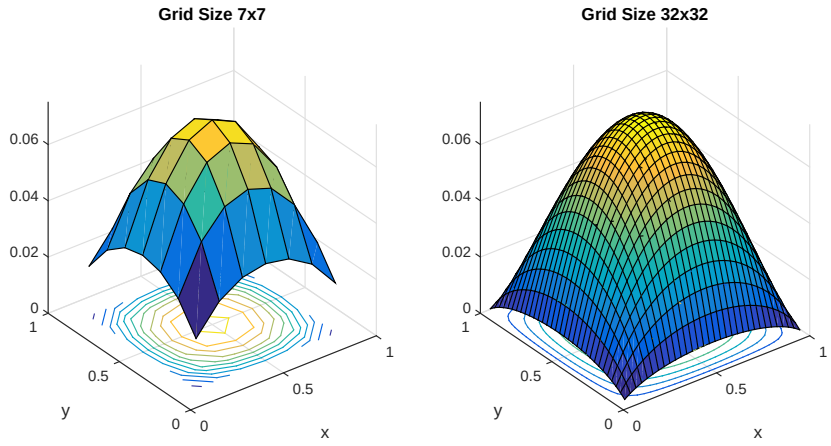


Рис. 3.18. Решение уравнения Пуассона методом конечных элементов с различными размерами шага четки

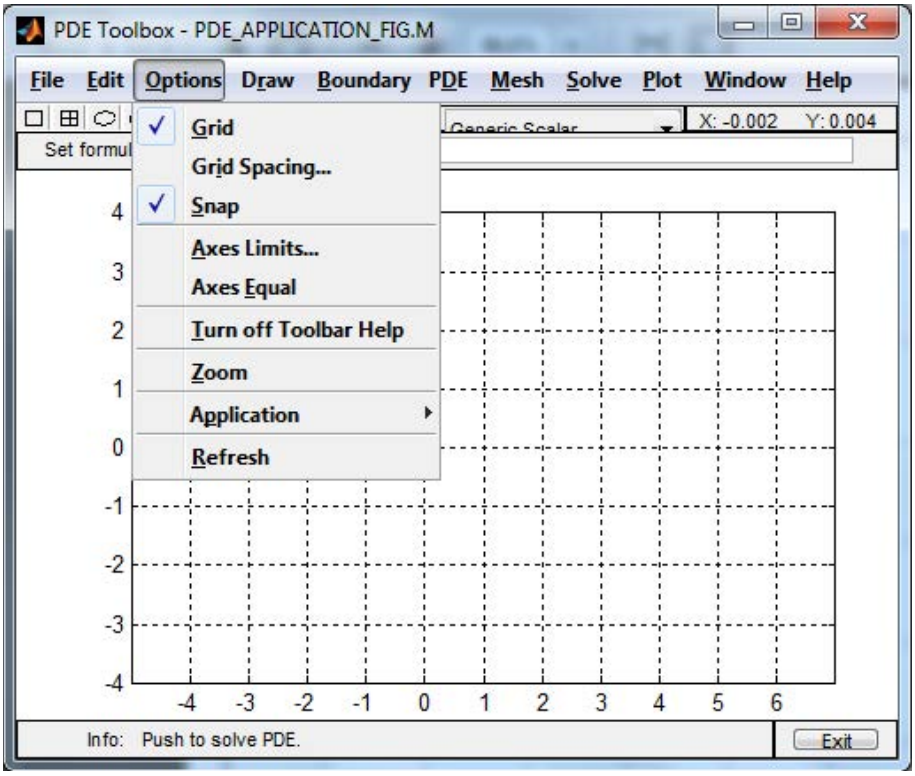


Рис. 3.19. Графический интерфейс приложения **PDETool** и некоторые полезные опции.

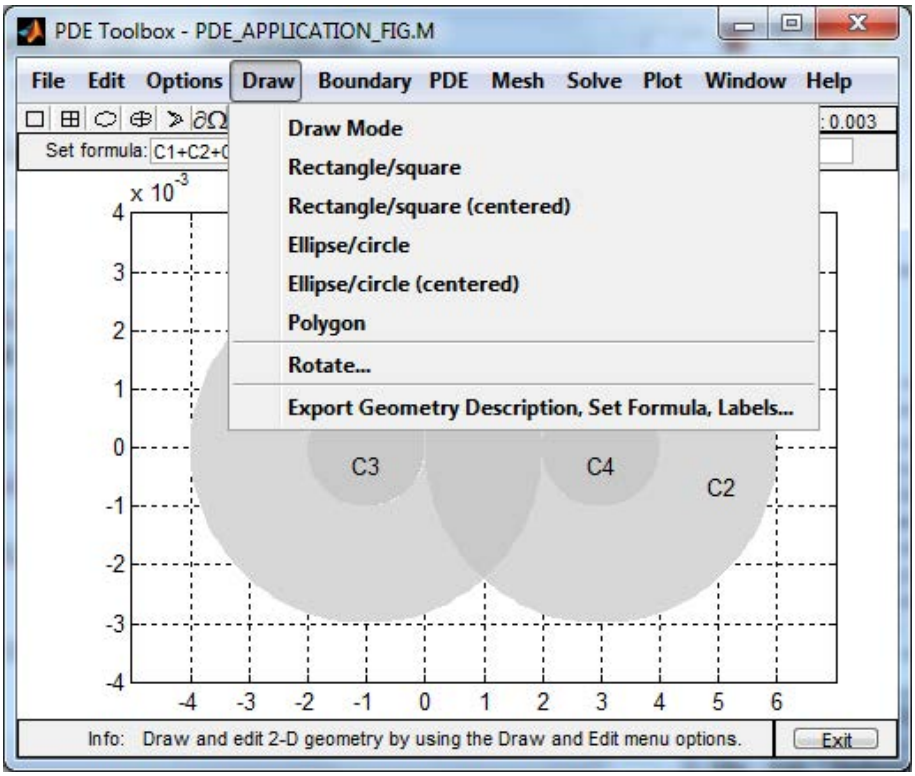


Рис. 3.20. Задание декомпозиционной геометрии задачи.

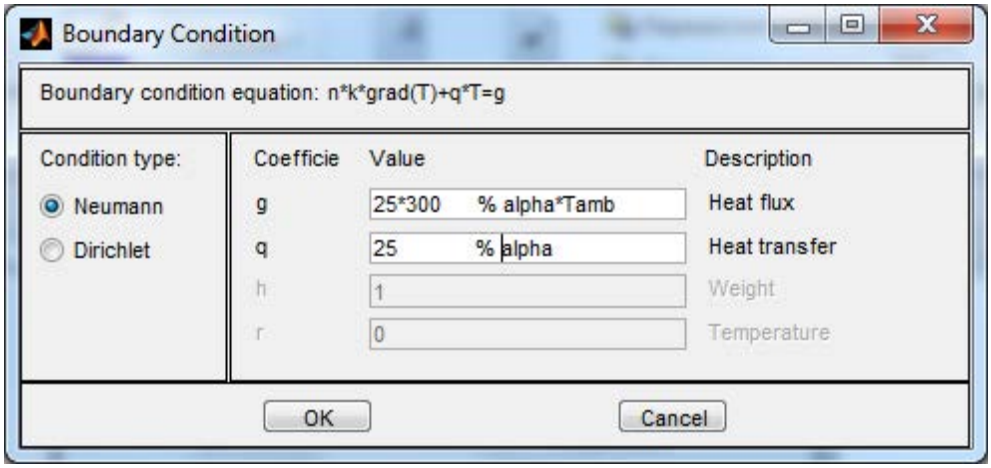


Рис. 3.21. Диалоговое окно для установки краевых условий

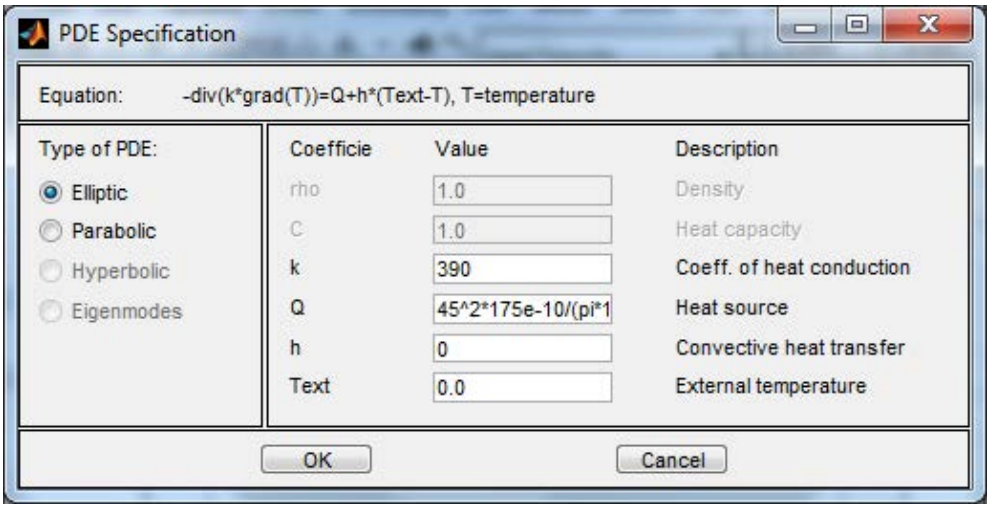


Рис. 3.22. Диалоговое окно для установки праметров в области поперечного сечения проводника

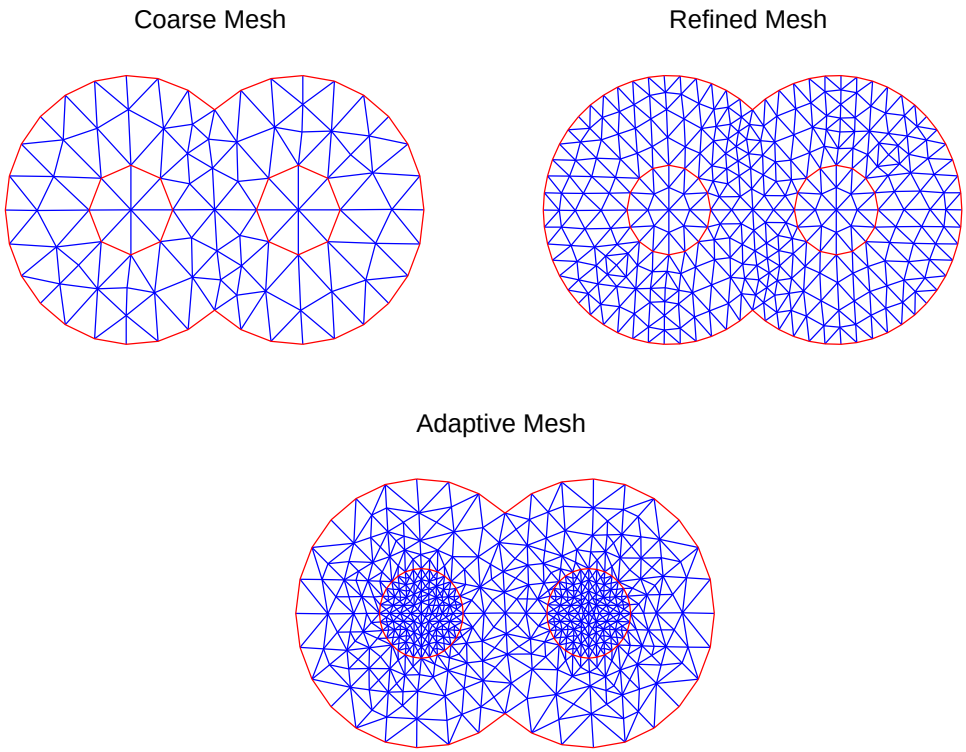


Рис. 3.23. Примеры триангуляции области.

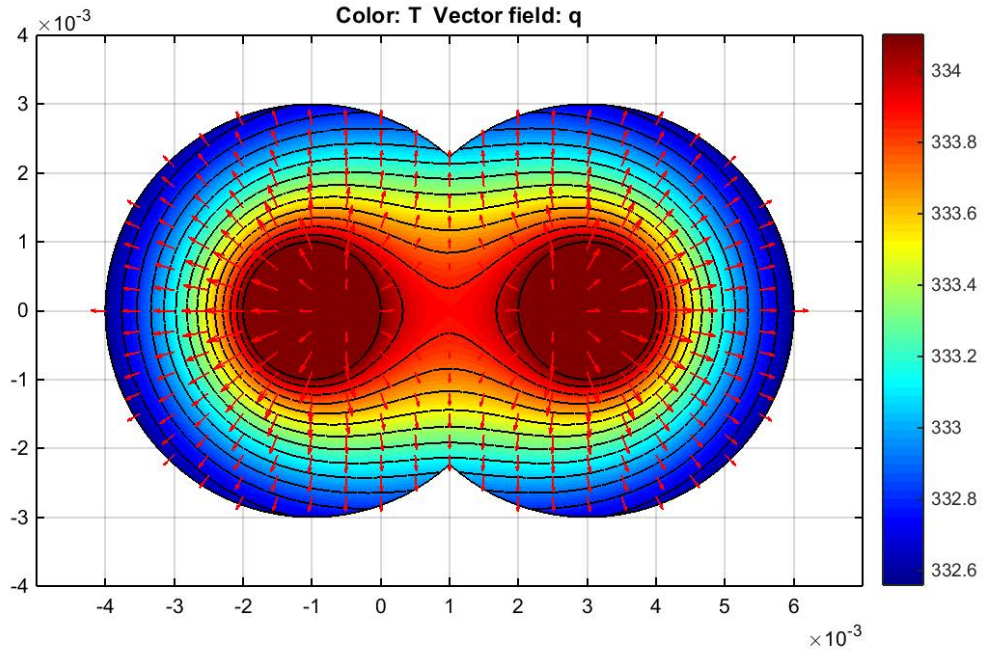


Рис. 3.24. Поле температур и тепловых потоков в поперечном сечении кабеля при коротком замыкании согласно результатам численного эксперимента с использованием математической модели (3.273) – (3.276), $\alpha_p = 0$.

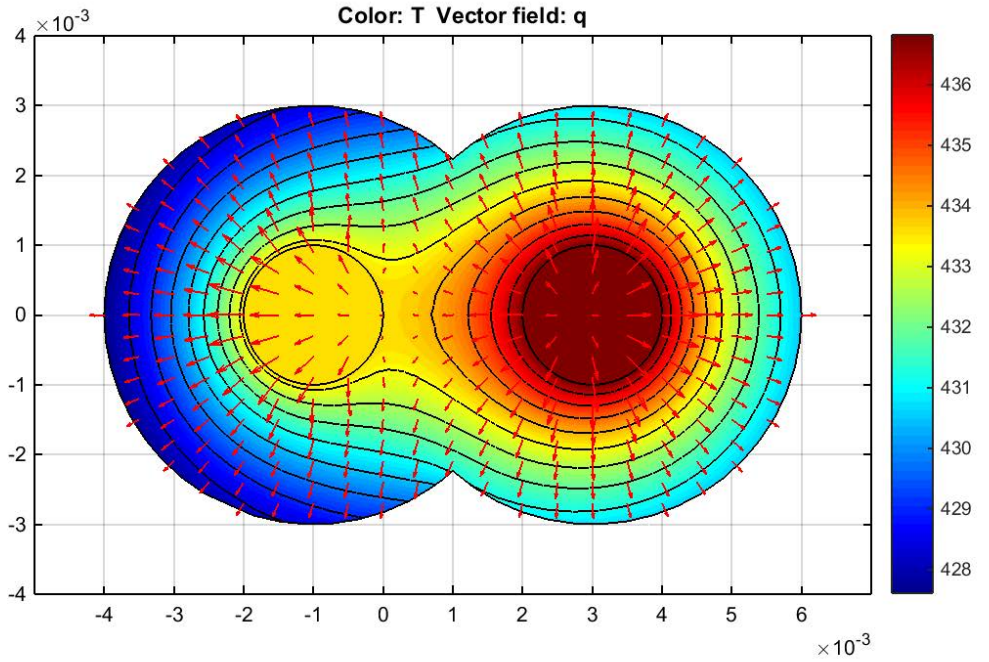


Рис. 3.25. Поле температур и тепловых потоков в поперечном сечении кабеля при коротком замыкании согласно результатам численного эксперимента с использованием математической модели (3.273) , (3.276), (3.280). В правом проводнике мы пренебрегли температурной зависимостью удельного сопротивления ($\alpha_p = 0$, $\beta_p = 0$).

Глава 4

Методы оптимизации

4.1. Введение

Проблема оптимизации — одна из наиболее важных и распространенных проблем, встречающихся при решении научных и инженерных задач как теоретического, так и прикладного характера. Во многих случаях основной целью исследования, организации и управления различными процессами, создания и проектирования разнообразных конструкций является определение наилучших в некотором смысле условий, решений или значений параметров.

Чтобы решать самые разнообразные задачи оптимизации, необходимо иметь математическую модель решаемой задачи.

Пусть $\mathbb{R}^N$ — это N-мерное Евклидово пространство.

В большинстве случаев математическая модель определяется следующим образом:

Дано: множество(область) $D \subset \mathbb{R}^N$ и функция $f : D \to \mathbb{R}$.

Найти: вектор (точку, решение) $\mathbf{x}^* \in D$ такой, что

$$f(\mathbf{x}^*) = \min_{\mathbf{x} \in D} f(\mathbf{x})$$

для задач минимизации, и

$$f(\mathbf{x}^*) = \max_{\mathbf{x} \in D} f(\mathbf{x})$$

для решения задачи максимизации.

Множество (область) D для функции f называется **область поиска** или **множество допустимых решений**, элементы множества D принято называть **допустимыми решениями**, $\mathbb{R}^N$ — это N-мерное Евклидово пространство. Функция f называется, как правило, **функция!целевая** или **функция цели**. Целевая функция, в широком смысле этого слова, есть математическое выражение критерия оптимальности.

Решить задачу оптимизации – это значит найти такой набор переменных из множества допустимых решений, которому соответствует наилучшее значение критерия на этом множестве. Критерий в общем случае может оценивать свойства как желательные (например, качество, производительность, прочность, прибыль и т.д.),

так и нежелательные (расход ресурсов, потери мощности, помехи и т.д.). В первом случае говорят о максимизации критерия, а во втором — о его минимизации.

Задачи минимизации и максимизации взаимосвязаны, и могут быть заменены друг на друга в результате замены знака перед функцией на противоположный. Вектор $\mathbf{x}^*$, определяющий экстремум целевой функции, называют оптимальным. Экстремум (или оптимум) может быть как глобальным (функция достигает в этой точке своего минимального или максимального значения на множестве), так и локальным (значение функции минимальное или максимальное в границах конечной окрестности множества , но не на всем множестве). В дальнейшем для определенности будем говорить о минимизации целевой функции.

Множество допустимых решений , как правило, определяется системой линейных или нелинейных ограничений (условий), накладываемых на вектор $\mathbf{x}$. В этом случае задача оптимизации называется условной. Если ограничения отсутствуют, то задача называется задачей безусловной оптимизации. Хотя в большинстве реальных задач присутствуют ограничения, определяющие область допустимых решений, изучение методов безусловной оптимизации важно по нескольким причинам. Во-первых, до тех пор, пока точка поиска находится внутри области допустимых решений, процесс определения направления поиска и шага перемещения вдоль этого направления может быть таким же, как в одном из алгоритмов безусловной оптимизации. Единственным отличием является необходимость постоянно осуществлять проверку выполнения ограничений. Во-вторых, задача условной оптимизации может быть преобразована в эквивалентную задачу безусловной оптимизации.

В настоящее время разработано множество численных методов условной и безусловной оптимизации. Для успешного применения конкретного метода необходимо знать для какого класса задач его можно использовать, какова его скорость сходимости, вычислительная сложность и т.д.

Ряд методов условной и безусловной оптимизации реализованы как в известных коммерческих программных пакетах, таких как Microsoft Excel, Matlab, MathCad, Mathematic, Maple и др., так и в свободно распространяемых пакетах, например, в GNU Octave, FreeMat, Scilab и др.

Пакет Microsoft Excel предоставляет возможность решения задач оптимизации в надстройке «Поиск решения». В диалоговом окне «Параметры» надстройки «Поиск решения» можно выбрать один из следующих алгоритмов или методов поиска решения:

— Нелинейный метод обобщенного понижающего градиента (ОПГ), который используется для гладких нелинейных задач.

— Симплекс-метод, который используется для линейных задач.

— Эволюционный метод, который используется для негладких задач.

Этот набор методов может быть расширен за счет подключения любых других алгоритмов оптимизации, код которых можно разработать на VBA.

В состав программного пакета Matlab входят пакеты расширения Optimization Toolbox и Global Optimization Toolbox, функции которых реализуют основные алгоритмы оптимизации.

Понимание работы конкретного алгоритма оптимизации позволяет эффективно использовать встроенные функции при решении конкретной прикладной проблемы. Кроме того, возможности данного пакета позволяют разрабатывать новые алгорит-

мы в виде m-file и m-function, которые можно запускать из рабочей среды или из редактора MatLab.

GNU Octave, FreeMat и Scilab — это пример свободно распространяемых пакетов, в которых реализованы многие функции аналогичные Matlab. Согласно проведенным исследованиям, GNU Octave — это высокоуровневый интерпретирующий язык для математических расчетов.Он предоставляет возможности для численного решения линейных и нелинейных задач, а также для выполнения других численных экспериментов. Язык Octave очень похож на язык Matlab, так что большинство программ легко переносимы из одной среды в другую. Пакет Octave-Forge дополняет основную функциональность Octave. В Octave-Forge в виде функций реализованы большинство алгоритмов оптимизации, которые встречаются в Matlab.

С помощью описанных выше пакетов можно с успехом решать не только научные задачи, но и несложные инженерные проблемы. Более того, средствами Matlab могут быть успешно решены и некоторые довольно сложные инженерные задачи, такие, как поиск спектра частот собственных колебаний и критических сил потери устойчивости стержневых, пластинчатых и оболочечных систем, решение краевых задач для упругих систем и задач сейсмостойкости сооружений и др.

Однако при проектировании сложных технических систем, требующем детального моделирования 3-х мерных физических объектов, при проведении серийных численных экспериментов использование Matlab вызывает затруднение. В этом случае целесообразно воспользоваться преимуществами современных систем автоматизации инженерных расчетов (или систем инженерного анализа) — CAE (Computer-aided engineering) систем. CAE системы – это разнообразные программные продукты, предназначенные для решения различных инженерных задач. Расчетная часть продукта чаще всего основана на численных методах решения дифференциальных уравнений. Современные CAE системы могут иметь встроенный модуль для создания геометрической модели или применяются совместно с CAD (Computer-aided design) системами или интегрируются в CAD систему, в этом случае получаются гибридные CAD/CAE системы.

Одним из лидеров CAE-рынка в течение многих последних лет является фирма ANSYS Inc., которая разрабатывает и предлагает широкую линейку программных продуктов для автоматизированного инженерного анализа. Одним из основных программных продуктов фирмы является ANSYS – мощный инструмент для решения задач прочности, теплофизики и электромагнетизма. Оптимизация в ANSYS выполняется на основе конечно-элементного анализа. В основе оптимизации лежит создание параметризованных расчетных моделей с использованием ANSYS Parametric Design Language (APDL). В ANSYS разработаны два метода проведения оптимизации с использованием поверхности отклика:

— Sub-problem approximation method,

— First-order method.

Как указано в документации по ANSYS, первый метод оказывается более эффективным с точки зрения временных затрат, однако метод первого порядка более надежен, отличается высокой точностью, хотя и требует больших вычислительных ресурсов.

В среде ANSYS Workbench также существуют средства для проведения оптимизации конструкций – DesignXplorer. DesignXplorer предлагает несколько методов

оптимизации – метод последовательного квадратичного программирования, генетический алгоритм и др., которые могут быть использованы как для решения многокритериальных задач, так и для задач с единственной целевой функцией.

Все специалисты по CAE знают, что в большинстве CAE пакетов можно сделать сложную, очень представительную модель, но чего нельзя делать в этих всех пакетах так это решать сложные самостоятельные задачи математического программирования, как при работе в математических пакетах. Интеграция математических пакетов с CAE пакетами помогает устранить их недостатки – сложность моделирования 3-х мерных физических объектов, присущую математическим пакетам, и отсутствие свободы в математическом программировании в CAE пакетах. Такой подход получает все большее распространение, т.к. позволяет избавиться от трудоемкой работы по созданию собственной численной модели, заменив ее моделью в готовом CAE пакете, и, следовательно, предоставляет возможность сосредоточиться на исследовании собственно процессов в технической системе и проведении оптимизационных вычислений.

4.2. Методы безусловной оптимизации

4.2.1. Базовые принципы

Пусть функция f определена и дважды непрерывно дифферецируема в некоторой окрестности точки $\mathbf{x}^*$.

Тогда необходимым условием существования безусловного оптимума в точке $\mathbf{x}^*$ является

$$\nabla f(\mathbf{x}^*) = 0,$$

где

$$\nabla f(\mathbf{x}) = \left(\frac{\partial f}{\partial x_1}(\mathbf{x}), \ldots, \frac{\partial f}{\partial x_N}(\mathbf{x}) \right) \tag{4.1}$$

градиент функции f по $\mathbf{x}$.

Это условие и дифференцируемость функции f являются необходимыми, но не достаточными с той точки зрения, что ситуация $\nabla f(\mathbf{x}) = 0$ может соответствовать не только максимальной или минимальной, но и седловой точкам. Однако, если $\nabla f(\mathbf{x}) \neq 0$, тогда точка $\mathbf{x}$ не может быть точкой оптимума.

Достаточные условия включают необходимые условия и ряд других условий, выполнение совокупности всех этих условий позволяет заявлять о существовании оптимума.

Достаточными условиями существования безусловного оптимума в точке $\mathbf{x}^*$ являются

$$\begin{cases} \text{функция } f \text{ дифференцируема в } \mathbf{x}^*, \\ \nabla f(\mathbf{x}^*) = 0, \\ \text{Гессиан } \nabla^2 f(\mathbf{x}^*) \text{ положительно определен} \end{cases}$$

для решения задач минимизации и

$$\begin{cases} \text{функция } f \text{ дифференцируема в } \mathbf{x}^*, \\ \nabla f(\mathbf{x}^*) = 0, \\ \text{Гессиан } \nabla^2 f(\mathbf{x}^*) \text{ отрицательно определен} \end{cases}$$

для решения задач максимизации.

Здесь **гессиан** $\nabla^2 f(\mathbf{x})$ (матрица Гессе) — это матрица квадратичной формы, образованная вторыми частными производными функции в точке $\mathbf{x}$

$$\begin{pmatrix} \dfrac{\partial^2 f}{\partial x_1 \partial x_1}(\mathbf{x}) & \dfrac{\partial^2 f}{\partial x_1 \partial x_2}(\mathbf{x}) & \cdots & \dfrac{\partial^2 f}{\partial x_1 \partial x_N}(\mathbf{x}) \\ \dfrac{\partial^2 f}{\partial x_2 \partial x_1}(\mathbf{x}) & \dfrac{\partial^2 f}{\partial x_2 \partial x_2}(\mathbf{x}) & \cdots & \dfrac{\partial^2 f}{\partial x_2 \partial x_N}(\mathbf{x}) \\ \vdots & \cdots & \ddots & \vdots \\ \dfrac{\partial^2 f}{\partial x_N \partial x_1}(\mathbf{x}) & \dfrac{\partial^2 f}{\partial x_N \partial x_2}(\mathbf{x}) & \cdots & \dfrac{\partial^2 f}{\partial x_N \partial x_N}(\mathbf{x}) \end{pmatrix}. \tag{4.2}$$

Если все вторые частные производные функции f непрерывны, тогда Гессиан — симметричная матрица. По матрице Гессе можно судить об искривлении функции и о поведении градиента.

Положительная (или отрицательная) определенность матрицы Гессе означает, что соответствующая квадратичная форма

$$H(f, \mathbf{x}, h) = \sum_{i=1}^{N} \sum_{j=1}^{N} \frac{\partial^2 f}{\partial x_i \partial x_j}(\mathbf{x}) h_i h_j$$

сохраняет свой знак для всех $h \neq 0$.

Это утверждение может быть проверено с помощью критерия Сильвестра:

— все главные миноры матрицы Гессе должны быть положительны (для положительной определенности),

— все главные миноры матрицы Гессе имеют знак $(-1)^s$, где s — порядок минора (для отрицательной определенности).

Существует два основных подхода к решению задач безусловной оптимизации. Первый из них основан на применении необходимых и достаточных условий безусловного экстремума, второй – на выполнении последовательности определенных итераций. Применение первого подхода возможно лишь в том случае, когда целевая функция задана явно и имеет непрерывные производные до второго порядка включительно. Для решения большинства прикладных задач используют второй подход.

Введем следующие обозначения: $\mathbf{x}^0$ — для начальной (стартовой) точки, $\mathbf{x}^*$ — для точки оптимума и $\mathbf{x}^k$ — для (текущей) точки на k-ой итерации. Итерацией во втором подходе для задачи минимизации будем называть переход от точки $\mathbf{x}^k$ к точке $\mathbf{x}^{k+1}$, для которых

$$f(\mathbf{x}^k) > f(\mathbf{x}^{k+1}), \quad k = 0, 1, \ldots$$

Для выполнения итерации необходимо выбрать направление движения из точки $\mathbf{x}^k$ и определить шаг вдоль этого направления. Математически итерация описывается соотношением :

$$\mathbf{x}^{k+1} = \mathbf{x}^{k+1} + s^k \mathbf{d}^k$$

где $\mathbf{d}^k \in \mathbb{R}^N$ — это **направление поиска**, s^k — это положительная скалярная величина, называемая **шагом поиска**, которая определяется на этапе одномерного (линейного) поиска.

В связи со значимостью методов одномерного поиска остановимся на следующих важных моментах:

— В методах одномерного поиска предполагается, что искомая оптимальная точка принадлежит априорно известному начальному интервалу неопределенности.

— Большинство известных методов одномерного поиска разработано для унимодальных функций. Если оптимизируемая функция локально не унимодальна, то большой шаг может привести к неограниченному решению, хотя выполняется поиск локального экстремума (рис. 4.2.1). Поэтому необходимо предусмотреть процедуру по корректировке величины шага.

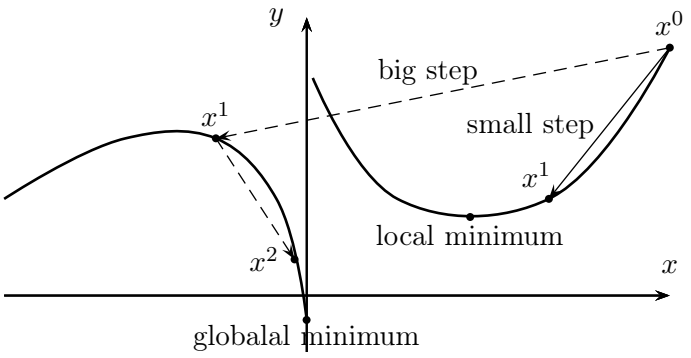


Рис. 4.1. Влияние шага одномерного поиска на решение

— В ряде работ описываются различные методы одномерного поиска, среди которых можно выделить так называемые последовательные (например, метод дихотомии, метод деления пополам, метод золотого сечения, метод Фибоначи и т.д.) и непоследовательные методы (например, метод равномерного поиска). В непоследовательных методах набор точек, в которых производится вычисление функции, задается заранее. В последовательных — эти точки выбираются последовательно с учетом результатов предыдущих вычислений. Для сравнения различных методов используется понятие их эффективности. Относительная эффективность двух хорошо известных методов Фибоначи и золотого сечения несколько выше эффективности методов дихотомии и деления пополам и существенно превосходит эффективность непоследовательных методов.

— В Matlab поиск локального минимума функции одной переменной для фиксированного интервала осуществляет функция `fminbnd`. В основу алгоритма работы этой функции положены два метода — метод золотого сечения и метод параболической интерполяции.

В настоящее время разработаны различные итерационные методы безусловной оптимизации, которые отличаются между собой способами выбора направления поиска и длиной шага вдоль этого направления. Если при выборе используется какой-либо случайный механизм, то алгоритм поиска называется методом случайного поиска, сочетание элементов стохастического и детерминированного подходов при выборе необходимых величин реализовано в генетических алгоритмах, наконец, если направление поиска и длина шага выбираются однозначно, то алгоритм поиска называется детерминированным. Все детерминированные алгоритмы поиска можно условно разделить на три типа: методы нулевого порядка, методы первого порядка и методы второго порядка. **Методы нулевого порядка** в процессе поиска оптимального решения используют только значения функции. В **методах первого порядка** для поиска оптимального решения используются значения функции и значения ее первых производных.И, наконец, в **методах второго порядка** наряду со значениями функции используются значения ее первых и вторых производных.

В следующих разделах будет выполнено обсуждение и сравнение специфических особенностей различных методов. Результаты любого сравнения методов в значительной степени зависят от того, как эти методы запрограммированы. Критерии окончания счета, используемый метод одномерного поиска, даже изменения начального шага в одномерном поиске и т.п. сильно влияют на эффективность метода. Для сравнение различных методов безусловной оптимизации предлагается использовать такие критерии, как точность достижения оптимального решения при заданных критериях окончания счета и число необходимых вычислений целевой функции или производной. Получение объективных результатов относительно эффективности сравниваемых методов возможно лишь внутри каждой категории при использовании единого набора тестовых задач, критериев окончания счета и процедур подсчета количества вычислений функций или производных.

Существуют целевые функции, которые можно назвать «классическими» тестовыми задачами, поскольку они много раз применялись различными авторами для сравнения качества работы разных алгоритмов. Примеры тестовых функций приведены на рис. 4.2–4.3.

Построение линий уровня для приведенных функций выполнено в Matlab. Ниже приведен скрипт для построения линий уровня для функции Розенброка:

```
[X1,X2]=meshgrid(-3:0.01:3);
F=100*(X2-X1.^2).^2+(1-X1).^2;
[C,h]=contour(X1,X2,F,[1,10,100,1000]);
clabel(C,h)
```

В качестве критериев окончания счета можно использовать следующие критерии:

$$|f(\mathbf{x}^k) - f(\mathbf{x}^{k+1})| \leqslant \varepsilon_0,$$

$$|x_i^k - x_i^{k+1}| \leqslant \varepsilon_i \quad \text{для всех} \quad i = 0, \ldots, N,$$

где ε_0 и ε_i — заданные малые положительные числа.

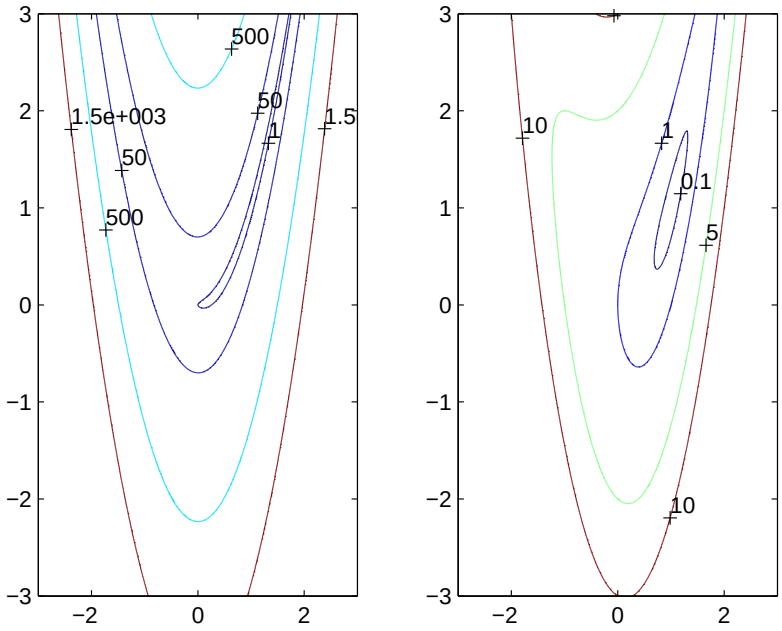


Рис. 4.2. Функция Розенброка $f(x) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2$ и функция $f(x) = (x_2 - x_1^2)^2 + (1 - x_1)^2$

4.2.2. Методы нулевого порядка

В типичных методах нулевого порядка направление поиска полностью определяется на основании последовательных вычислений целевой функции. Эти методы не требуют регулярности и непрерывности целевой функции и существования производных. Это является существенным достоинством при решении сложных прикладных задач.

Рассмотрим кратко стратегии поиска, применяемые в наиболее известных методах нулевого порядка.

Метод покоординатного спуска (метод Гаусса)

Метод покоординатного спуска (метод Гаусса) на каждой итерации по очереди минимизирует целевую функцию вдоль каждой из координат. На каждой итерации решение улучшается за счет минимизации вдоль выбранной i-ой координаты (компоненты) x_i вектора $\mathbf{x} \in D$ при фиксированных остальных координатах. Стратегия метода проста и включает ледующие основные этапы:

given a starting point $\mathbf{x}^0 = (x_1^0, \ldots, x_N^0) \in D$
$i = 1$, $k = 0$,
while the stopping criterion is not satisfied
1. Determinate the search direction $\mathbf{d}^{k+1}$ along x_i.
2. Choose the method of one-dimensional (line) search.

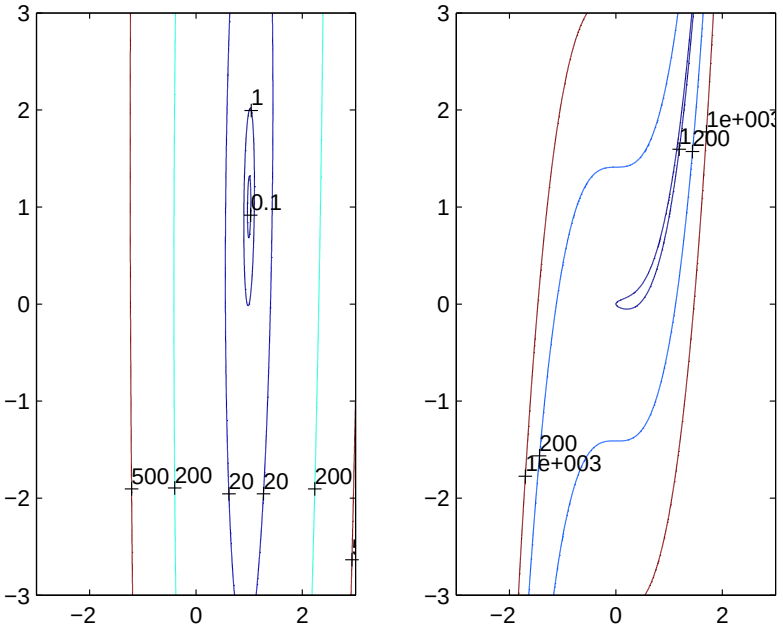


Рис. 4.3. Тестовые функции $f(x) = (x_2 - x_1^2)^2 + 100(1 - x_1)^2$ и $f(x) = 100(x_2 - x_1^3)^2 + (1 - x_1)^2$

3. Minimize $f(\mathbf{x})$ along x_i. Find the point $\mathbf{x}^{k+1} = (x_1^{k+1}, x_2^k \ldots, x_N^k)$.
4. Change the starting point to $\mathbf{x}^{k+1}$.
5. If $i < N$, then $i = 1 + 1$ else $i = i$.

end.

Для поиска минимума по одной координате вектора $\mathbf{x}$ могут применяться различные методы линейного поиска. Более того, настройки одного и того же метода линейного поиска также могут отличаться для разных координат. Достоинством метода является его простота при определении перемещения в пространстве координат. Для гладких функций метод обеспечивает сходимость к точке локального минимума.

Скрипт Matlab и результаты работы **метода Гаусса** с использованием метода деления пополам в качестве метода линейного поиска приведены ниже на рисунках рис. 4.4 и 4.5.

```
function Gauss_bisection = Gauss_bisection()
t = input ('Enter the left grid boundary:','s'); Ax = str2num(t);
t = input ('Enter the right grid boundary:','s'); Bx = str2num(t);
t = input ('Enter the upper grid boundary:','s'); A1y = str2num(t);
t = input ('Enter the bottom grid boundary:','s');B1y = str2num(t);
t = input ('Enter x_0:','s');x_0 = str2num(t);
t1 = input ('Enter y_0:','s');y_0= str2num(t1);
t1 = input ('Enter eps:','s');eps= str2double(t1);
```

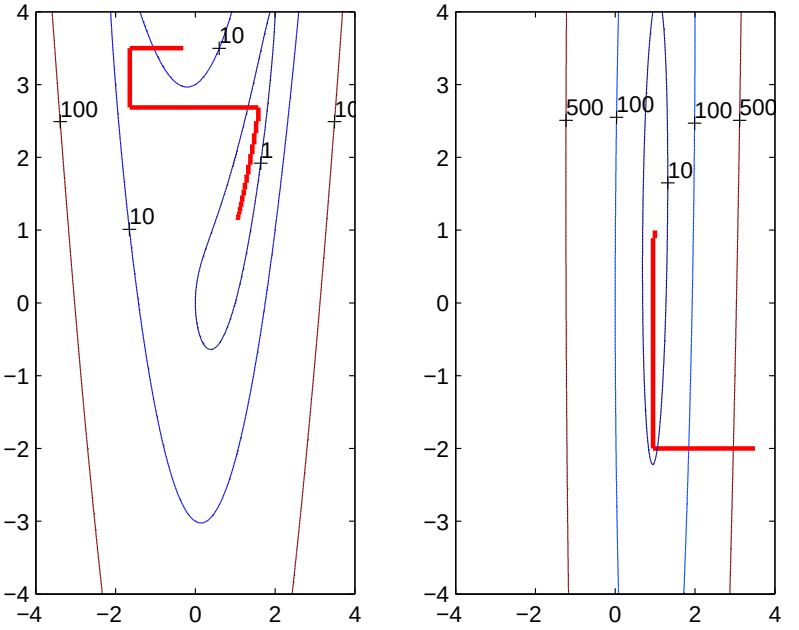


Рис. 4.4. Удачные варианты работы метода Гаусса на 2 и 3 тестовых функциях

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
[X,Y] = meshgrid(Ax:.02:Bx);
Z=(Y-X.^2).^2 + (1 - X).^2;
[C,h]= contour(X,Y,Z,[0,0.1,1,3,5,10,100]);
clabel(C,h)
v = [x_0, y_0];
min_z = fun2(v);
zl = min_z;
min_x = x_0;
min_y = y_0;
min_con=0;i=1;
count_steps = 0;
while abs(zl-min_con)>eps*eps*eps
count_steps = count_steps + 1;
A = Ax;
B = Bx;
A1 = A1y;
B1 = B1y;
y_0 = min_y;
x_0 = min_x;
zl = min_z;
while (B - A) > 3*eps
m = A + (B - A)/4;
```

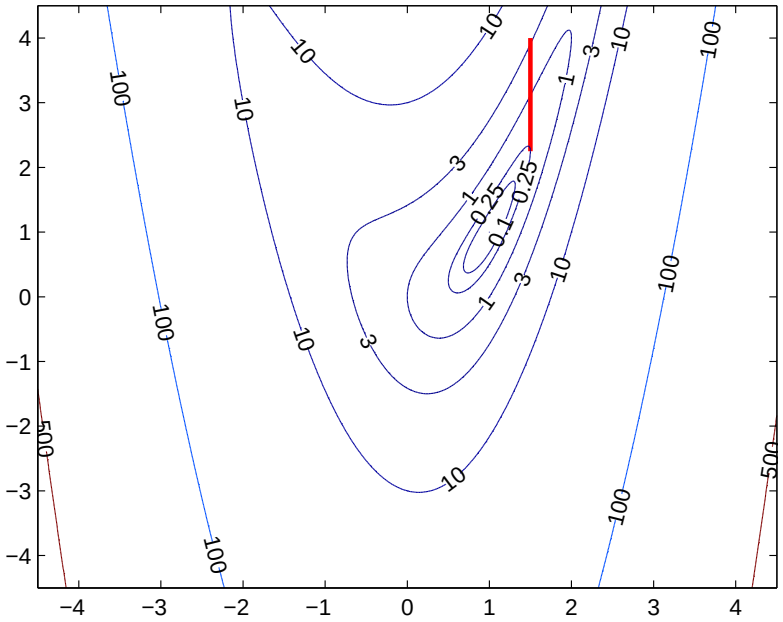


Рис. 4.5. Неудачные варианты работы метода Гаусса на 2 тестовой функции

```
n = B - (B - A)/4;
x_sr = (B + A)/2;
v1 = [m, y_0];
v2 = [x_sr, y_0];
v3 = [n, y_0];
if fun2(v1) < fun2(v2)
B = x_sr;
end
if fun2(v1) > fun2(v2)
if fun2(v3) < fun2(v2)
A = x_sr;
end
if fun2(v3) >= fun2(v2)
A = m;
B = n;
end
end
end
while A < B
v4 = [A, y_0];
Z = fun2(v4);
if min_z > Z
min_z=Z;
```

```
min_x=A;
end
A = A + eps*eps;
end
Xl = [x_0,min_x];
Yl = [y_0, y_0];
Zl = [zl, min_z];
line([x_0,min_x],[y_0, y_0],[zl, min_z],'Color','g','LineWidth',3)
zl = min_z;
while (B1 - A1) > 3*eps
m = A1 + (B1 - A1)/4;
n = B1 - (B1 - A1)/4;
y_sr = (B1 + A1)/2;
v1y = [min_x,m];
v2y = [min_x, y_sr];
v3y = [min_x, n];
if fun2(v1y) < fun2(v2y)
B1 = y_sr;
end
if fun2(v1y) > fun2(v2y)
if fun2(v3y) < fun2(v2y)
A1 = y_sr;
end
if fun2(v3y) >= fun2(v2y)
A1 = m;
B1 = N;
end
end
end
while A1 < B1
v4y = [min_x, A1];
Z = fun2(v4y);
if min_z > Z
min_z=Z;
min_y=A1;
end
A1 = A1 + eps*eps;
end
min_con=min_z;
Xl1 = [min_x,min_x];
Yl1 = [y_0, min_y];
Zl1 = [zl, min_z];
line(Xl1,Yl1,Zl1,'Color','g','LineWidth',3)
i = i+1;
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Метод не всегда дает точный результат. Погрешности могут возникнуть как из-за погрешностей в линейном поиске, так и из-за особенностей целевой функции. Иногда в результате линейного поиска по одной из координат решение может оказаться в точке перегиба линий уровня, расположенной как показано на рис. 4.5. Тогда любое движение по следующей координате приводит к увеличению значения целевой функции. В этом случае дальнейшее перемещение вдоль координат вектора $\mathbf{x}$ больше не будет выполняться, хотя минимум еще не достигнут.

Метод Хука–Дживса (метод конфигураций)

Метод Хука–Дживса заключается в комбинации двух шагов: **исследующего шага** и **шага по образцу**. В общем виде стратегию поиска в этом методе можно представить следующим образом: (обозначим через

$$\mathbf{d}_1 = (1, 0, \ldots, 0), \quad \mathbf{d}_2 = (0, 1, \ldots, 0), \quad \mathbf{d}_N = (0, 0, \ldots, 1) \tag{4.3}$$

элементы стандартного базиса $\mathbb{R}^N$, λ – это ускоряющий множитель):

given a starting point $\mathbf{x}^0 = (x_1^0, \ldots, x_N^0) \in D$, s^0, λ
$i = 1$, $k = 0$,
while the stopping criterion is not satisfied

1. Perform the exploration step:
   - for i from 1 to N evaluate $f(\mathbf{x})$ along $\mathbf{d}_i$ by stepping $\mathbf{d}_i$ a distance s^k to each side relative to $\mathbf{x}^k$. If there is no decrease of $f(\mathbf{x})$, then $x_i^{k+1} = x_i^k$, else x_i^{k+1} is the point, where $f(\mathbf{x})$ decrease,
   - evaluate $f(\mathbf{x}^{k+1})$. If $f(\mathbf{x}^{k+1}) = f(\mathbf{x}^k)$, then the s^k is reduced and searches are made from the $\mathbf{x}^k$, else determinate the direction from $\mathbf{x}^k$ to $\mathbf{x}^{k+1}$ as the direction for pattern step (pattern direction).
2. Perform the pattern step:
   - for $\overline{\mathbf{x}}^{k+1} = \mathbf{x}^k + \lambda(\mathbf{x}^k - \mathbf{x}^{k+1})$ evaluate $f(\overline{\mathbf{x}}^{k+1})$ along pattern direction by stepping $\lambda(\mathbf{x}^k - \mathbf{x}^{k+1})$, if $f(\overline{\mathbf{x}}^{k+1}) < f(\mathbf{x}^{k+1})$, then $\mathbf{x}^{k+1} = \overline{\mathbf{x}}^{k+1}$,
   - return to the exploration step

end.

Величина исследующего шага (s^k) вдоль $\mathbf{d}_i$ может быть различной и, как видно из алгоритма, может меняться в процессе исследующего шага. Возможны модификации алгоритма, в которых в процессе исследующего шага выполняется минимизация целевой функции по каждой переменной с помощью какого-нибудь линейного поиска. Кроме того, в процессе шага по образцу можно также искать минимум функции спомощью какого-нибудь линейного поиска или изменять λ в зависимости от значения функции $f(\mathbf{x})$.

Ниже приведен скрипт Matlab для метода Хука–Дживса с использованием минимизации функции (метод деления пополам) на исследующем шаге и на этапе поиска по образцу.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function HJ_bisection = HJ_bisection()
%% bisection search on exploration step
t= input('Enter the left grid boundary:', 's'); a = str2double(t);
```

```
t= input('Enter the right grid boundary:', 's'); b = str2double(t);
t = input ('Enter x_0:','s');x= str2double(t);
t = input ('Enter y_0:','s');
y= str2double(t);
[X,Y] = meshgrid(a:0.01:b);
Z = ftest2(X,Y);
[C,hh] = contour(X,Y,Z,[0.3,1,3,5,10,40,100]);
clabel(C,hh);
N = 0;
x0 = x;
y0 = y;
i=0;
while N <= 5
line([x0,x0],[y0,y0],[ftest2(x0,y0),ftest2(x0,y0)],
'Marker','X','Color','g','LineStyle','-','LineWidth',1);
if (ftest2(f(-10,x0,y0),y0)<(ftest2(f(x0,10,y0),y0)))
minx = f(-10,x0,y0);
else
minx = f(x0,10,y0);
end
if ftest2(minx, y0) > ftest2(x0, y0)
minx = x0;
end
line([minx,minx],[y0,y0],[ftest2(minx,y0),ftest2(minx,y0)],
'Marker','.','Color','g','LineStyle','-','LineWidth',2);
if (ftest2(minx,f_y(-10,y0,minx))<(ftest2(minx,f_y(y0,10,minx))))
miny = f_y(-10,y0,minx);
else
miny = f_y(y0,10,minx);
end
if ftest2(minx, miny) > ftest2(minx, y0)
miny = y0;
end
line([minx,minx],[miny,miny],[ftest2(minx,miny),ftest2(minx,miny)],
'Marker','.','LineStyle','-','LineWidth',2,'Color','r');
h=sqrt((minx-x0)^2+(miny-y0)^2);
k1 = (miny - y0)/(minx - x0);
pb = (minx*y0 - x0*miny)/(minx - x0);
sin_alfa=(miny - y0)/sqrt((minx-x0)^2+(miny-y0)^2);
cos_alfa=(minx - x0)/sqrt((minx-x0)^2+(miny-y0)^2);
iterat=0; x1=x0; y1=y0; k=1; i=14;
while iterat<1
if (ftest2(x1, y1)>ftest2(cos_alfa*k*h+x1,sin_alfa*k*h+y1))
x1=cos_alfa*k*h+x1;
y1=sin_alfa*k*h+y1;
k=k+1;
```

```
else
iterat=iterat+1;
end
i=i+2;
end
if x0 == x1 && y0 == y1
N = N + 1;
h = h / 2;
else
if (ftest2(x0,y0)>ftest2(x1,y1))
line([x0,x1],[y0,y1],[ftest2(x0,y0),ftest2(x1,y1)],
'Marker','.','LineStyle','-','LineWidth',2,'Color','r')
x0 = x1;y0 = y1;i=i+4;i=0;
else
N = N + 1;
h = h / 2;
i=i+2;
end
end
end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Результаты работы метода Хука–Дживса с использованием минимизации функции (деление пополам) на исследующем шаге и на этапе поиска по образцу показаны на рис. 4.6 и 4.7.

Так же, как и в методе Гаусса, в случае сильно вытянутых, изогнутых и круто изгибающихся линий уровня, метод Хука–Дживса может работать со значительными погрешностями.

Метод Розенброка

Метод Розенброка является итерационной процедурой, в нем, подобно методу Хука–Дживса, предусмотрен исследующий поиск. Однако, если в методе Хука – Дживса направления исследующего поиска фиксированы, то в методе Розенброка выбор этих направлений проводят на основании анализа скорости изменения целевой функции. Выбор новых направлений координатных осей $\mathbf{d}^k$ на каждой k–ой итерации определяется с помощью продцедуры Gram-Smidt:

$$a_j = \begin{cases} \mathbf{d}_j, & \text{if} \quad s_j = 0, \\ \sum\limits_{i=j}^{N} \mathbf{d}_i s_i, & \text{if} \quad s_j \neq 0, \end{cases} \qquad b_j = \begin{cases} a_j, & \text{if} \quad j = 1, \\ a_j - \sum\limits_{i=1}^{j-1} (a_i, \mathbf{d}_i)\mathbf{d}_i, & \text{if} \quad j \geqslant 2, \end{cases}$$

где $\mathbf{d}_j = b_j \|b_j\|^{-1}$.

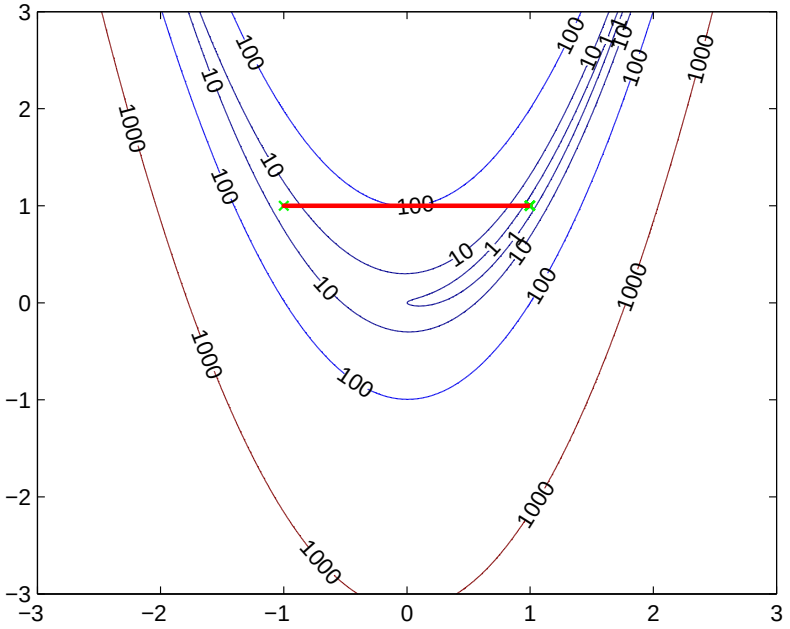


Рис. 4.6. Результат работы метода Хука–Дживса с использованием метода деления пополам на исследующем шаге

В общем виде стратегия поиска следующая (i – номер координаты вектора $\mathbf{x}$, k – номер итерации, s_i^k – шаг вдоль i–ой координаты для k итерации, α – коэффициент растяжения, β – коэффициент сжатия, $\mathbf{d}^0$ определен аналогично выражению (4.3)):

given a starting point $\mathbf{x}^0 = (x_1^0, \ldots, x_N^0) \in D$, $\mathbf{s}^0 = (s_1^0, \ldots, s_N^0) \geqslant 0$, α, β
$i = 1$, $k = 0$,
while the stopping criterion is not satisfied

1. For i from 1 to N evaluate $f(\mathbf{x})$ in the directions of $\mathbf{d}^k$. If $f(\mathbf{x})$ decrease (in the case of a success) then $s_i^k = \alpha s_i^k$ else (in the case of a failure) $s_i^k = \beta s_i^k$.
2. Once a success has been found and exploited in each base direction $\mathbf{d}^k$. Then $x_i^{k+1} = x_i^k$, $s_i^{k+1} = s_i^k$ and the coordinate system is rotated in order to make the first base vector point into the direction of the gradient.

end

Этот метод хорошо приспособлен для поиска оптимума целевой функции, которая представляет собой узкий овраг (или гребень), произвольно расположенный по отношению к варьируемым переменным.

Скрипт Matlab и результаты работы метода Розенброка приведены ниже и на рис. 4.8.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function result = rosenbrok(x0,num,a,b,threshold,dx0, myFunc)
%x0 - starting point; num - the maximum number of failed steps per
iteration;
```

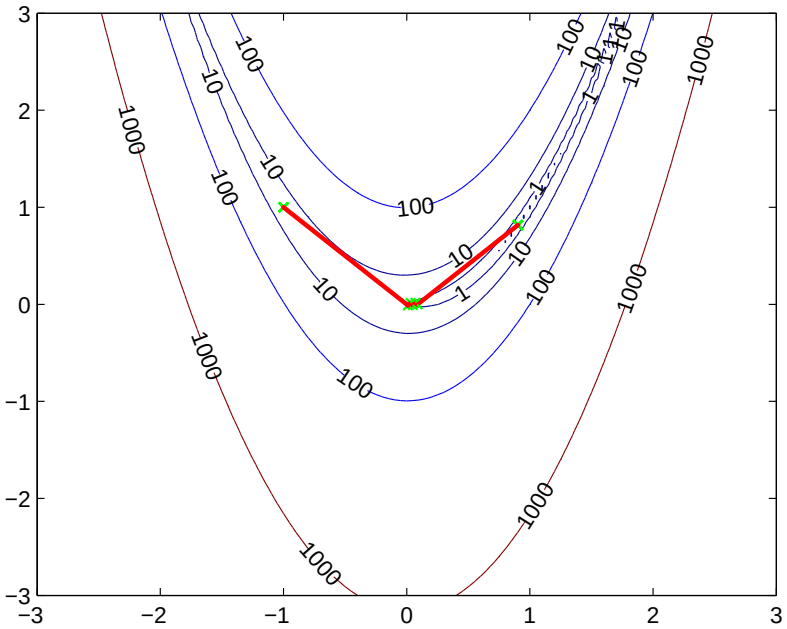


Рис. 4.7. Результаты работы метода Хука–Дживса с использованием метода деления пополам на этапе поиска по образцу

```
%xmax, xmin - the left and the right grid boundaries;
%threshold - stopping criterion
%dx0 - the length of the initial steps; myFunc -function reference
failCount = 0;    isEndWhile = 0;    directions = [1 0; 0 1];
lambda = [0 0]; dx = dx0; xCur = x0; yCur = myFunc(x0(1), x0(2));
xk = x0; yk = yCur; vy = [ xk yk ];
while isEndWhile == 0
yPrev = myFunc(xCur(1), xCur(2));
dxPrev = dx;
for i = 1:2
xNext = xCur;
for j = 1:2
xNext(j) = xNext(j) + directions(i,j) * dx(i);
end
yNext = myFunc(xNext(1), xNext(2));
if(yNext < yCur
xCur = xNext;
yCur = yNext;
lambda(i) = lambda(i) + dx(i);
dx(i) = dx(i) * a;
vy = [vy; xCur yCur];
else
```

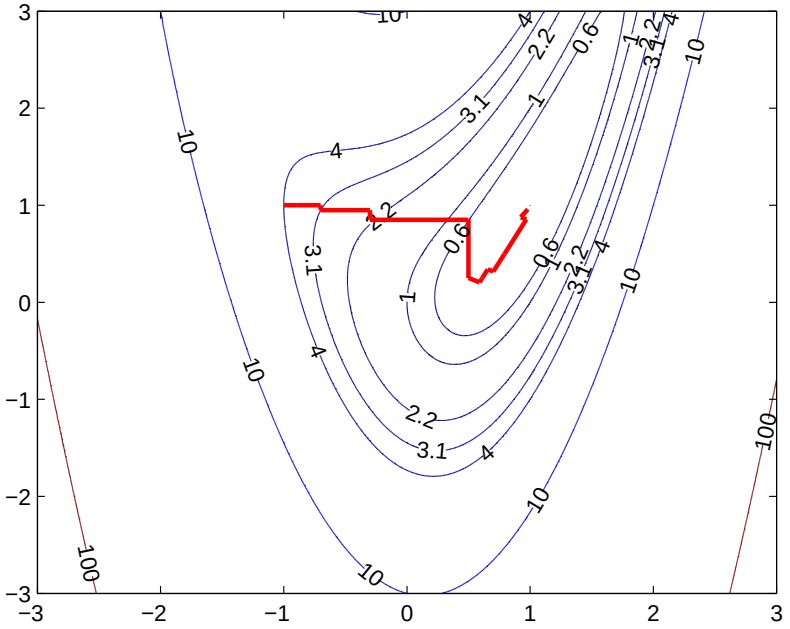


Рис. 4.8. Результат работы метода Розенброка

```
dx(i) = dx(i) * b;
end
end
if(yCur == yPrev)
failCount = failCount + 1;
if(yCur < yk)
dist = 0;
for i = 1:2
dist = dist + (xk(i) - xCur(i))^2;
end
if(sqrt(dist) < threshold)
isEndWhile = 1;
else
xk = xCur;
yk = yCur;
end
directions = GramN(directions, lambda);
lambda = [0 0];
dx = dx0;
failCount = 0;
else
if(failCount >= num)
dist = 0;
```

```
for i = 1:2
dist = dist + (xk(i) - xCur(i))^2;
end
if(sqrt(dist) < threshold)
isEndWhile = 1;
else
xk = xCur;
yk = yCur;
end
directions = GramN(directions, lambda);
lambda = [0 0];
dx = dx0;
failCount = 0;
else
if(abs(dxPrev) < threshold)
isEndWhile = 1;
end
end
end
end
end
[x ,y] = meshgrid(-3:0.01:3, -3:0.01:3);
z = myFunc(x,y);
[C, h] = contour(x, y, z,[0, 1, 10,100,1000]);
clabel(C, h);
shading interp
hold on
plot3(vy(:,1),vy(:,2),vy(:,3),'r','LineWidth',2,'MarkerSize', 5);
hold off
disp(vy);
result = [ xCur yCur ];
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Экспериментальное сравнение алгоритмов Хука–Дживса и Розенброка по числу вычислений целевой функции в процессе оптимизации говорит в пользу алгоритма Розенброка, этот выигрыш растет при увеличении размерности. При этом необходимо учитывать более существенные вычислительные затраты на пересчет системы ортогональных направлений.

Метод Нелдера–Мида

Метод Нелдера–Мида (метод деформируемого многогранника) использует понятие симплекса. Впервые **симплекс – метод** был описан в 1962 году в работе Spendley, Hext и Himsworthin. Нелдер и Мид разработали более эффективную версию симплекс– метода, которая заключается в последовательном перемещении и деформировании симплекса вокруг точки экстремума. Их метод минимизирует

функцию N переменных, используя $N+1$ вершину деформируемого многогранника.

В общем виде стратегию поиска можно представить следующим образом:

given a starting point $N+1$ vertices $\mathbf{x}^0, \ldots, \mathbf{x}^N \in D$
while the stopping criterion is not satisfied
1. Evaluate $f(\mathbf{x})$ at each of the $N+1$ vertices.
2. Define the vertices where the function $f(\mathbf{x})$ assumes the maximum, minimum and second highest values.
3. Discard the maximum vertex.
4. Replaces maximum vertex by a point where $f(\mathbf{x})$ has a lower value. This is achieved by three operations namely reflection, contraction and expansion.

end.

Основные операции **отражение**, **сжатие**, **растяжение** и **редукция** представлены на рис. 4.9 и 4.10. Для каждой из операций вводятся свои коэффициенты: α — для отражения, β — для сжатия, γ — для растяжения.

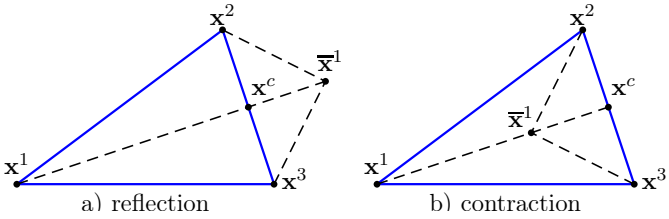


Рис. 4.9. Операции отражения и сжатия

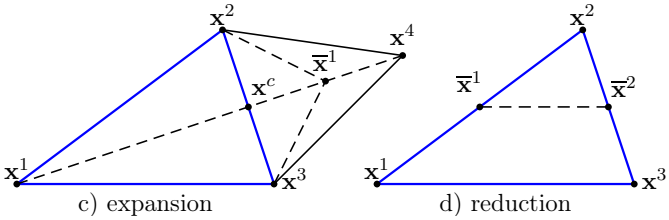


Рис. 4.10. Операции растяжения и редукции

Пусть $f(\mathbf{x}^1) > f(\mathbf{x}^2) > f(\mathbf{x}^3)$, это значит, что $\mathbf{x}^1$ является максимальной вершиной, а $\mathbf{x}^3$ является минимальной вершиной. Обозначим через $\mathbf{x}^c$ центр тяжести всех вершин многогранника за исключением $\mathbf{x}^1$.

Операция отражение (рис. 4.9) создает новую точку $(\overline{\mathbf{x}}^1)$ на продолжении линии, соединяющей вершину $\mathbf{x}^1$ с центром тяжести $\mathbf{x}^c$. Вершина, получаемая в результате отражения, может быть вычислена следующим образом:

$$(\overline{\mathbf{x}}^1) = \mathbf{x}^c + \alpha(\mathbf{x}^c - \mathbf{x}^1).$$

Операция сжатие (рис. 4.9) выполняется при условии, если операция отражения не привела к вершине со значением $f(\mathbf{x})$, меньшим, чем второе по величине, (после

наибольшего) значение $f(\mathbf{x}^1)$, полученное до отражения. Тогда в процессе сжатия получается новая точка $(\overline{\mathbf{x}}^1)$:

$$(\overline{\mathbf{x}}^1) = \mathbf{x}^c + \beta(\mathbf{x}^c - \mathbf{x}^1).$$

Операция растяжение (рис. 4.10) выполняется при условии, если операция отражения дает вершину со значением $f(\mathbf{x})$ меньшим, чем наименьшее значение $f(\mathbf{x})$, полученное до отражения. Тогда в процессе растяжения получается новая точка $\mathbf{x}^4$:

$$\mathbf{x}^4 = \mathbf{x}^c + \gamma(\mathbf{x}^c - \mathbf{x}^1).$$

Редукция (рис. 4.10) выполняется при условии, если операция отражения дает вершину со значением $f(\mathbf{x})$ большим, чем наибольшее значение $f(\mathbf{x})$, полученное до отражения. Тогда в процессе редукции формируется новый симплекс с уменьшенными (как правило, вдвое) сторонами и вершиной в $\mathbf{x}^3$.

Рассмотрим работу операции растяжение (рис. 4.11).

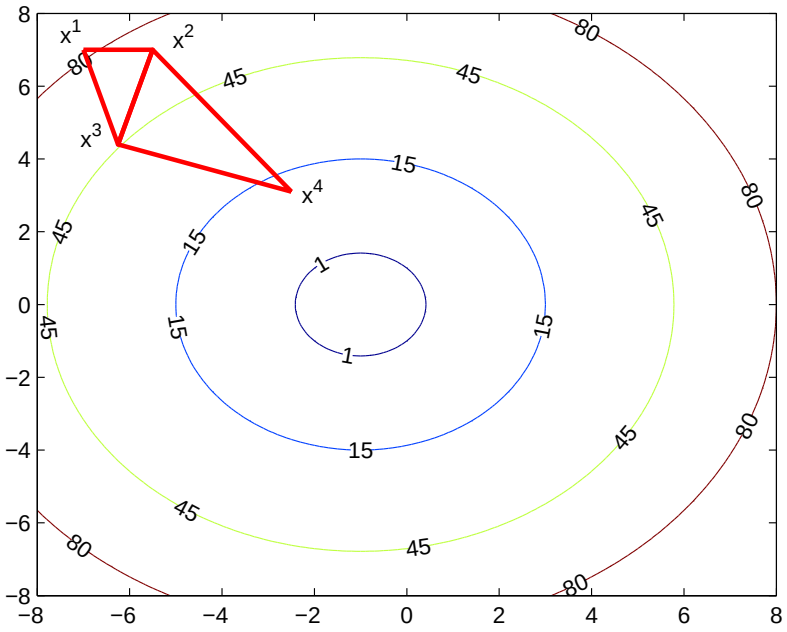


Рис. 4.11. Пример минимизации с растяжением ($\mathbf{x}^1$ – максимальная вершина, $\mathbf{x}^3$ – минимальная вершина)

Операции отражение, сжатие и растяжение позволяют методу Нелдера–Мида, в отличие от жесткого симплекса, адаптироваться к топографии целевой функции, вытягиваясь вдоль длинных наклонных плоскостей, изменяя направление в изогнутых впадинах и сжимаясь в окрестности минимума. Нелдер и Мид рекомендавали использовать следующие значения коэффициентов $\beta = 0.5$ и $\gamma = 2$. Известно, что влияние β на эффективность поиска более заметно, чем влияние of γ.

Метод работает эффективно при $N \leqslant 6$, алгоритм основан на циклическом движении по координатам. Это может привести к вырождению алгоритма в бесконечную последовательность исследующих поисков без поиска по образцу.

В Matlab для минимизации функции нескольких переменных используется функция `fminsearch` . Эта функция ищет минимум безусловной многопараметрической функции, используя метод Нелдера–Мида.

Метод Пауэлла (метод сопряженных направлений)

В **методе Пауэлла** для нахождения минимума функции используются N взаимно **сопряженных направлений**.

Направления поиска $\mathbf{d}_i$, $i = 1, \ldots, N$ называются Q-сопряженными, или просто сопряженными по отношению к положительно определенной квадратной матрице Q, если

$$\mathbf{d}_i Q (\mathbf{d}_j)^T = 0, \quad 0 \leqslant i \neq j \leqslant N - 1.$$

В случае решения задач оптимизации матрица Q представляет собой матрицу Гессе для целевой функции. Минимум N-параметрической квадратичной функции $f(\mathbf{x})$ может быть найден не более чем за N шагов при условии, что поиск ведется вдоль сопряженных относительно матрицы Гессе направлений.

Стратегия метода Пауэлла по генерированию сопряженных направлений основана на следующем свойстве. Если $\mathbf{x}^1$ и $\mathbf{x}^2$ — это любые две точки, а $\mathbf{d}$ — некое направление, при этом x_d^1 соответствует точке с минимальным значением функции $f(\mathbf{x})$ на линии, начинающейся в точке $\mathbf{x}^1$ и проведенной вдоль направления $\mathbf{d}$, а x_d^2 соответствует точке с минимальным значением функции $f(\mathbf{x})$ на линии, начинающейся в точке $\mathbf{x}^2$ и проведенной вдоль направления $\mathbf{d}$, тогда направления $\mathbf{d}$ и $(x_d^1 - x_d^2)$ являются Q-сопряженными. Пауэлл отмечал, что в некоторых случаях поиск может привести к линейно зависимым направлениям поиска. Он указывал, что не следует заменять старое направление поиска на новое, если после этого направления нового набора становятся линейно зависимыми. Поэтому Пауэлл модифицировал свой метод.

Модифицированный метод Пауэлла включает цикл линейных минимизаций. Каждый цикл состоит из следующих шагов (k — это номер цикла): пусть аналогично (4.3) задан стандартный базис в $\mathbb{R}^N$

given a starting point $\mathbf{x}^0 = (x_1^0, \ldots, x_N^0) \in D$, $k = 0$,
while the stopping criterion is not satisfied

1. Minimize f along each of the coordinate directions $\mathbf{d}_i$ (line search) and generating the point $x_1^k, \ldots, x_N^k$.
2. Find the index m corresponding to the direction of the line search which yields the largest function decrease in going from x_{m-1}^k to x_m^k.
3. Denote the direction $\mathbf{d}_p^k$ as a direction from x_N^k to x_0^k (which is the sum of all the line moves) and determine the value of s^k from x_0^k along $\mathbf{d}_p^k$ that minimizes f. Denote this new point by x_0^{k+1}.

4. If

$$\|s^k| < \left(\frac{f(\mathbf{x}_0^k) - f(\mathbf{x}_0^{k+1})}{f(\mathbf{x}_{m-1}^k) - f(\mathbf{x}_m^k)} \right)^{1/2}.$$

then use the same old directions else replace the m-th direction by $\mathbf{d}_p^k$.

5. Begin the next univariate cycle with the directions from step 4.

end.

Анализ результатов работы метода Пауэлла показал, что использование сопряженных направлений более эффективно, чем произвольный выбор направлений для линейного поиска.

Методы случайного поиска

Методы случайного поиска относятся к наиболее простым методам поиска оптимума целевой функции. Однако при этом они могут быть весьма эффективны для минимизации различных функций. Случайность в этих методах чаще всего связана с направлением поиска, которое определяется с помощью случайного вектора единичной длины ξ^k. Базовая стратегия алгоритмов случайного поиска заключается в следующем:

given a starting point $\mathbf{x}^0 = (x_1^0, \ldots, x_N^0) \in D$, s^0,$k = 0$,
while the stopping criterion is not satisfied

1. Sample a new point $\mathbf{y} = \mathbf{x}^k + s^k \xi^k$ in direction ξ^k from the hypersphere with radius s^k surrounding the current point $\mathbf{x}^k$.
2. If $f(\mathbf{y}) < f(\mathbf{x}^k)$, then move to the new position by setting $\mathbf{x}^{k+1} = \mathbf{y}$

end.

Существуют различные варианты методов случайного поиска. Рассмотрим некоторые из них.

Случайный поиск с постоянным шагом. В алгоритме с возвратом при неудачном шаге (в случае, если $f(\mathbf{y}) \geqslant f(\mathbf{x}^k)$) происходит возврат в текущую точку $\mathbf{x}^k$ и поиск продолжается. Если число неудачных шагов из текущей точки достигнет некоторого числа M, то алгоритм останавливается, и $\mathbf{x}^k$ является искомым приближением.

В алгоритме *наилучшей пробы* на текущей итерации при помощи генерирования случайных векторов ξ^k получается M точек, лежащих на гиперсфере радиуса s^k с центром в точке $\mathbf{x}^k$. Среди полученных точек выбирается такая точка $\mathbf{y}$, в которой значение функции наименьшее. Если в выбранной точке значение функции меньше, чем в центре, т.е. $f(\mathbf{y}) < f(\mathbf{x}^k)$, то дальнейший поиск продолжается из этой точки. Иначе алгоритм останавливается, и точка $\mathbf{x}^k$ является искомым приближением.

Модифицированный алгоритм наилучшей пробы. Аналогично предыдущему алгоритму определяется точка $\mathbf{y}$ с наилучшим значением функции. Затем проводят поиск вдоль прямой, проходящей через $\mathbf{x}^k$ и $\mathbf{y}$, и определяют точку $\mathbf{x}^{k+1}$, в которой функция $f(\mathbf{x})$ будет иметь свое минимальное значение.

Случайный поиск с адаптацией шага. В адаптивном методе случайного поиска при неудачном шаге (в случае, если $f(\mathbf{y}) \geqslant f(\mathbf{x}^k)$) происходит возврат в текущую

точку $\mathbf{x}^k$ и поиск продолжается. Если число неудачных шагов из текущей точки достигнет некоторого числа M, то дальнейший поиск продолжается из этой же точки, но с меньшим шагом до тех пор, пока шаг не станет меньше заранее заданной величины ε. Если шаг удачный, то в заданном направлении делается увеличенный шаг. Если при этом значение функции снова меньше, чем в текущей точке $\mathbf{x}^k$, направление считается удачным и дальнейший поиск продолжается из этой точки. Иначе направление считается неудачным и поиск продолжается из точки $\mathbf{x}^k$.

Ниже приведен скрипт Matlab для метода случайного поиска с адаптацией шага.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function[array_x0,array_y0,array_Fz]=adaptive(x0,y0, a,b,N,R,t0,M)
k = 0;j = 1;infinite = 0;array_x0= [];array_y0 = [];array_Fz = [];
while infinite == 0
e1 = randi([-1000,1000])/1000;
e2 = randi([-1000,1000])/1000;
e = sqrt(e1^2+e2^2);
x1 = x0 + t0*(e1/e);
y1 = y0 + t0*(e2/e);
F = Func(x0,y0);
Fy = Func(x1,y1);
if (Fy < F)
z_x = x0 + a*(x1-x0);
z_y = y0 + a*(y1-y0);
Fz = Func(z_x,z_y);
if (Fz < F)
array_Fz = [array_Fz, Fz];
x0 = z_x;
y0 = z_y;
array_x0 = [array_x0, z_x];
array_y0 = [array_y0, z_y];
t0 = a*t0;
k = k + 1;
if (k < N)
j = 1;
else
break;
end
else
if j < M
j = j + 1
else
if t0 <= R
break;
else
t0 = b*t0;
j = 1;
```

```
end
end
end
else
if j < M
j = j + 1
else
if t0 <= R
break;
else
t0 = b*t0;
j = 1;
end
end
end
end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Поиск может быть ускорен, если точки на гиперсфере выбирать случайно, но с ограничениями, чтобы они отклонялись по крайней мере не более, чем на некоторый минимальный угол как по отношению друг к другу, так и к предыдущему направлению поиска. Этот минимальный угол может также меняться в процессе поиска.

Для того, чтобы задать направление в случайном поиске можно использовать направляющие гиперквадраты, гиперсферы или гиперконусы. Стратегия поиска таких алгоритмов будет отличаться от базовой.

Рассмотрим, например, стратегию поиска алгоритма наилучшей пробы с направляющим гиперквадратом (a — размер стороны гиперквадрата, β — коэффициент):

given (random) a starting hypersquare $Q^0 \subset D$ and m points $(\mathbf{x}^0)^1, (\mathbf{x}^0)^2, \ldots, (\mathbf{x}^0)^m \in Q^0$, a^0, β, $k = 0$,
while the stopping criterion is not satisfied
1. Evaluate $f((\mathbf{x}^k)^m)$. Select the best point $(\mathbf{x}^k)^*$ with minimum value of $f(\mathbf{x})$.
2. Make a new hypersquare Q^{k+1}. Point $(\mathbf{x}^k)^*$ is the center of the hypersquare Q^{k+1}, $a^{k+1} = a^k/\beta$.
3. Given(random) a new M points from Q^{k+1}.

end.

Пример работы алгоритма наилучшей пробы с направляющим гиперквадратом приведен на рис. 4.12.

Выше были рассмотрены основные методы нулевого порядка, для которых были разработны скрипты Matlab. Теперь эти скрипты могут быть использованы для углубленного анализа и сравнения различных методов.

Пример 4.1. В примере на рис. 4.13 показан графический интерфейс пользователя (так называемое GUI), разработанный для анализа и изучения свойств метода Нелдера–Мида. (Скрипт GUI приведен ниже).

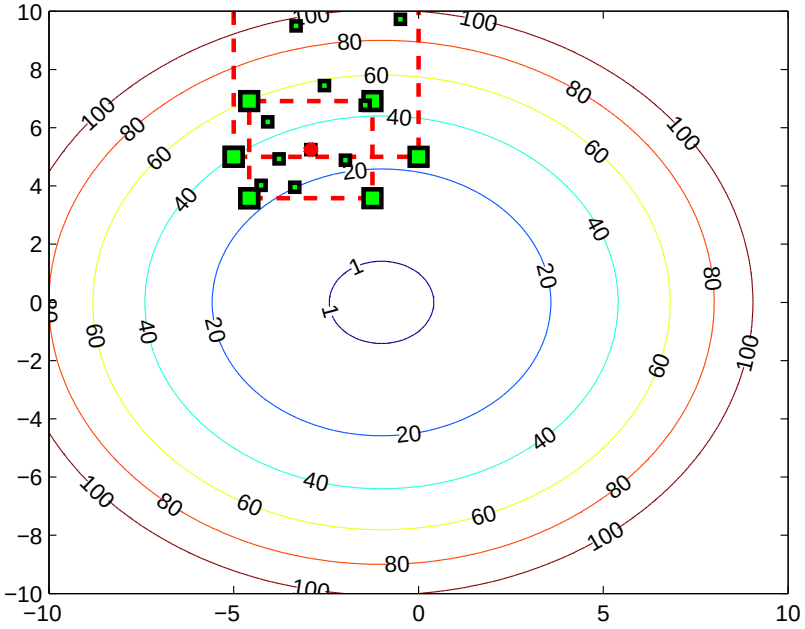


Рис. 4.12. Алгоритм наилучшей пробы с направляющим гиперквадратом (красная точка — лучшая точка начального гиперквадрата и центр следующего гиперквадрата)

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function varargout = Part2(varargin)
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
'gui_Singleton',  gui_Singleton, ...
'gui_OpeningFcn', @Part2_OpeningFcn, ...
'gui_OutputFcn',  @Part2_OutputFcn, ...
'gui_LayoutFcn',  [] , ...
'gui_Callback',   []);
if nargin && ischar(varargin{1})
gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
[varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
gui_mainfcn(gui_State, varargin{:});
end

function Part2_OpeningFcn(hObject, eventdata, handles, varargin)
handles.output = hObject;
guidata(hObject, handles);
```

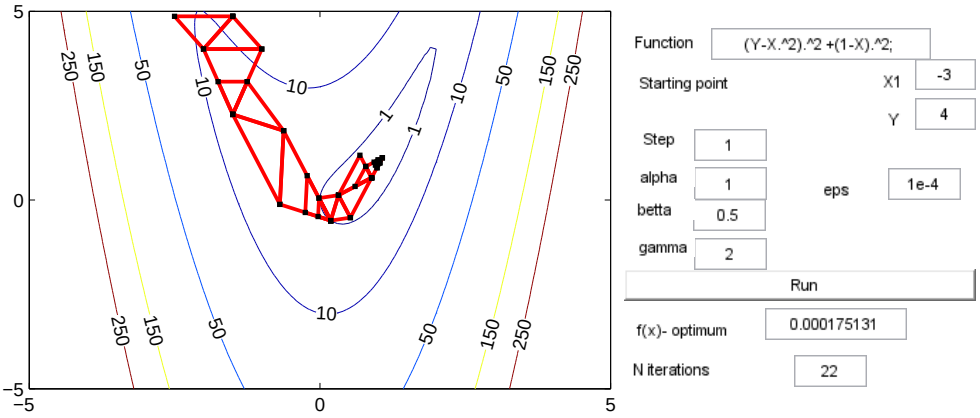


Рис. 4.13. GUI для анализа метода Нелдера–Мида

```
function varargout = Part2_OutputFcn(hObject, eventdata, handles)
varargout{1} = handles.output;

function pushbutton1_Callback(hObject, eventdata, handles)
formula = get(handles.edit1, 'String');
f2=str2func(['@(X,Y)' formula]);
x(1)=str2double(get(handles.edit6, 'String'));
y(1)=str2double(get(handles.edit7, 'String'));
alpha=str2double(get(handles.edit3, 'String'));
beta=str2double(get(handles.edit4, 'String'));
gamma=str2double(get(handles.edit5, 'String'));
shag=str2double(get(handles.edit2, 'String'));
accuracy=str2double(get(handles.edit10, 'String'));
[min,count]=
         NelderMid(f2,x,y,shag,accuracy,alpha,beta,gamma,handles.axes1);
set(handles.edit8, 'String',min);
set(handles.edit9, 'String',count);

function [min,count] = ...
 NelderMid(f2,x,y,shag,accuracy,alpha,beta,gamma,handl)

cla(handl);
[X,Y]=meshgrid(-5:0.1:5);
f=f2(X,Y);
[CMatr,h]=contour(X,Y,f,[1,10,50,100]);
clabel(CMatr,h)
hold on;

%shag =1.2;
```

```
%x2 = random('Poisson',1:1,1,2);
%x(1)=-10;
%y(1)=10;
x(2)=x(1)+shag;
y(2)=y(1);
y(3)=y(1)+sqrt(3)*shag/2;
x(3)=x(1)+shag/2;
%x(1)=0.5;
%y(1)=1;
%x(2)=0;
%y(2)=0.5;
%x(3)=1;
%y(3)=0.5;
xx=x;
yy=y;
count = 3;
length=5000;
for i=1:length
max=f2(x(1),y(1));
min=f2(x(1),y(1));
kandidat=1;
for s= 2:3
if (max <= f2(x(s),y(s)))
max = f2(x(s),y(s));
kandidat=s;
end;
end;
for s= 1:3
if s ~= kandidat
if (min >= f2(x(s),y(s)))
min = f2(x(s),y(s));
kandmin=s;
end;
end;
end
for s= 1:3
if ((s ~= kandidat)&&(s~=kandmin))
middle = f2(x(s),y(s));
end
end

count = count+1;
switch kandidat
case {1}
[x(4),y(4)]=median(x(2),y(2),x(3),y(3),x(1),y(1),alpha);
case {2}
```

```
[x(4),y(4)]=median(x(1),y(1),x(3),y(3),x(2),y(2),alpha);
case {3}
[x(4),y(4)]=median(x(1),y(1),x(2),y(2),x(3),y(3),alpha);
end

if ((f2(x(4),y(4))>middle )&&(f2(x(4),y(4)) <=max))
switch kandidat
case {1}
[x(kandidat),y(kandidat)] = ...
sgat(x(2),y(2),x(3),y(3),x(kandidat),y(kandidat),beta);
case {2}
[x(kandidat),y(kandidat)] = ...
sgat(x(1),y(1),x(3),y(3),x(kandidat),y(kandidat),beta);
case {3}
[x(kandidat),y(kandidat)] = ...
sgat(x(1),y(1),x(2),y(2),x(kandidat),y(kandidat),beta);
end
end
if f2(x(4),y(4))>max
switch kandmin
case {1}
[x(2),y(2),x(3),y(3)]=redukt(x(2),y(2),x(3),y(3),x(1),y(1));
case {2}
[x(1),y(1),x(3),y(3)]=redukt(x(1),y(1),x(3),y(3),x(2),y(2));
case {3}
[x(1),y(1),x(2),y(2)]=redukt(x(1),y(1),x(2),y(2),x(3),y(3));   end

else
if f2(x(4),y(4)) < min
switch kandidat
case {1}
[x(kandidat),y(kandidat)]=rast9j(x(2),y(2),x(3),y(3),x(4),y(4),gamma);
case {2}
[x(kandidat),y(kandidat)]=rast9j(x(1),y(1),x(3),y(3),x(4),y(4),gamma);
case {3}
[x(kandidat),y(kandidat)]=rast9j(x(1),y(1),x(2),y(2),x(4),y(4),gamma);
end
end
if f2(x(kandidat),y(kandidat))>min
x(kandidat)=x(4); y(kandidat)=y(4);
end
end;


x1=x; y1=y;
x1(4)=x1(1); y1(4)=y1(1);
```

```
plot(x1,y1,'--rs','LineWidth',2,...
'MarkerEdgeColor','k',...
'MarkerFaceColor','g',...
'MarkerSize',10);
yy(count)=y(kandidat);
xx(count)=x(kandidat);
if abs(max-min)<accuracy*10
%if(min<0.0000000000000001)
length = 1;
count=count-3;
break;
end;
end;

function [x2,y2] = median( x , y , x1 , y1 ,x3,y3,alpha)
x2=(x1-x)/2+x;
y2=(y1-y)/2+y;
if (x3 <= x2)
x2=x2+abs((x2-x3)*alpha);
else
x2=x2-abs((x2-x3)*alpha);
end;
if (y3 <= y2)
y2=y2+abs((y2-y3)*alpha);
else
y2=y2-abs((y2-y3)*alpha);
end;

function [x2,y2] = rast9j( x , y , x1 , y1 ,x3,y3,gamma)
x2=(x1-x)/2+x;
y2=(y1-y)/2+y;
gamma=gamma-1;
if (x3 <= x2)
x2=x3-abs((x2-x3)*gamma);
else
x2=x3+abs((x2-x3)*gamma);
end
if (y3 <= y2)
y2=y3-abs((y2-y3)*gamma);
else
y2=y3+abs((y2-y3)*gamma);
end

function [xx1,yy1,xx2,yy2] = redukt(x,y,x1,y1,x3,y3)
beta=2;
if(x3<=x)
```

```
xx1=x3+abs((x3-x)/beta);
else
xx1=x3-abs((x3-x)/beta);
end
if(y3<=y)
yy1=y3+abs((y3-y)/beta);
else
yy1=y3-abs((y3-y)/beta);
end

if(x3<=x1)
xx2=x3+abs((x3-x1)/beta);
else
xx2=x3-abs((x3-x1)/beta);
end
if(y3<=y1)
yy2=y3+abs((y3-y1)/beta);
else
yy2=y3-abs((y3-y1)/beta);
end

function [x2,y2] = sgat( x , y , x1 , y1 ,x3,y3,beta)
x2=(x1-x)/2+x;
y2=(y1-y)/2+y;
if (x3 <= x2)
x2=x2-(x2-x3)*beta;
else
x2=x3-abs((x2-x3)*beta);
end;
if (y3 <= y2)
y2=y2-(y2-y3)*beta;
else
y2=y3-abs((y2-y3)*beta);
end;

function edit1_Callback(hObject, eventdata, handles)
function edit1_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),...
get(0,'defaultUicontrolBackgroundColor'))
set(hObject,'BackgroundColor','white');
end


function edit2_Callback(hObject, eventdata, handles)
function edit2_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),...
```

```
get(0,'defaultUicontrolBackgroundColor'))
set(hObject,'BackgroundColor','white');
end

function edit3_Callback(hObject, eventdata, handles)
function edit3_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),...
get(0,'defaultUicontrolBackgroundColor'))
set(hObject,'BackgroundColor','white');
end

function edit4_Callback(hObject, eventdata, handles)
function edit4_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),...
get(0,'defaultUicontrolBackgroundColor'))
set(hObject,'BackgroundColor','white');
end

function edit5_Callback(hObject, eventdata, handles)
function edit5_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),...
get(0,'defaultUicontrolBackgroundColor'))
set(hObject,'BackgroundColor','white');
end

function edit6_Callback(hObject, eventdata, handles)
function edit6_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),...
get(0,'defaultUicontrolBackgroundColor'))
set(hObject,'BackgroundColor','white');
end

function edit7_Callback(hObject, eventdata, handles)
function edit7_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), ...
get(0,'defaultUicontrolBackgroundColor'))
set(hObject,'BackgroundColor','white');
end

function edit8_Callback(hObject, eventdata, handles)
function edit8_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),...
get(0,'defaultUicontrolBackgroundColor'))
set(hObject,'BackgroundColor','white');
end
```

```
function edit9_Callback(hObject, eventdata, handles)
function edit9_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),...
get(0,'defaultUicontrolBackgroundColor'))
set(hObject,'BackgroundColor','white');
end

function edit10_Callback(hObject, eventdata, handles)
% hObject    handle to edit10 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit10 as text
% str2double(get(hObject,'String')) returns contents of edit10 as double

% --- Executes during object creation, after setting all properties.
function edit10_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit10 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), ...
get(0,'defaultUicontrolBackgroundColor'))
set(hObject,'BackgroundColor','white');
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Упражнение 4.1. 1. Используя приведенные выше скрипты, разработать GUI для сравнения эффективности работы различных методов линейного поиска в методе наискорейшего спуска.

2. Используя приведенные выше скрипты, разработать GUI для сравнения эффективности работы различных методов линейного поиска на этапах исследующего поиска и поиска по образцу в методе Хука–Дживса.

3. Используя приведенные выше скрипты, разработать GUI для анализа значений коэффициентов α и β в методе Розенброка.

4. Используя приведенные выше скрипты, разработать GUI для сравнения эффективности работы различных методов линейного поиска в методе Пауэлла.

5. Используя приведенные выше скрипты, разработать GUI для сравнения эффективности работы различных методов случайного поиска. Проанализируйте влияние различных параметров поиска на результат.

4.2.3. Методы первого порядка

Методы первого порядка для поиска оптимума функции используют градиент этой функции.

Пусть функция f, ограниченная снизу на множестве $\mathbb{R}^N$, является дифференцируемой на множестве $\mathbb{R}^N$. Градиент — это вектор в точке $\mathbf{x}$, направление которого указывает направление (локальное) наибольшего возрастания функции f. Направление, противоположное направлению градиента, является направлением для минимизации или направлением наискорейшего спуска:

$$\mathbf{d}^k = -\nabla f(\mathbf{x}^k).$$

Тогда алгоритмы поиска оптимальной точки $\mathbf{x}^*$ с минимальным значением функции могут быть построены на основании информации, по крайней мере, о градиенте этой функции. Такие методы называются градиентными. Старейший из известных методов минимизации функции N переменных — метод наискорейшего спуска был предложен известным французским математиком Коши для решения задачи минимизации без ограничений. Все **градиентные методы** основаны на общей итерационной стратегии:

given a starting point $\mathbf{x}^0 = (x_1^0, \ldots, x_N^0) \in D$ and a step s^k, $k = 0$,
while the stopping criterion is not satisfied
1. Evaluate $\nabla f(\mathbf{x}^k)$.
2. Compute the next point $\mathbf{x}^{k+1} = \mathbf{x}^k - s^k \nabla f(\mathbf{x}^k)$.

end.

Градиентные методы безусловной оптимизации различаются способом построения последовательности точек $\mathbf{x}^k$ и способом выбора размера шага s^k.

Процесс построения последовательности точек $\mathbf{x}^k$ может быть одним из следующих: либо для нахождения точки $\mathbf{x}^k$ значения координат вектора этой точки определяют последовательно, либо все координаты вектора точки $\mathbf{x}^k$ вычисляют одновременно.

Наиболее распространенными способами выбора размера шага являются следующие:

1) Величина шага выбирается фиксированной – постоянный шаг, который задается до начала решения задачи. Используется в том случае, когда скорость решения не существенна.

2) Величина шага меняется от итерации к итерации – переменный (адаптивный) шаг.

3) Размер шага выбирается таким образом, чтобы минимизировать значение функции $f(\mathbf{x})$ вдоль выбранного направления с помощью одного из методов однопараметрической минимизации.

Рассмотрим кратко особенности стратегий поиска, применяемых в наиболее известных методах первого порядка.

Градиентный спуск с постоянным шагом

В этом методе величина шага s^k задается пользователем и остается постоянной до тех пор, пока функция убывает в точках последовательности $\mathbf{x}^k$. На каждой итерации метод «приближается» к решению, вычисляя новое значение $\mathbf{x}^k$ в соответствии с формулой:

$$\mathbf{x}^{k+1} = \mathbf{x}^k - s^k \nabla f(\mathbf{x}^k).$$

Выбрать подходящий размер шага достаточно сложно. Как видно из рис. 4.14, слишком маленький шаг s^k может вызвать очень медленную сходимость алгоритма. С другой стороны, слишком большой шаг s^k может привести к тому, что алгоритм не отыщет минимум.

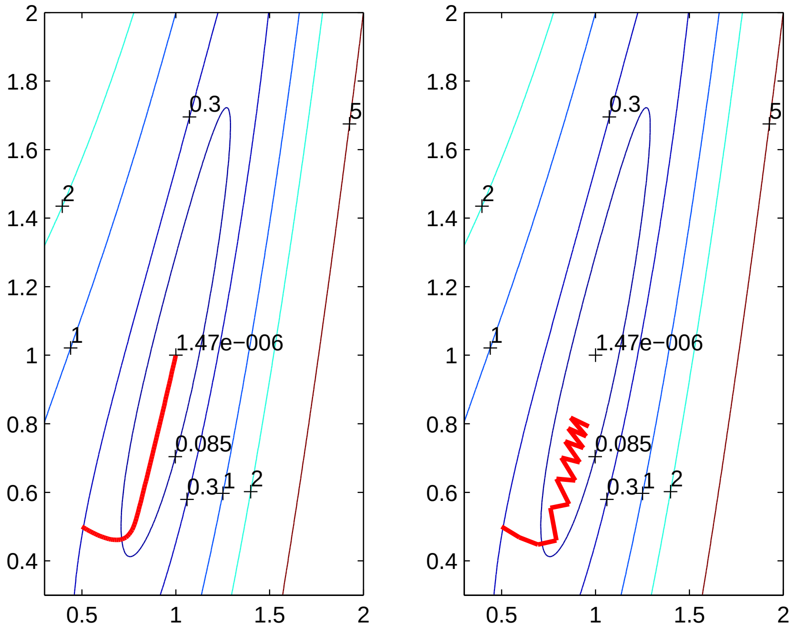


Рис. 4.14. Градиентный спуск с маленьким (левый рисунок) и большим шагом (правый рисунок)

Скрипт Matlab для **градиентного спуска с постоянным шагом** приведен ниже.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
k=0;d = 0.000001; step = 0.01; % fixed step
x1=str2num(input(' Enter coordinate X1 of starting point  =','s'));
x2=str2num(input(' Enter coordinate X2 of starting point  =','s'));
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
p0 = [x1; x2];
x1trace=[p0(1)];
x2trace=[p0(2)];
grad = differ(p0);
funcgrad = func2(p0(1),p0(2));
while k ~= 1
p1 = p0 - grad * step;
funcNewgrad = func2(p1(1),p1(2));
```

```
if funcNewgrad < funcgrad
p0=p1;
grad = differ(p0);
funcgrad = funcNewgrad;
x1trace=[x1trace, p0(1)];
x2trace=[x2trace, p0(2)];
else
k=1;
end
end
[X,Y]=meshgrid(0.3:0.01:2);
Z = func2(X,Y);
[C, h] = contour(X, Y, Z,[0,0.085,0.3,1,2,5]);
clabel(C, h);
frezult=func2(x1trace, x2trace)
hold on;
plot3(x1trace, x2trace,frezult,'r','LineWidth',2);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Таким образом, целесообразно, чтобы величина шага s^k была переменной и существовала возможность ее уменьшения при приближении к точке минимума.

Градиентный спуск с адаптацией шага

В методе *Градиентного спуска с делением шага* на каждой итерации проверяется, уменьшился ли модуль вектора-градиента. Если модуль вектора-градиента не уменьшился, величина шага s^k делится, например, пополам, и итерация повторяется заново (рис. 4.15).

В методе *Линейного поиска с возвратом* (Backtracking line search) величина шага также выбирается адаптивно. Neculai и Burachik утверждали, что этот метод является более простым и эффективным по сравнению с другими методами. Метод включает следующие основные этапы. Во-первых, задается параметр $0 < \beta < 1$, затем на каждой итерации, начиная с $s^k = 1$, и пока выполняется условие

$$f(\mathbf{x}) - \nabla f(\mathbf{x}) > f(\mathbf{x}) - \frac{s^k}{2} \|\nabla f(\mathbf{x})\|^2$$

величина шага изменяется следующим образом: $s^{k+1} = \beta s^k$. Линейный поиск с возвратом прост и дает хорошие результаты.

Еще один из возможных методов контроля s^k предполагает установление некоторого критерия для s^k, основанного на использовании угла между последовательными векторами шагов в процессе минимизации. Например, если этот угол становится меньше, чем некоторая заданная величина, то s^k должен быть умножен на некоторую заранее определенную константу, если угол становится больше, то s^k надо разделить на эту же константу.

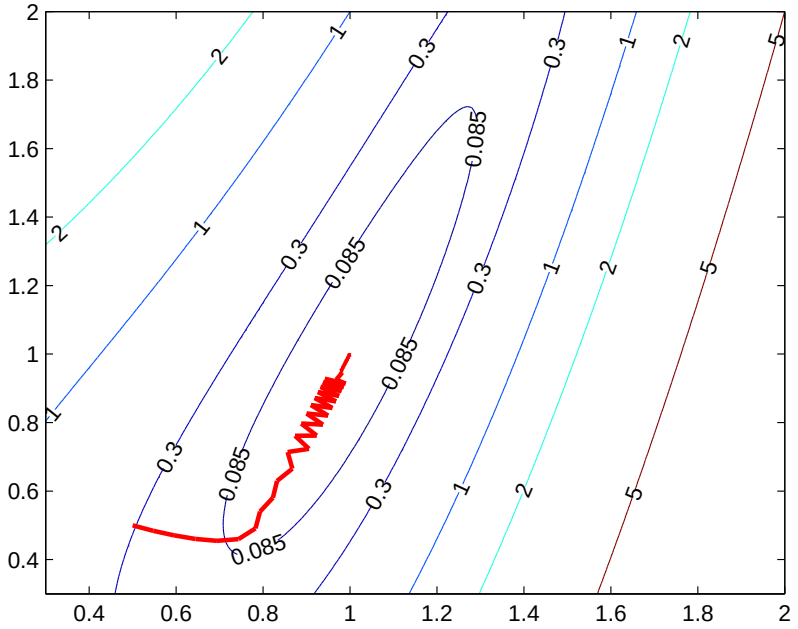


Рис. 4.15. Метод градиентного спуска с делением шага

Метод наискорейшего спуска

В этом методе s^k выбирается так, чтобы минимизировать функцию $f(\mathbf{x})$ в направлении спуска. Т.е. движение вдоль антиградиента происходит до тех пор, пока значение функции $f(\mathbf{x})$ убывает. Величина шага s^k определяется с помощью линейного поиска:

$$s^k = \arg\min_s f\left(\mathbf{x}^k + s\mathbf{d}^k\right)$$

В этом случае градиент в точке очередного приближения будет ортогонален направлению предыдущего спуска. Это означает, что движение от точки к точке будет происходить по взаимно-ортогональным направлениям (рис. 4.16).

Наискорейший спуск может закончиться в любой стационарной точке. Поэтому необходимо определить, является ли данная точка локальным минимумом или седловой точкой. В последнем случае необходимо использовать какой-нибудь неградиентный метод, чтобы выйти из нее, после этого можно продолжать минимизацию.

Метод наискорейшего спуска обладает в общем случае очень медленной сходимостью. Дело в том, что спуск является «наискорейшим» в локальном смысле. Если гиперпространство поиска сильно вытянуто («овражная» функция), то антиградиент направлен почти ортогонально дну «оврага», т.е. наилучшему направлению достижения минимума.

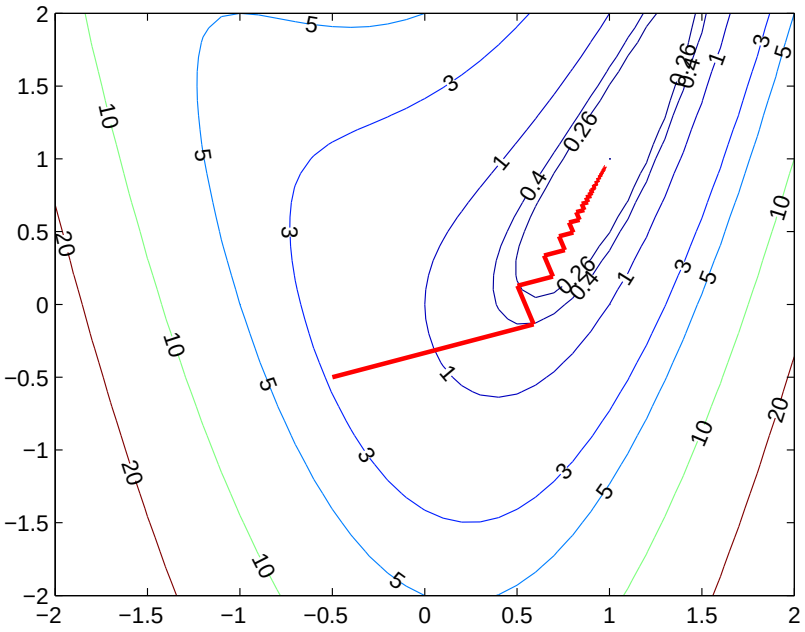


Рис. 4.16. Пример работы метода наискорейшего спуска для второй тестовой функции

Метод Гаусса–Зейделя

В **методе Гаусса–Зейделя** для определения положения точки $\mathbf{x}^k$ вектор значений ее координат определяется последовательно. Так, только одна проекция точки, имеющая номер i, меняется в течении одной итерации с номером $k+1$ и фиксированным i:

$$x_i^{k+1} = \arg\min_{x_i} f(x_1^{k+1}, \ldots, x_{i-1}^{k+1}, x_i, x_{i+1}^{k}, \ldots, x_N^{k})$$

В этом случае движение выполняется вдоль анти-градиента

$$-\nabla f_i(x_1^{k+1}, \ldots, x_{i-1}^{k+1}, x_i^{k}, x_{i+1}^{k}, \ldots, x_N^{k}).$$

Все N координат x_i^{k+1}, $i = 1, \ldots, N$ точки $\mathbf{x}^{k+1}$ изменяются в течение всего этапа с номером $k+1$, т.е., начиная с $i = 1$ и заканчивая $i = N$. После этого точка берется за начальную точку для вычисления в $(k+2)$-м этапе.

Известны случаи, когда метод Гаусса–Зейделя не сходится в том смысле, что он может создавать такую последовательность точек, которые не являются точками решения задачи.

Скрипт Matlab для метода Гаусса-Зейделя приведен ниже.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clear all; clc;close all;
x1=str2num(input(' Enter coordinate X1 of starting point  =','s'));
x2=str2num(input(' Enter coordinate X1 of starting point  =','s'));
```

```
e1 = 0.001;e2 = 0.0015;M = 1000000;j=1;n=2;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
p=[x1;x2];x1trace = [p(1)]; x2trace = [p(2)];
pointresult=[0;0];
temp_func =@ func1;
temp_func(x1,x2)
syms x1 x2
graid =char( diff(temp_func(x1,x2),'x1'));
graid2 =char( diff(temp_func(x1,x2),'x2'));
edv=[1;0];
edv2=[0;1];
funExeqCount=0;
tableresult=[0 0 0 0 0 0 0];
while j < M
x1=p(1); x2=p(2);
gradx1 = eval(graid);
gradx2 = eval(graid2);
deltaF=[gradx1; gradx2];
sizedeltaF=abs(sqrt((deltaF(1))^2+(deltaF(2))^2));
if(sizedeltaF>e1)
if(gradx1~=0)
syms t0
temp = p - (t0*gradx1*edv);
tempF2=temp_func(temp(1),temp(2));
tempfort0 = char( diff(tempF2,'t0'));
res = (solve(tempfort0));
resx1=sym2poly(res);
pointx1=p-resx1(1)*gradx1*edv;
temppoint=pointx1-p;
pointdistance=sqrt((temppoint(1))^2+(temppoint(2))^2);
funcx0=temp_func(p(1),p(2));
funcx1=temp_func(pointx1(1),pointx1(2));
funExeqCount=funExeqCount+2;
if(pointdistance<e2 & (funcx1-funcx0)<e2)
j=M;
x1trace=[x1trace,pointx1(1)];
x2trace=[x2trace,pointx1(2)];
pointrezult=pointx1;
else
x1trace=[x1trace,pointx1(1)];
x2trace=[x2trace,pointx1(2)];
p=pointx1;
end
end
j
if (j<M)
```

```
if(gradx2~=0)
syms t0
temp = p - (t0*gradx2*edv2);
tempF2=temp_func(temp(1),temp(2));
funExeqCount=funExeqCount+1;
tempfort0 = ( diff(tempF2,'t0'));
res = (solve(tempfort0));
resx2=sym2poly(res);
pointx1=p-resx2(1)*gradx2*edv2;
temppoint=pointx1-p;
pointdistance=sqrt((temppoint(1))^2+(temppoint(2))^2);
funcx0=temp_func(p(1),p(2));
funcx1=temp_func(pointx1(1),pointx1(2));
funExeqCount=funExeqCount+2;
if(pointdistance<e2 & (funcx1-funcx0)<e2)
j=M;
x1trace=[x1trace,pointx1(1)];
x2trace=[x2trace,pointx1(2)];
pointrezult=pointx1;
else
x1trace=[x1trace,pointx1(1)];
x2trace=[x2trace,pointx1(2)];
p=pointx1;
end
end
end
else
pointrezult = [x1,x2];
j=M;
end
tableresult=[tableresult;
j funExeqCount real(funcx1) real(pointx1(1)) real(pointx1(2))
real(gradx1) real(gradx2)];
j=j+1;
end
pointrezult
temp_func(pointrezult(1),pointrezult(2))
[X,Y]=meshgrid(-2:0.01:2);
f=temp_func(X,Y);
tableresult
[cMatr,h]=contour(X,Y,f,[0,0.12,0.33,0.76,2,5,10,20]);
clabel(cMatr,h)
frezult=temp_func(x1trace,x2trace);
hold on
plot3(x1trace,x2trace,frezult,'R','LineWidth',2);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Результаты работы метода Гаусса–Зейделя приведены на рис. 4.17.

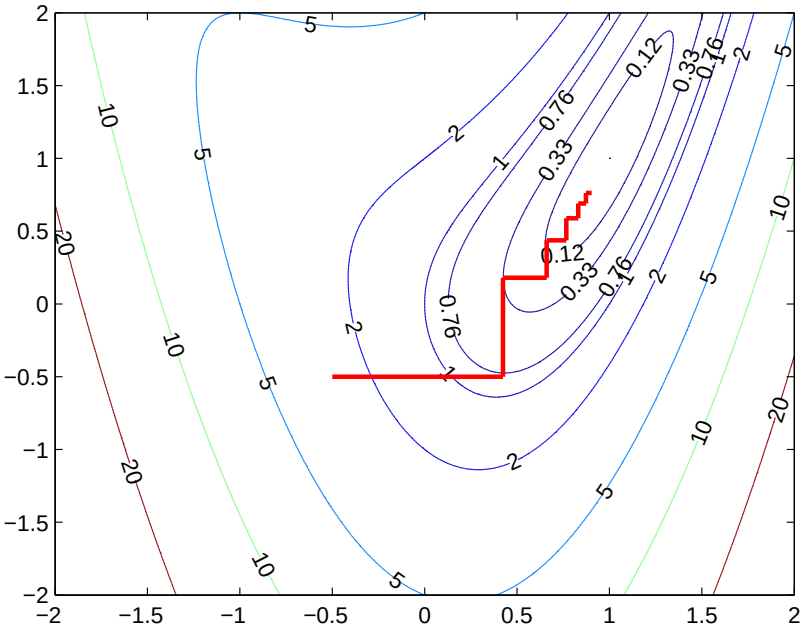


Рис. 4.17. Результаты работы метода Гаусса–Зейделя

Методы сопряженных градиентов

Метод сопряженных градиентов был предложен Флетчером и Ривсом (1964). В этом методе строится последовательность направлений поиска $\mathbf{d}^k$, которые являются линейной комбинацией текущего градиента и ранее найденного направления. Весовые коэффициенты β^k выбираются так, чтобы сделать направления поиска сопряженными.

Рассмотрим стратегию поиска метода сопряженных градиентов.

given a starting point $\mathbf{x}^0 = (x_1^0, \ldots, x_N^0) \in D$, $k = 0$,
while the stopping criterion is not satisfied

1. Evaluate $\nabla f(\mathbf{x}^k)$.
2. Compute the next point $\mathbf{x}^{k+1} = \mathbf{x}^k + s^k \mathbf{d}^k$. Where the step size s^k is determined by a line search:

$$s^k = \arg\min_s f(\mathbf{x}^k + s\mathbf{d}^k)$$

and

$$\mathbf{d}^0 = -\nabla f(\mathbf{x}^0), \quad \mathbf{d}^k = -\nabla f(\mathbf{x}^k) + \beta^{k-1}\mathbf{d}^{k-1},$$

$$\beta^{k-1} = \frac{\left\|\nabla f(\mathbf{x}^k)\right\|^2}{\left\|\nabla f(\mathbf{x}^{k-1})\right\|^2}$$

end

Одним из преимуществ метода сопряженных градиентов является то, что он позволяет сэкономить память на хранение информации на каждом этапе вычислений. Это преимущество возникает из-за того, что в данном методе от матрицы требуется только возможность умножать ее на вектор, что позволяет использовать специальные форматы хранения матрицы, такие как разряженный. Еще одно преимущество заключается в том, что в этом методе квадратичная функция минимизируется после N итераций ($k = N$). Для неквадратичной функции точка $\mathbf{x}^{N+1}$ переопределяется как $\mathbf{x}^0$, и процедура поиска повторяется заново. Следует также учитывать, что метод сопряженных градиентов (алгоритм Флетчера-Ривса) зависит от точности одномерного поиска.

Траектория поиска минимума для 3 -ей тестовой функции методом сопряженных градиентов показана на рис. 4.18.

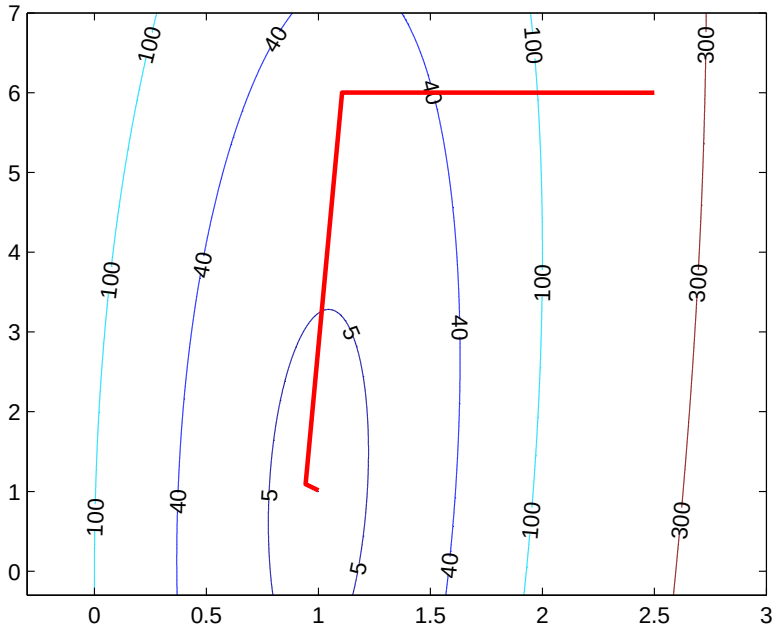


Рис. 4.18. Траектория поиска минимума для 3 -ей тестовой функции методом сопряженных градиентов

При минимизации неквадратичной функции метод сопряженных градиентов не является конечным. При этом, точное вычисление s^k на каждом шаге возможно лишь в редких случаях. Поэтому накопление погрешностей приводит к нарушению не только перпендикулярности градиентов, но и сопряженности направлений. Чтобы модифицировать метод сопряженных градиентов для решения неквадратичных функций можно иначе вычислить коэффициент β^k. Если целевая функция квадратичная и строго выпуклая, то все варианты вычисления коэффициента β^k дают одинаковый результат. Если минимизируется произвольная функция, то каждому способу вычисления коэффициента β^k соответствует своя модификация метода сопряженных градиентов. Для неквадратичных функций, как правило, используется

алгоритм Полака-Рибьера (Полака-Райбера), в котором величина коэффициента β^k вычисляется следующим образом:

$$\beta^{k-1} = \begin{cases} \dfrac{\left(\nabla f(\mathbf{x}^k), \left(\nabla f(\mathbf{x}^k) - \nabla f(\mathbf{x}^{k-1})\right)\right)}{\left\|\nabla f(\mathbf{x}^{k-1})\right\|^2}, & \text{if} \quad k \notin J, \\ 0, & \text{if} \quad k \in J. \end{cases}$$

где $J = \{in : i = 0, 1, \ldots\}$.

В отличие от метода Флетчера–Ривса, в методе Полака– Рибьера предусматривается использование итерации наискорейшего грпдиентного спуска через каждые N шагов с заменой точки $\mathbf{x}^0$ на точку $\mathbf{x}^{N+1}$.

На практике метод Полака–Рибьера работает лучше, чем метод Флетчера–Ривса. При этом в качестве критериев останова для этого метода наиболее распространены следующие: «норма градиента становится меньше некоторого порога» и «значение функции в течение нескольких последовательных итераций практически не изменилось».

Методы сопряженных градиентов являются одними из наиболее эффективных для решения задач оптимизации. Однако они чувствительны к погрешностям, возникающим в процессе вычислений. При большом числе переменных погрешность может настолько вырасти, что процесс поиска оптимума даже квадратичной функции придется повторять более, чем N раз.

Упражнение 4.2. 1. Используя представленные ранее скрипты, разработать GUI для изучения влияния размера шага на сходимость метода градиентного спуска с постоянным шагом.

2. Используя представленные ранее скрипты, разработать GUI для сравнения эффективности поиска минимума функций методами градиентного спуска с делением шага и линейного поиска с возвратом.

3. Используя представленные ранее скрипты, разработать GUI для сравнения эффективности различных методов расчета коэффициента β в методах сопряженных градиентов.

4.2.4. Методы второго порядка

Старейшим методом второго порядка, который используется для минимизации функции, является **метод Ньютона**. В этом методе выполняется квадратичная аппроксимация функции $f(\mathbf{x})$ в каждой точке $\mathbf{x}^k$

$$f(\mathbf{x}^{k+1}) \approx f(\mathbf{x}^k) + \left(\nabla f(\mathbf{x}^k)\Delta\mathbf{x}^k\right) + \frac{1}{2}(\Delta\mathbf{x}^k)H(\mathbf{x}^k)(\Delta\mathbf{x}^k)^T,$$

где $H(\mathbf{x}^k)$ — матрица Гессе, представляющая собой квадратную матрицу вторых частных производных функции f в точке $\mathbf{x}^k$ (смотри (4.2)).

Для определения направления поиска, т.е. направления, в котором значение $f(\mathbf{x}^{k+1})$ будет минимальным, выполняют дифференцирование f по каждой компоненте $\Delta\mathbf{x}$ и приравнивают к нулю полученные выражения. В результате получают соотношение:

$$\Delta \mathbf{x}^k = -(H(\mathbf{x}^k))^{-1}(\nabla f(\mathbf{x}^k)^T,$$

где $(H(\mathbf{x}^k))^{-1}$ — это матрица, обратная матрице Гессе в точке $\mathbf{x}^k$.

Метод Ньютона включает следующие основные шаги

given a starting point $\mathbf{x}^0 = (x_1^0, \ldots, x_N^0) \in D$,$k = 0$,
while the stopping criterion is not satisfied

1. Evaluate $\nabla f(\mathbf{x}^k)$, $H(\mathbf{x}^k)$.
2. Compute Newton direction $\mathbf{d}^k = -(H(\mathbf{x}^k))^{-1}(\nabla f(\mathbf{x}^k)^T$.
3. Compute the next point $\mathbf{x}^{k+1} = \mathbf{x}^k + s^k\mathbf{d}^k$.

end

В классической версии метода Ньютона значение $s^k = 1$ на каждом шаге.

Скрипт Matlab для метода Ньютона приведен ниже.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function varargout = Newton(varargin)
x1=str2num(input(' Enter coordinate X1 of starting point  =','s'));
x2=str2num(input(' Enter coordinate X2 of starting point  =','s'));
point = [x1; x2]; x1mass=[point(1)]; x2mass=[point(2)];
E1=0.01; E2=0.15; M=30; i=1;  kol=0;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
temp_func =@ ftest2; funcMass = [temp_func(point(1),point(2)) ];
syms x1 x2
str1=diff(temp_func(x1,x2),'x1');
str2= diff(temp_func(x1,x2),'x2');
graid =char(str1);
graid2 =char(str2);
graiddouble = char(diff(str1,'x1'));
graiddouble2 = char(diff(str1,'x2'));
graiddouble3 = char(diff(str2,'x1'));
graiddouble4 = char(diff(str2,'x2'));
outTableResult=
[0 kol temp_func(point(1),point(2)) point(1) point(2) 0 0];
while i<M
x1=point(1);
x2=point(2);
gradx1 = eval(graid);
gradx2 = eval(graid2);
dougradx1 = eval(graiddouble);
dougradx2 = eval(graiddouble2);
dougradx3 = eval(graiddouble3);
dougradx4 = eval(graiddouble4);
deltaF=[gradx1; gradx2];
sizedeltaF=abs(sqrt((deltaF(1))^2+(deltaF(2))^2));
if(sizedeltaF>E1)
H1=dougradx1;
```

```
H2=dougradx2;
H3=dougradx3;
H4=dougradx4;
H=[H1 H2;H3 H4]; Hobr=inv(H); determin = det(Hobr);
if(determin>0)
d0=-(Hobr*deltaF);
else
d0=-sizedeltaF;
end
pointnext=point+d0;
temppoint=pointnext-point;
pointdistance=sqrt((temppoint(1))^2+(temppoint(2))^2);
funcx0=temp_func(point(1),point(2));
funcx1=temp_func(pointnext(1),pointnext(2));
kol=kol+2;
x1mass=[x1mass,pointnext(1)];
x2mass=[x2mass,pointnext(2)];
outTableResult=[outTableResult;i kol funcx1 pointnext(1) pointnext(2)
gradx1 gradx2];
if(pointdistance<E2 & (funcx1-funcx0)<E2)
i=M;
pointrezult=pointnext;
else
point=pointnext;
i=i+1;
end
else
pointrezult = point;
i=M;
end
end;
pointrezult
temp_func(pointrezult(1),pointrezult(2))
[X,Y]=meshgrid(-5:0.05:5);
f=temp_func(X,Y);
[cMatr,h]=contour(X,Y,f,[1,2,5,10,100]);
clabel(cMatr,h)
outTableResult
frezult=temp_func(x1mass,x2mass);
hold on
plot3(x1mass,x2mass,frezult,'R','LineWidth',2);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Траектория поиска оптимума методом Ньютона для 2-ой тестовой функции показана на рис. 4.19.

Существенным недостатком классической версии метода Ньютона является зависимость его сходимости от начального приближения для невыпуклых функций.

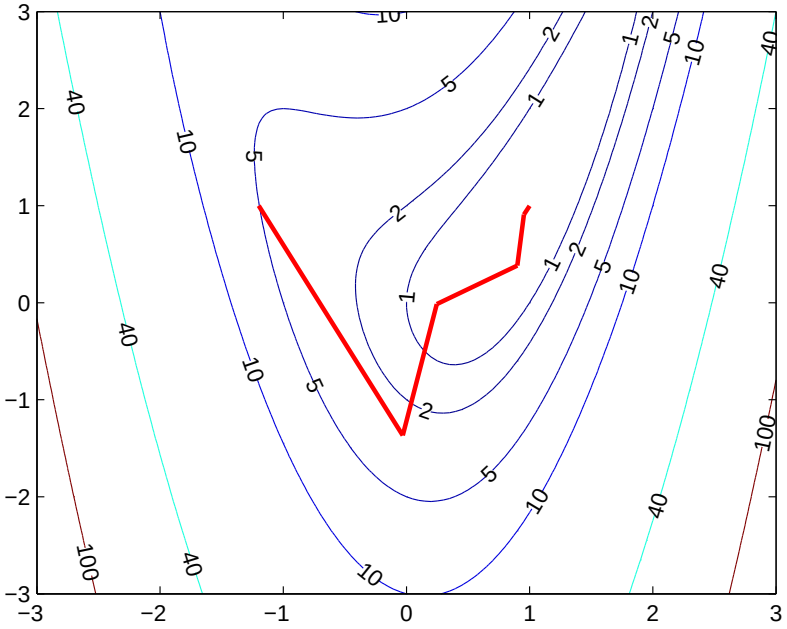


Рис. 4.19. Траектория поиска минимума методом Ньютона для 2-ой тестовой функции)

Если начальная точка находится достаточно далеко от минимума, то метод может расходиться, т.е. при проведении итерации каждая следующая точка будет дальше от точки минимума, чем предыдущая. Существуют два основных класса методов, модифицирующих классический метод Ньютона таким образом, чтобы обеспечить его гарантированную сходимость к локальному минимуму из любой начальной точки: методы доверительной окрестности и методы с линейным поиском.

Методы доверительной окрестности. Термин «доверительная окрестность» («доверительный интервал» или «доверительная область») определяет область (окрестность) вблизи точки $\mathbf{x}^k$, в которой функция f может быть достаточно надежно описана квадратичной моделью. Использование доверительной окрестности позволяет определить длину пробного шага. Пробный шаг рассчитывается посредством минимизации (или приблизительной минимизации) функции в выбранной окрестности. Пауэлл предложил в качестве пробного шага использовать расстояние от точки $\mathbf{x}^k$ до минимума квадратичной модели функции f в направлении наискорейшего спуска. Затем, если Ньютоновский шаг оказывается неуспешным, доверительная окрестность сокращается, и расчет пробного шага повторяется.

Методы Ньютона с линейным поиском. Рассмотрим два основных подхода к модификации классического метода Ньютона:

1. В классической версии метода Ньютона значение $s^k = 1$ на каждом шаге. Вместо этого для определения величины шага предлагается использовать линейный поиск. Старейший и наиболее простой метод вычисления s^k для определения $\Delta\mathbf{x}^k$ состоит в применении однопараметрического линейного поиска. В методе Ньютона–Рафсона шаг s^k определяется исходя из следующего условия:

$$s^k = \arg\min_s f\left(\mathbf{x}^k - s(H(\mathbf{x}^k))^{-1}(\nabla f(\mathbf{x}^k)^T\right).$$

Траектория поиска минимума 2-ой тестовой функции методом Ньютона-Рафсона приведена на рис. 4.20.

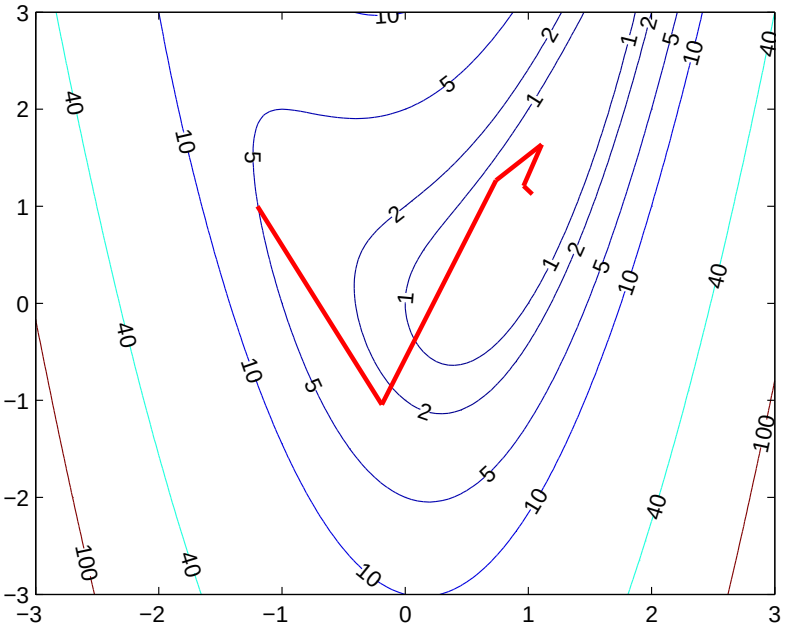


Рис. 4.20. Траектория поиска минимума 2-ой тестовой функции методом Ньютона-Рафсона

2. Если матрица Гессе $H(\mathbf{x}^k)$ не является положительно определенной, ее предлагают заменить положительно определенной матрицей $\widetilde{H}(\mathbf{x})$. Гринштадт, Марквардт, Левенберг и другие авторы предложили проводить дополнительные преобразования, делающие матрицу Гессе положительно определенной на каждом этапе минимизации. Гринштадт построил схему на основе анализа собственных значений, которая обеспечивает положительную определенность приближения. Марквардт и Левенберг предложили другую вычислительную процедуру, добавляющую определенные значения к диагональным элементам матрицы $H(\mathbf{x})$

$$\widetilde{H}(\mathbf{x}) = H(\mathbf{x}) + \beta I$$

где β — это положительная константа, значение которой подбирается таким, чтобы сделать положительно определенной матрицу $H(\mathbf{x})$, I — это единичная матрица. Использование подходящего значения β в сущности «уничтожает» отрицательные и малые собственные значения матрицы, аппроксимирующей матрицу Гессе. При этом, при достаточно большом β произведение βI может «подавить» матрицу $H(\mathbf{x})$, и тогда процесс минимизации приблизится к поиску методом наискорейшего спуска.

Рассмотрим основные шаги метода Марквардта:

given a starting point $\mathbf{x}^0 = (x_1^0, \ldots, x_N^0) \in D$, $k = 0$, $\beta = 10^3$,
while the stopping criterion is not satisfied

1. Evaluate $\nabla f(\mathbf{x}^k)$.
2. Solve for $\mathbf{d}^k$ the following equation : $(H(\mathbf{x}^k) + \beta^k I)\mathbf{d}^k = -\nabla f(\mathbf{x}^k)$.
3. If $(\nabla f(\mathbf{x}^k)^T \mathbf{d}^k) < 0$, go to step 5.
4. Set $\beta^k = 2\beta^k$ and go to step 2.
5. Choose s^k by a line search procedure so that $f(\mathbf{x}^k + s^k \mathbf{d}^k) < f(\mathbf{x}^k)$.
6. If certain conditions are met, reduce β. Go to step 1 with $k = k + 1$.

end

Таким образом, метод Ньютона, с одной стороны, может сходиться с более высоким, чем градиентный метод, порядком, а, с другой стороны, для его сходимости требуются достаточно хорошие начальные приближения. Кроме того, как метод второго порядка, метод Ньютона требует большого объема вычислительной работы, поскольку приходится вычислять вторые производные функции.

В ряде случаев целесообразно комбинированное использование градиентных методов и метода Ньютона. На начальном этапе поиска можно применять какой-либо вариант градиентных методов. Далее, при уменьшении сходимости градиентного метода, можно использовать метод Ньютона.

Пример 4.2. Пример на рис. 4.21 демонстрирует GUI, разработанный для сравнения эффективности работы методов Ньютона и Ньютона–Рафсона.

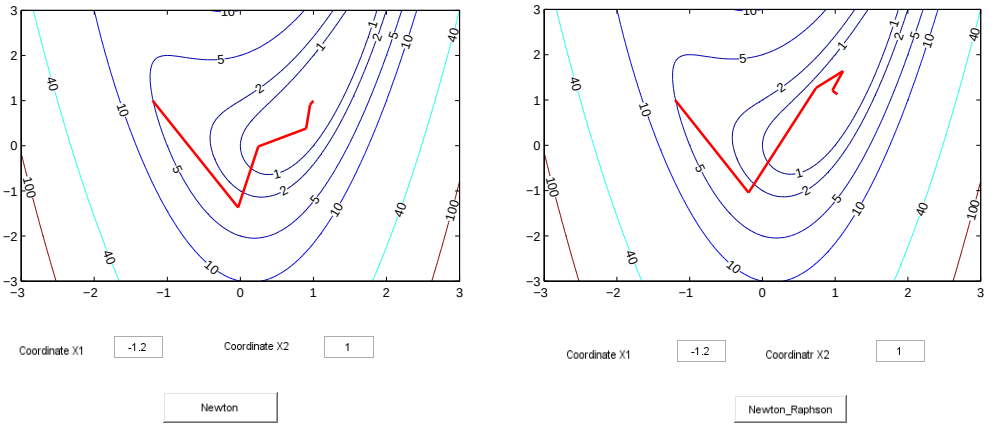


Рис. 4.21. Сравнение эффективности работы методов Ньютона и Ньютона–Рафсона

Упражнение 4.3. 1. Используя скрипты, представленные ранее, разработать GUI для сравнения эффективности работы методов Ньютона–Рафсона и Марквардта.

2. Используя скрипты, представленные ранее, разработать GUI для сравнения эффективности работы методов Ньютона и Марквардта.

4.2.5. Квазиньютоновские методы (методы переменной метрики или градиентные методы с большим шагом)

Квазиньютоновские методы объединяют достоинства градиентных и ньютоновских методов, исключив многие их недостатки. В квазиньютоновских методах отсутствует явное формирование матрицы Гессе ($H(\mathbf{x}^k)$). Матрица Гессе (или обратная к ней матрица) аппроксимируется в процессе поиска, но для этого используются только первые производные. Основой для всех квазиньютоновских методов является то, что обновление матрицы B^k (матрица B представляет собой аппроксимацию матрицы $(H(\mathbf{x})^{-1})$) должно выполняться согласно выражению:

$$B^{k+1}(\Delta\mathbf{g}^k)^T = \Delta\mathbf{x}^k,$$

где

$$\Delta\mathbf{g}^k = \nabla f(\mathbf{x}^{k+1}) - \nabla f(\mathbf{x}^k), \quad \Delta\mathbf{x}^k = \mathbf{x}^{k+1} - \mathbf{x}^k.$$

при этом симметричность и положительная определенность матрицы B^k должны быть сохранены.

На каждой итерации с помощью B^k определяется следующее направление поиска, и матрица B обновляется с учетом вновь полученной информации о кривизне функции. Эта информации иногда называется поправкой и представляется в виде матрицы поправки.

Типовой алгоритм для квазиньютоновских методов включает следующие основные шаги:

given a starting point $\mathbf{x}^0 = (x_1^0, \ldots, x_N^0) \in D$, $B^0 = I$, $k = 0$,
while the stopping criterion is not satisfied

1. Evaluate $\nabla f(\mathbf{x}^k)$.Compute quasi-Newton direction $\mathbf{d}^k = -B^k(\nabla f(\mathbf{x}^k))^T$.
2. Determine step size by a line search: $s^k = \arg\min_s f(\mathbf{x}^k + s\mathbf{d}^k)$.
3. Compute the next point $\mathbf{x}^{k+1} = \mathbf{x}^k + s^k\mathbf{d}^k$.
4. Compute B^{k+1}.

end

Среди наиболее популярных подходов, используемых для обновления матрицы B^{k+1}, можно выделить следующие:

Метод Бройдена

$$B^{k+1} = B^k + \frac{\left(\Delta\mathbf{x}^k - B^k(\Delta\mathbf{g}^k)^T\right)\left(\Delta\mathbf{x}^k - B^k(\Delta\mathbf{g}^k)^T\right)^T}{\left(\Delta\mathbf{x}^k - B^k(\Delta\mathbf{g}^k)^T\right)^T(\Delta\mathbf{g}^k)^T}.$$

В методе Бройдена один шаг дает информацию о кривизне вдоль одного направления, поэтому ранг матрицы поправок полагают малым и даже единичным. Бройден предложил выбирать длину шага s^k несколькими способами: либо с помощью линейного поиска, либо приравнивая $s^k = 1$ для всех шагов, либо задавая s^k таким образом, чтобы минимизировать или, по крайней мере, уменьшить значение $\left\|\nabla f(\mathbf{x}^k)\right\|$.

Независимо от типа линейного поиска, который используется для нахождения величины шага, метод Бройдена гарантирует для квадратичной функции сходимость за N шагов. Однако из-за отсутствия надежности при минимизации неквадратичной функции (например, возможна потеря положительной определенности матрицы B^{k+1}) предпочтительнее является использование матрицы поправок второго ранга.

Метод Дэвидона–Флетчера–Пауэлла (DFP метод)

$$B^{k+1} = B^k - \frac{B^k(\Delta\mathbf{g}^k)^T\Delta\mathbf{g}^k B^k}{\Delta\mathbf{g}^k B^k(\Delta\mathbf{g}^k)^T} + \frac{\Delta\mathbf{x}^k(\Delta\mathbf{x}^k)^T}{\Delta\mathbf{x}^k(\Delta\mathbf{g}^k)^T}$$

Эта формула была впервые предложена Дэвидоном, а затем доработана Флетчером и Пауэллом. Здесь не нужна операция обращения матрицы. DFP метод можно использовать без применения процедуры линейного поиска с целью определения s^k. Однако заданная длина шага должна гарантировать уменьшение значения функции и должна быть такой, чтобы $\Delta\mathbf{x}^k(\Delta\mathbf{g}^k)^T > 0$, что гарантирует сохранность положительной определенности матрицы B^k.

При квадратичной функции используемые направления поиска являются сопряженными. Именно это определяет эффективность метода. В большинстве случаев DFP метод работает вполне успешно.

Метод Бройдена–Флетчера–Гольдфарба–Шанно (BFGS метод)

$$B^{k+1} = \left(I - \frac{\Delta\mathbf{x}^k(\Delta\mathbf{g}^k)^T}{(\Delta\mathbf{x}^k)^T\Delta\mathbf{g}^k}\right) B^k \left(I - \frac{(\Delta\mathbf{x}^k)^T\Delta\mathbf{g}^k}{(\Delta\mathbf{x}^k)^T\Delta\mathbf{g}^k}\right) + \frac{\Delta\mathbf{x}^k(\Delta\mathbf{x}^k)^T}{\Delta\mathbf{x}^k(\Delta\mathbf{g}^k)^T}$$

BFGS метод находит минимум любой дважды непрерывно дифференцируемой выпуклой функции. При этом он хорошо работает и с невыпуклыми функциями. BFGS формулу часто называют дополнением формулы DFP. Считается, что метод BFGS менее чувствителен к различным погрешностям вычислительного процесса, чем DFP.

В квазиньютоновских методах в ходе оптимизации происходит постепенный переход от градиентного направления спуска к ньютоновскому. При этом используются преимущества каждого метода на соответствующем шаге. Первая итерация проводится в соответствии с методом наискорейшего спуска, который хорошо работает вдали от оптимума. Затем, ближе к оптимуму, поиск осуществляется ньютоновскими методами. Поиск выполняется без вычисления и обращения матрицы Гессе целевой функции. Рассмотренные способы обновления матрицы B сохраняют ее свойство положительной определенности на каждой итерации.

4.2.6. Дополнительная литература

В книгах [10],[22],[21],[25],[38] и [53] дается описание методов оптимизации, которые наиболее широко используются в настоящее время при решении различных практических задач в различных областях науки и техники. Предлагаются направления по повышению надежности и эффективности применения описываемых методов. Также приводятся необходиимые теоретические сведения для понимания выводов и функционирования основных методов.

Книги [15] и [29] могут мотивировать инженеров, экономистов, статистиков и других специалистов-практиков широко применять различные методы оптимизации для решения конкретных задач.

Книга [50] предоставляет теоретические результаты, лежащие в основе рассматриваемого класса задач безусловной оптимизации. Основное внимание уделяется тем теоретическим аспектам, которые имеют важное значение при практическом применении и программной реализации изучаемых методов. Приведена многочисленная коллекция m-файлов Matlab, находящихся в свободном доступе.

4.3. Методы условной оптимизации

4.3.1. Базовые принципы

Пусть $m \in \mathbb{N}$ и

$$\{1, \ldots, m\} = \mathcal{E} \cup \mathcal{I},$$

где $\mathcal{E}$ и $\mathcal{I}$ — это два непересекающихся множества. Пусть эти множества будут соответствовать множеству ограничений типа равенств и множеству ограничений типа неравенств.

Тогда стандартная форма постановки **задачи условной оптимизации** выглядит следующим образом:

$$\begin{cases} \text{minimize} & f(\mathbf{x}), & x \in D, \\ \\ \text{subject to} & g_j(\mathbf{x}) = 0, & j \in \mathcal{E}, \\ & g_j(\mathbf{x}) \geqslant 0, & j \in \mathcal{I}. \end{cases} \tag{4.4}$$

где g_j, $j \in \mathcal{E}$ — это множество ограничений типа равенств, а g_j, $j \in \mathcal{I}$ — это множество ограничений типа неравенств.

Ограничения делят область поиска на две части: допустимую область, в которой все ограничения выполняются, и недопустимую область, где, по крайней мере, одно из ограничений нарушается. В большинстве практических задач искомый оптимум находится на границе между допустимой и недопустимой областями.

Все ограничения можно разделить на ограничения в виде равенств и ограничения в виде неравенств, линейные и нелинейные ограничения. При этом задачи линейного программирования можно рассматривать как частный случай задач нелинейного программирования. Кроме того, среди задач условной оптмизации можно выделить задачи с ограничениями в виде равенств, задачи с ограничениями в виде неравенств и задачи с ограничениями в виде равенств и неравенств.

Решение задачи условной оптимизации часто может быть найдено с использованием так называемой **функции Лагранжа**. Функция Лагранжа определяется следующим образом:

$$L(\mathbf{x}, \lambda) = f(\mathbf{x}) + \sum_{j=1}^{m} \lambda_j g_j(\mathbf{x}), \tag{4.5}$$

гле λ_j — параметры, называемые **множителями Лагранжа**.

Условия существования минимума в задачах с ограничениями в виде равенств

В общем виде задача условной оптимизации с m ограничениями в виде равенств и n переменными может быть записана следующим образом:

$$\begin{cases} \text{minimize} & f(\mathbf{x}), \quad x \in D, \\ \text{subject to} & g_j(\mathbf{x}) = 0, \quad j = 1, \ldots, m \leqslant n, \end{cases}$$

(в этом случае множество $\mathcal{I}$ пусто).

Предположим, что $\mathbf{x}^* \in D$ и

— f, g_j, $j = 1, \ldots, m$ являются дважды непрерывно дифференцируемыми в окрестности точки $\mathbf{x}^*$,

— градиенты $\nabla g_j(\mathbf{x}^*)$ являются линейно независимыми векторами. Это означает, что

$$\sum_{j=1}^{m} \lambda_j g_j(\mathbf{x}^*) = 0 \quad \text{только в том случае, если} \quad \lambda_1 = \lambda_2 = \ldots = \lambda_m = 0,$$

где $\mathbf{x}^*$ является точкой локального минимума.

Пусть $A(\mathbf{x}^*) \in \mathbb{R}^{m \mathbf{x} n}$ — это Якобиан всех ограничений для $\mathbf{x}^*$ (полный строчный ранг), и $Z(\mathbf{x}^*) \in \mathbb{R}^{n \mathbf{x}(m-n)}$-матриц базиса для ядра отображения $A(\mathbf{x}^*)$.

Необходимые условия существования локального минимума.

Если вышеописанные предположения имеют место, то существуют такие множители Лагранжа $\lambda^* \in \mathbb{R}^m$, для которых:

— $g_j(\mathbf{x}^*) = 0$, $j = 1, \ldots, m$,

— $\nabla_{\mathbf{x}} L(\mathbf{x}^*, \lambda^*) = (Z(\mathbf{x}^*))^T (\nabla f(\mathbf{x}^*))^T = \nabla f(\mathbf{x}^*) - (A(\mathbf{x}^*))^T (\lambda^*)^T = 0$

— $(Z(\mathbf{x}^*))^T \nabla^2_{\mathbf{xx}} L(\mathbf{x}^*, \lambda^*) Z(\mathbf{x}^*)$ положительно определена.

Здесь $L(\mathbf{x}, \lambda)$ — это функция Лагранжа.

$\lambda^* \in \mathbb{R}^m$ — называют множителями Лагранжа.

Достаточные условия.

Если существуют такие множители Лагранжа $\lambda^* \in \mathbb{R}^m$, для которых выполняются следующие условия:

— $g_j(\mathbf{x}^*) = 0$, $j = 1, \ldots, m$,

— $\nabla_{\mathbf{x}} L(\mathbf{x}^*, \lambda^*) = (Z(\mathbf{x}^*))^T (\nabla f(\mathbf{x}^*))^T = \nabla f(\mathbf{x}^*) - (A(\mathbf{x}^*))^T (\lambda^*)^T = 0$

— $(Z(\mathbf{x}^*))^T \nabla^2_{\mathbf{xx}} L(\mathbf{x}^*, \lambda^*) Z(\mathbf{x}^*)$ положительно определена,

тогда $\mathbf{x}^*$ является локальным минимумом.

Условия существования минимума в задачах с ограничениями в виде неравенств

В общем виде задача условной оптимизации с m ограничениями в виде равенств и n переменными может быть записана следующим образом:

$$\begin{cases} \text{minimize} & f(\mathbf{x}), \quad x \in D, \\ \text{subject to} & g_j(\mathbf{x}) \geqslant 0, \quad j = 1, \ldots, m \leqslant n, \end{cases}$$

(в этом случае множество $\mathcal{E}$ пусто).

Предположим, что $\mathbf{x}^* \in D$ и

— f, g_j, $j = 1, \ldots, m$ являются дважды непрерывно дифференцируемыми в окрестности точки $\mathbf{x}^*$,

— градиенты $\nabla g_j(\mathbf{x}^*)$ являются линейно независимыми векторами. Это означает, что

$$\sum_{j=1}^{m} \lambda_j g_j(\mathbf{x}^*) = 0 \quad \text{только в том случае, если} \quad \lambda_1 = \lambda_2 = \ldots = \lambda_m = 0,$$

где $\mathbf{x}^*$ является точкой локального минимума.

Пусть $Z(\mathbf{x}^*)$ — это матрица базиса для ядра отображения Якобиана ограничений точки $\mathbf{x}^*$.

Необходимые условия для локального минимума.

Если вышеописанные предположения имеют место, то существуют такие множители Лагранжа $\lambda^* \in \mathbb{R}^m$, для которых:

— $g_j(\mathbf{x}^*) \geqslant 0$, $j = 1, \ldots, m$,

— $\nabla_{\mathbf{x}} L(\mathbf{x}^*, \lambda^*) = (Z(\mathbf{x}^*))^T (\nabla f(\mathbf{x}^*))^T = 0$,

— $\lambda_j^* \geqslant 0$, $j = 1, \ldots, m$,

— $(\lambda^*)^T g(\mathbf{x}^*) \geqslant 0$,

— $(Z(\mathbf{x}^*))^T \nabla^2_{\mathbf{xx}} L(\mathbf{x}^*, \lambda^*) Z(\mathbf{x}^*)$ положительно определена.

Достаточные условия.

Если существуют такие множители Лагранжа $\lambda^* \in \mathbb{R}^m$, для которых выполняются следующие условия:

— $g_j(\mathbf{x}^*) \geqslant 0$, $j = 1, \ldots, m$,

— $\nabla_{\mathbf{x}} L(\mathbf{x}^*, \lambda^*) = (Z(\mathbf{x}^*))^T (\nabla f(\mathbf{x}^*))^T = 0$,

— $\lambda_j^* \geqslant 0$, $j = 1, \ldots, m$,

— $(\lambda^*)^T g(\mathbf{x}^*) = 0$,

— $(Z(\mathbf{x}^*))^T \nabla^2_{\mathbf{xx}} L(\mathbf{x}^*, \lambda^*) Z(\mathbf{x}^*)$ положительно определена,

тогда $\mathbf{x}^*$ является локальным минимумом.

Условия существования минимума в задачах с ограничениями в виде равенств и неравенств

Вернемся к общему виду (4.4) описания задачи условной минимизации.

$$\begin{cases} \text{minimize} & f(\mathbf{x}), & x \in D, \\ \\ \text{subject to} & g_j(\mathbf{x}) = 0, & j = 1, \ldots, m, \\ & g_j(\mathbf{x}) \geqslant 0, & j = m+1, \ldots, p. \end{cases}$$

где g_j, $j \in \mathcal{E}$ — это множество ограничений в виде равенств, а g_j, $j \in \mathcal{I}$ — это множество ограничений в виде неравенств.

Необходимые условия первого порядка.

Пусть

$$A(\mathbf{x}) = \mathcal{E} \cup \{j \in \mathcal{I} : g_j(\mathbf{x}) = 0\}$$

- это множество всех действующих ограничений в точке $\mathbf{x}$.

Предположим, что в точке $\mathbf{x}^*$

— градиенты всех действующих ограничений $\nabla g_j(\mathbf{x}^*)$, $j \in A(\mathbf{x}^*)$ являются линейно независимыми. Это означает, что

$$\sum_{j=1}^{m} \lambda_j g_j(\mathbf{x}^*) = 0 \quad \text{только в том случае, если} \quad \lambda_1 = \lambda_2 = \ldots = \lambda_m = 0,$$

где $\mathbf{x}^*$ является точкой локального минимума.

Тогда существуют такие множители Лагранжа $\lambda^* = (\lambda_1^*, \ldots, \lambda_m^*) \in \mathbb{R}^m$, для которых выполняются следующие условия:

— $\nabla_{\mathbf{x}} L(\mathbf{x}^*, \lambda^*) = 0$,
— $g_j(\mathbf{x}^*) = 0$ for all $j \in \mathcal{E}$,
— $g_j(\mathbf{x}^*) \geqslant 0$ for all $j \in \mathcal{I}$,
— $\lambda_j^* \geqslant 0$ for all $j \in \mathcal{I}$,
— $\lambda_j^* g_j(\mathbf{x}^*) = 0$ for all $j \in \mathcal{I} \cup \mathcal{E}$

(эти условия в теории оптимизации называются условиями Каруша-Куна-Таккера (ККТ)).

Для неактивных ограничений $g_j(\mathbf{x}^*) > 0$ получаем $\lambda_j^* = 0$, таким образом функция Лагранжа становится

$$0 = \nabla_{\mathbf{x}} L(\mathbf{x}^*, \lambda^*) = \nabla f(\mathbf{x}^*) - \sum_{j \in A(\mathbf{x}^*)} \lambda_j^* \nabla g_j(\mathbf{x}^*).$$

Необходимые условия второго порядка.

Пусть точка $\mathbf{x}^*$ является точкой локального минимума и в ней выполняется условие линейной независимости. (Т.е. $\sum_{j=1}^{m} \lambda_j g_j(\mathbf{x}^*) = 0$ только, если $\lambda_1 = \lambda_2 = \ldots = \lambda_m = 0$,). Пусть λ^* будет таким, что условия ККТ выполняются для $\mathbf{x}^*$ и λ^*, и определим $F(\lambda^*)$ следующим образом:

$$w \in F(\lambda^*) \iff \begin{cases} \nabla g_j(\mathbf{x}^*)^T w = 0 & \text{if} \quad j \in \mathcal{E}, \\ \nabla g_j(\mathbf{x}^*)^T w = 0 & \text{if} \quad j \in A(\mathbf{x}^*) \cap \mathcal{I},\ \lambda_j^* > 0, \\ \nabla g_j(\mathbf{x}^*)^T w \geqslant 0 & \text{if} \quad j \in A(\mathbf{x}^*) \cap \mathcal{I},\ \lambda_j^* = 0. \end{cases}$$

Тогда

$$w \nabla_{\mathbf{xx}}^2 L(\mathbf{x}^*, \lambda^*) w^T \geqslant 0 \quad \text{для всех} \quad w \in F(\lambda^*).$$

Достаточные условия второго порядка.

Допустим, что для какой-либо допустимой точки $\mathbf{x}^* \in D$ существует такой множитель Лагранжа λ^*, что ККТ условия выполняются. В этом случае, если следующие условия выполняются

$$w\nabla^2_{\mathbf{xx}}L(\mathbf{x}^*,\lambda^*)w^T > 0 \quad \text{for all} \quad 0 \neq w \in F(\lambda^*),$$

тогда $\mathbf{x}^*$ является точкой локального минимума.

Необходимо отметить, что

— если $\nabla^2_{\mathbf{xx}}L(\mathbf{x}^*,\lambda^*)$ положительно определена, тогда $Z^T\nabla^2 f(\mathbf{x}^*)Z$ также положительно определена,

— требуется "отрицательная определенность" вместо "положительной определенности" в случае локального максимизации,

— множители Лагранжа λ_j^* всегда равны 0 для неактивных ограничений.

4.3.2. Методы для решения задач условной оптимизации

В настоящее время существуют различные подходы к классификации методов условной оптимизации. По одному из них все многочисленные методы решения задачи условной оптимизации функции n переменных при наличии m ограничений можно разделить на: методы штрафов и барьеров (работают в n-пространстве), методы отсечений (работают в m-пространстве), методы множителей Лагранжа (работают в $(n+m)$-пространстве) и методы непосредственного решения задач условной оптимизации (работают в $(n-m)$-, n,m или $(n+m)$-пространстве).

Процедура отсечения, которая используется для решения полностью целочисленных задач линейного программирования, впервые была предложена Гомори в 1958 году. Постановка такой задачи получается из задачи линейного программирования добавлением условий целочисленности переменных. Существенный недостаток такого способа решений целочисленных задач связан с трудностями, возникающими при построении конечной системы линейных ограничений, определяющих выпуклую оболочку. Данциг предложил более простой подход, основанный на использовании линейной релаксации задачи целочисленного линейного программирования, получаемой удалением условий целочисленности. В настоящее время классические методы отсечений мало используются для решения практических задач.

Методы штрафных и барьерных функций

Метод штрафных функций может применяться для решения задач условной оптимизации с ограничениями как в виде равенств, так и в виде неравенств. Основная идея метода заключается в том, чтобы аппроксимировать исходную задачу условной оптимизации некоторой последовательностью вспомогательных задач безусловной оптимизации, и затем решить задачи этой последовательности известными методами безусловной оптимизации. В методах штрафных функций ограничения задачи (все или некоторая их часть) отбрасываются, а к целевой функции добавляется штраф за их нарушение, т.е. слагаемое, которое принимает большие положительные значения в точках, не удовлетворяющих отброшенным ограничениям (для решения задач минимизации). Штрафная функция равна нулю в допустимой области.

Таким образом, задача условной оптимизации записывается в следующем виде

$$\left.\begin{array}{lll} \text{minimize} & f(\mathbf{x}), & x \in D, \\ \\ \text{subject to} & g_j(\mathbf{x}) = 0, & j = 1, \ldots, m, \\ & g_j(\mathbf{x}) \geqslant 0, & j = m+1, \ldots, p \end{array}\right\} \Longrightarrow \text{minimize } F(\mathbf{x}, r) = f(\mathbf{x}) + P(\mathbf{x}, r)$$

где $P(\mathbf{x}, r)$ — это штрафная функция, а r — положительный параметр штрафа.

Параметр штрафа r определяет величину штрафа в недопустимых точках и тем самым регулирует степень приближения вспомогательной задачи оптимизации без ограничений к исходной задаче с ограничениями.

Наример, в случае ограничений в виде равенств, **штрафная функция** наиболее часто строится на основе квадратичного штрафа:

$$P(\mathbf{x}, r) = \frac{r}{2} \sum_{j \in \mathcal{E}} g_j^2(\mathbf{x}) + \frac{r}{2} \sum_{j \in \mathcal{I}} [-g(\mathbf{x}_j)]^2,$$

где

$$[a] = \max\{a, 0\}$$

Сходимость метода зависит от величины штрафа r. Чем больше r, тем выше скорость сходимости. Однако при очень больших значениях r штрафная функция приобретает ярко выраженную овражную структуру, поэтому скорость сходимости резко падает.

Метод барьерных функций (также известный как метод барьеров, или метод внутренних штрафов) добавляет слагаемое к исходной целевой функции, которое не позволяет генерируемым точкам выходить за пределы допустимой области. Тем самым в методах барьерных функций исключен основной недостаток штрафных функций, связанный с тем, что при использовании штрафов аппроксимируемые точки не принадлежат множеству допустимых решений. Метод барьерных функций обычно используется в случае ограничений в виде неравенств. В общем виде преобразования в барьерном методе выглядят следующим образом:

$$\left.\begin{array}{lll} \text{minimize} & f(\mathbf{x}), & x \in D, \\ \\ \text{subject to} & g_j(\mathbf{x}) \geqslant 0, & j = 1, \ldots, m \end{array}\right\} \Longrightarrow \text{minimize} \quad F(\mathbf{x}, r) = f(\mathbf{x}) + B(\mathbf{x}, r)$$

где $B(x, r)$ — это барьерная функция, and r — это положительный барьерный параметр.

В качестве барьерных функций можно использовать обратную и логарифмическую функции

$$B(\mathbf{x}, r) = r \sum_{j=1}^{m} \frac{1}{g_j(\mathbf{x})}, \quad B(\mathbf{x}, r) = r \sum_{j=1}^{m} \ln[-g_j(\mathbf{x})].$$

Обе эти функции определены и непрерывны внутри допустимого множества и стремяться к бесконечности при приближении к границе множества изнутри. Чем ближе значение параметра r к нулю, тем больше скорость сходимости метода. Однако с уменьшением r барьерная функция становится все более овражной. Поэтому принимать сразу r малым числом нецелесообразно.

Методы множителей Лагранжа

Решение задачи условной оптимизации можно найти и с помощью так называемых **методов множителей Лагранжа**. В этих методах также используется штрафная функция, однако она добавляется не к целевой функции, а к классической функции Лагранжа. В результате задача условной оптимизации сводится к решению последовательности задач поиска безусловного минимума модифицированной функции Лагранжа.

Множители Лагранжа можно трактовать как коэффициенты чувствительности точки оптимального решения $\mathbf{x}^*$ относительно возмущения ограничений. Кроме того, множители Лагранжа являются переменными задачи, двойственной к исходной.

Метод множителей Лагранжа сожно использовать для решения задачи оптимизации с условиями в виде равенст:

$$\begin{cases} \text{minimize} & f(\mathbf{x}), & x \in D, \\ \text{subject to} & g_j(\mathbf{x}) = 0, & j = 1, \ldots, m < n. \end{cases}$$

Модификация целевой функции выполняется согласно (4.5).

В случае решения задачи оптимизации с условиями в виде неравенств

$$\begin{cases} \text{minimize} & f(\mathbf{x}), & x \in D, \\ \text{subject to} & g_j(\mathbf{x}) \geqslant 0, & j = 1, \ldots, m < n. \end{cases}$$

необходимо минимизировать целевую функцию (4.5) относительно $\mathbf{x}$ и максимизировать относительно λ.

4.3.3. Дополнительная литература

Источники [14],[22],[25],[38],[40],[53] представляют собой достаточно полный обзор методов решения задач оптимизации с ограничениями и демонстрируют работу различных подходов на конкретных примерах. Помимо теоретических исследований сходимости, значительное внимание уделено обсуждению вычислительной эффективности рассматриваемых подходов.

В книгах [15] и [17] обсуждаются базовые основы методов преобразования задач условной оптимизации к последовательности задач безусловной оптимизации. Представляются как традиционные подходы, так и их модификаций. Изложение сопровождается рассмотрением примеров.

Библиографические ссылки

1. Бахвалов Н. С., Жидков Н. П., Кобельков Г. М. Численные методы. М. : БИНОМ. Лаб. знаний, 2003. 636 с.
2. Зенкевич О., Морган К. Конечные элементы и аппроксимация. М. Мир, 1996. – 318 с.
3. Митчелл Э., Уэйт Р. Метод конечных элементов для уравнений с частными производными. М.: Мир, 1981. - 216 с.
4. Норри Д., де Фриз Ж. Введение в метод конечных элементов. - М.: Мир, 1981. - 155 с.
5. Ольшанский М.А.: Лекции и упражнения по многосеточным методам – М.: Издательство ЦПИ при механико-математическом факультете МГУ им. М.В. Ломоносова, 2003. 176 с.
6. Ортега Дж., Пул У. Введение в численные методы решения дифференциальных уравнений. М.: Наука, 1986. - 288 с.
7. Самарский А.А., Гулин А.В. Численные методы. М.: Наука, 1989. 432 с.
8. Калиткин Н. Н. Численные методы. 2 изд. – БХВ-Петербург, 2011, 592 с.
9. Allen M.B. and Isaacson E.L.: *Numerical analysis for applied science*, Vol. 35. John Wiley and sons, New Jersey (2011).
10. Andrei N.: *A new descent gradient method for unconstrained optimization*, Research Institute for Informatics, Romania (2004).
11. Antia H.M.: *Numerical methods for scientists and engineers*. Vol. 1. Springer Science and Business Media, New York (2002).
12. Ascher U.M., and Petzold L.R.: *Computer methods for ordinary differential equations and differential-algebraic equations*, Vol. 61. SIAM, Philadelphia (1998).
13. Avriel M.: *Nonlinear Programming*, Prentice-Hall, Englewood Cliffs, New Jersey (1976).
14. Bazaraa M.S., Sherali H.D. and Shetty C. M.: *Nonlinear Programming: Theory and Algorithms. Second Edition*, John Wiley & Sons, New York (1993).
15. Belegundu A.D., Chandrupatla T.R.: *Optimization Concepts and Applications in Engineering. 2nd Edition*, Cambridge University press, Cambridge (2014).
16. Borse G.J.: *Numerical methods with MATLAB: A resource for scientists and engineers*, International Thomson Publishing, Boston (1996).
17. Boyd J.P.: *Chebyshev and Fourier spectral methods*, Courier Corporation , New York (2001).
18. Boyd S., Vandenberghe L.: *Convex optimization*,Cambridge University press, Cambridge (2009).
19. Bramble J.H.: *Multigrid methods*. Vol. 294. CRC Press, New York (1993).

20. Butcher J.C.: *The numerical analysis of ordinary differential equations: Runge-Kutta and general linear methods*, Wiley-Interscience, New York, (1987).
21. Dennis J.E.,Schnabel R.B.: *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*, Prentice-Hall Inc.,New Jersey (1983).
22. Cheneyand E. Kincaid D.: *Numerical mathematics and computing*, Cengage Learning, Boston (2012).
23. J.R.Dormand: Numerical Methods for Differential Equations, CRC Press, Boca Raton (1996).
24. Elden L., Wittmeyer-Koch L., and Nielsen H.B.: *Introduction to Numerical Computation-analysis and MATLAB illustrations.* Studentlitteratur (2004).
25. Fletcher R.: *Practical Methods of Optimization. Second Edition*, John Wiley & Sons, New York (2000).
26. Fornberg B.: *A Practical Guide to Pseudospectral Methods.* Cambridge University Press, Cambridge (1998).
27. Gautschi W.: *Numerical analysis*, Springer Science & Business Media, New York (2011).
28. Gerald C.F., Wheatley P.O. and Bai F.: *Applied numerical analysis*, Addison-Wesley, New York (1989).
29. Gill P. E., Murray W., Wright M. H.: *Practical Optimization*, Academic Press, London (1981).
30. Gockenbac M.S.h: *Understanding and implementing the finite element method*, SIAM, Philadelphia (2006).
31. Golub G.H. and Ortega J.M., *Scientific Computing and DifferentialEquations: an Introduction to Numerical Methods*, Academic Press inc., London (1991).
32. Gottlieb D. and Orszag S.: The Numerical Analysis of Spectral Methods, SIAM, Philadelphia (1987).
33. Hairer E., Norsett S.P., Wanner G.: *Solving ordinary differential equations. I. Nonstiff problems. Second edition*, Springer Series in Computational Mathematics, 8. Springer-Verlag, Berlin (1993).
34. Hairer E., Wanner G.: *Solving ordinary differential equations. II. Stiff and differential-algebraic problems. Second revised edition, paperback.* Springer Series in Computational Mathematics, 14. Springer-Verlag, Berlin, (2010).
35. Hackbusch W.: *Multi-grid methods and applications.* Vol. 4. Springer-Verlag, Berlin (1985).
36. Hackbusch W.: *Multi-grid methods and applications.* Vol. 4. Springer Science & Business Media, Berlin, New York (2013).
37. Hamming R.: *Numerical methods for scientists and engineers*, Courier Corporation, New York (2012).
38. Himmelblau D. M.: *Applied nonlinear programming*, McGraw-Hill Book Company, United States (1972).
39. Hoffman J.D. and Frankel S.: *Numerical methods for engineers and scientists*, CRC press, Boca Raton (2001).
40. Horst R., Tuy H.: *Global Optimization: Deterministic Approaches. Third Edition*, Springer-Verlag, Berlin (1995).
41. Huebner K.H., Dewhirst D.L., Smith D.E., and Byrom T.G.: *The finite element method for engineers*, John Wiley and Sons, New York (2008).

42. Iserles A.: A first course in the numerical analysis of differential equations. Vol. 44. Cambridge University Press, Cambridge (2009).
43. Johnson C.: *Numerical solution of partial differential equations by the finite element method. Courier Corporation*, New York (2012).
44. Kincaid D.R. and Cheney E.W.: *Numerical analysis: mathematics of scientific computing.* Vol. 2. American Mathematical Soc., New York (2002).
45. Lapidus L. and Pinder G.F.: *Numerical solution of partial differential equations in science and engineering*, John Wiley and Sons, New York (2011).
46. Leader J.J.: *Numerical Analysis and Scientific Computation*, Addison-Wesley, New York (2004).
47. LeVeque R.J.: *Finite difference methods for ordinary and partial differential equations: steady-state and time-dependent problems.* Vol. 98. SIAM, Philadelphia (2007).
48. Lindfield G.R., Penny J.: *Numerical methods: using MATLAB*, Academic Press, London (2012).
49. Liu G.R. and Quek S.S.: *The finite element method: A practical course.* Butterworth-Heinemann (2013).
50. Lu A., Wu-Sheng: *Practical Optimization. Algorithms and Engineering Applications Antoniou*, Springer, Berlin (2007).
51. Suli E. and Mayers D.F.: *An introduction to numerical analysis*, Cambridge University Press, Cambridge (2003).
52. Ortega J.M.: *Numerical analysis: a second course* Vol. 3. SIAM, Philadelphia (1990).
53. Press W.H.: *Numerical recipes. 3rd edition: The art of scientific computing*, Cambridge university press, Cambridge (2007).
54. Quarteroni A.: *Numerical models for differential problems*, Springer Science and Business Media, T. 2. New York (2010).
55. Ralston A. and Rabinowitz P.: *A first course in numerical analysis*, Courier Corporation, New York (2012).
56. Rao K.S.: Numerical methods for scientists and engineers. PHI Learning Pvt. Ltd. (2007).
57. Saad Y.: *Iterative methods for sparse linear systems*, SIAM, Philadelphia (2003).
58. Quarteroni A., Sacco R., and Saleri F.: *Numerical mathematics.* Vol. 37. Springer Science & Business Media, New York (2010).
59. Schilling H.A., Harris S.L.: *Applied numerical methods for engineers using MATLAB.* Brooks/Cole Publishing Co. (1999).
60. Smith G.D.: *Numerical solution of partial differential: Finite Difference Methods*, Clarendon Press, Oxford, (1985).
61. Stoer J. and Bulirsch R.: *Introduction to numerical analysis*, Springer Science & Business Media, New York (1993).
62. Thomas J.W.: *Numerical partial differential equations: finite difference methods*, Vol. 22. Springer Science and Business Media, New York (1995).
63. Trefethen L.N.: *Spectral Methods in MATLAB*, Vol. 10. SIAM, Philadelphia (2000).
64. Versteeg H.K. and Malalasekera W. *An introduction to computational fluid dynamics: the finite volume method,* Pearson Education, Singapore (2007).
65. Watkins D.: *Fundamentals of Matrix Computations*, 2nd Ed., Wiley-Interscience, New York (2002).

66. Wilde D. J.: *Optimum Seeking Methods*, Prentice-Hall Inc., Englewood Cliffs. N.J. (1964).
67. Yang W.Y., Cao W., Chung T.S. and Morris J.: A*pplied numerical methods using MATLAB*, John Wiley and Sons, New York (2005).

Предметный указатель