НИЯУ МИФИ

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ЯДЕРНЫЙ УНИВЕРСИТЕТ «МИФИ»

ПРИКЛАДНАЯ
МАТЕМАТИКА
кафедра 31

Институт лазерных и плазменных технологий

Р. В. Муратов

Итерационные методы решения систем линейных уравнений

Курс лекций

Москва 2025

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ЯДЕРНЫЙ УНИВЕРСИТЕТ «МИФИ»

Институт лазерных и плазменных технологий

Р. В. Муратов

Итерационные методы решения систем линейных уравнений

Курс лекций

Москва 2025

УДК 519.612(075)
ББК 22.143я7
М 91

Муратов Р. В. Итерационные методы решения систем линейных уравнений: Курс лекций [Электронный ресурс]. — М.: НИЯУ МИФИ, 2025. — 92 с.

Настоящий курс посвящен численным методам решения задач математической физики, возникающих при аппроксимации эллиптических и параболических уравнений (уравнения теплопроводности, Шрёдингера, уравнение Пуассона). Основное внимание уделено итерационным алгоритмам решения больших разреженных систем линейных уравнений.

Предназначен для студентов, изучающих вычислительную математику и численные методы, и содержит как теоретические основы, так и практические аспекты реализации рассмотренных алгоритмов. Особое внимание уделено вопросам оптимизации вычислений при работе с разреженными матрицами, что делает материал полезным для решения прикладных задач в физике и инженерии.

Рецензенты:
доц. каф. прикладной математики Е. В. Степин (НИЯУ МИФИ);
зам. дир-ра Института ЛаПлаз П. Н. Рябов (НИЯУ МИФИ).

ISBN 978-5-7262-3145-7

Содержание

Предисловие

Данный курс лекций посвящен методам итерационного решения систем линейных уравнений. Необходимость решать системы линейных уравнений возникает в бесчисленном количестве приложений. Однако данный курс целиком сосредоточен на решении систем уравнений, которые возникают при численном решении задач математической физики. Речь идет о краевых задачах для эллиптических уравнений (двумерные уравнения Лапласа и Пуассона), смешанных краевых задачах параболического типа (уравнения теплопроводности, диффузии, уравнение Шрёдингера).

В курсе предложено шесть лабораторных работ. Среди них нет абстрактных задач, вроде поиска решения некоторой системы линейных уравнений со случайно заданной матрицей. Все системы линейных уравнений, которые рассматриваются в курсе, получаются из численной аппроксимации уравнений математической физики.

1. Лабораторная работа в разделе 1.5. Задачи на поиск приближенных собственных функций: моды колебания струн, мембран, энергии и волновые функции стационарных состояний квантовых частиц.
2. Лабораторная работа в разделе 4.3. Методы простой итерации и минимальной невязки для решения уравнения Пуассона в двумерной области.
3. Лабораторная работа в разделе 5.3. Методы Якоби и Зейделя для решения уравнения Пуассона в двумерной области.
4. Лабораторная работа в разделе 6.3. Разреженные матрицы, использование библиотек линейной алгебры для решения систем уравнений.
5. Лабораторная работа в разделе 7.3. Численное моделирование эволюционных задач по неявным схемам: двумерные уравнения теплопроводности, диффузии, уравнение Шрёдингера.
6. Лабораторная работа в разделе 9.3. Численное моделирование задач на неструктурированных сетках в произвольных областях, методы конечных объемов и конечных элементов.

Для выполнения лабораторной работы подходят любые языки программирования, в которых существуют библиотеки линейной алгебры и разреженных матриц, а также есть простая возможность для визуализации одномерных и двумерных графиков. Примеры в методичке приводятся на языке программирования Python как на одном из самых популярных и распространенных во всех областях.

Лекция 1. Задача на собственные векторы и собственные значения

Вспомним определения. Пусть в векторном (линейном) пространстве V над полем K задан линейный оператор $A : V \to V$. Требуется найти множество скаляров $\lambda \in K$ и множество соответствующих векторов $v \in V$, которые удовлетворяют соотношению:

$$Av = \lambda v.$$

Подходящие значения λ называют собственными значениями (СЗ) оператора. Множество собственных значений образует *спектр оператора* A. Соответствующие векторы v называются собственными векторами (СВ) или, если V — это линейное пространство функций, собственными функциями (СФ).

Задача поиска собственных векторов и собственных значений встречается во многих приложениях. Некоторые из них:

- *метод разделения переменных (Фурье)* для уравнений в частных производных требует поиска СФ и СЗ задачи Штурма-Лиувилля, а решение представляется в виде ряда по собственным функциям;
- *стационарное уравнение Шрёдингера* является задачей на собственные значения для гамильтониана. Собственные значения определяют возможные энергии частиц;
- в рамках *спектральной теории графов* исследуется спектр матрицы смежности графа, собственные значения которой являются важными инвариантами графа;
- *цепи Маркова* строятся на основе матриц переходных вероятностей. Спектры матриц переходов позволяют классифицировать марковские процессы;
- *анализ вибраций/резонансных частот* конструкций необходим при оценке динамической прочности. Резонансные частоты напрямую связаны с собственными значениями;
- *метод главных компонент* требует поиска СВ и СЗ ковариационной матрицы. Спектральная декомпозиция матриц в целом имеет большое значение во многих задачах;
- при решении *систем гиперболических уравнений* собственные векторы показывают характеристические направления.

Во многих прикладных задачах не требуется нахождение полного спектра оператора. Часто достаточно найти только несколько максимальных

(или минимальных) по модулю собственных значений, в некоторых задачах необходимо знать только нижнюю и верхнюю оценки спектра оператора. В рамках курса, посвященного итерационным методам решения систем линейных уравнений, самым важным фактом для нас является то, что спектр матрицы СЛАУ напрямую влияет на скорость сходимости различных итерационных методов. Также спектр матрицы позволяет определить оптимальные итерационные параметры для обеспечения самой быстрой сходимости.

1.1. Итерационные методы поиска СВ и СЗ

Степенной метод позволяет найти максимальное по модулю собственное значение (спектральный радиус матрицы) и соответствующий ему собственный вектор. Зададим случайный вектор $v^{(0)}$ и рассмотрим следующую итерационную процедуру:

$$\tilde{v}^{(k+1)} = Av^{(k)}, \qquad v^{(k+1)} = \frac{\tilde{v}^{(k+1)}}{\left\|\tilde{v}^{(k+1)}\right\|}, \tag{1.1}$$

вторым действием выполняется нормировка вектора, $\|v\| = \sqrt{(v,\, v)}$. Если начальное приближение $v^{(0)}$ не оказалось ортогонально собственному вектору матрицы A, соответствующему максимальному по модулю собственному значению, тогда итерации сходятся к этому собственному вектору. Почему это работает?

Пусть собственные значения матрицы A равны $\lambda_1, \lambda_2, \ldots, \lambda_m$, а соответствующие собственные векторы – $v_1, v_2, \ldots, v_m$. Система собственных векторов может быть неполной, а число m может быть меньше размерности линейного пространства V. Для любого начального приближения $v^{(0)}$ существует разложение по собственным векторам оператора A:

$$v^{(0)} = \sum_{i=1}^{m} c_i v_i + w, \qquad w \in \mathrm{Ker} A.$$

Предполагаем, что в данном разложении присутствует слагаемое с искомым собственным вектором (по условию, начальное приближение $v^{(0)}$ имеет ненулевую компоненту в направлении искомого собственного вектора). Тогда, если многократно подействовать оператором A на начальное приближение $v^{(0)}$, получим

$$A^k v^{(0)} = \sum_{i=1}^{m} \lambda^k c_i^k v_i.$$

При стремлении k к бесконечности в данной сумме быстрее всего растет слагаемое, соответствующее максимальному собственному значению, к нему и сходятся итерации. Текущее приближение для собственных значений можно

получить по формуле

$$\lambda^{(k)} = \frac{\left(Av^{(k)},\, v^{(k)}\right)}{\left(v^{(k)},\, v^{(k)}\right)}. \tag{1.2}$$

Итерации можно завершить после достижения необходимой точности по $\lambda^{(k)}$, когда отличия $\lambda^{(k)}$ между итерациями становятся меньше некоторого порогового значения $\varepsilon \ll 1$.

Обратный степенной метод позволяет найти собственное значение, наиболее близкое к наперед заданному значению μ. Пусть собственные значения матрицы A равны $\lambda_1, \lambda_2, \ldots, \lambda_m$, а соответствующие собственные векторы — $v_1, v_2, \ldots, v_m$. Рассмотрим матрицу $A_1 = A - \mu E$, где E — единичная матрица. Легко убедиться, что матрица A_1 имеет тот же набор собственных векторов v_i, но им соответствуют собственные значения $\lambda_i - \mu$. Возьмем теперь обратную матрицу $A_2 = A_1^{-1} = (A - \mu E)^{-1}$, для матрицы A_2 собственные векторы остались прежними, а собственные значения стали обратными $\frac{1}{\lambda_i - \mu}$. Применим теперь обычный степенной метод к матрице A_2:

$$\tilde{v}^{(k+1)} = (A - \mu E)^{-1} v^{(k)}, \qquad v^{(k+1)} = \frac{\tilde{v}^{(k+1)}}{\left\|\tilde{v}^{(k+1)}\right\|}. \tag{1.3}$$

Метод сходится к собственному вектору, соответствующему максимальному по модулю собственному значению из набора $\frac{1}{|\lambda_i - \mu|}$, или, что эквивалентно, минимальному из набора $|\lambda_i - \mu|$, т.е. самому близкому к значению μ. Оценка собственных значений на каждой итерации получается по формуле (1.2). Критерий остановки аналогичен предыдущему пункту.

Метод итераций Рэлея улучшает обратный степенной метод путем использования нового приближения к собственному значению на каждой итерации. Пусть, как и ранее, задан случайный вектор $v^{(0)}$, и известно начальное приближение для собственного значения $\lambda^{(0)}$. Итерационная процедура выглядит следующим образом:

$$\begin{aligned} \tilde{v}^{(k+1)} &= \left(A - \lambda^{(k)} E\right)^{-1} v^{(k)}, \qquad v^{(k+1)} = \frac{\tilde{v}^{(k+1)}}{\left\|\tilde{v}^{(k+1)}\right\|}, \\ \lambda^{(k+1)} &= \frac{\left(Av^{(k+1)},\, v^{(k+1)}\right)}{\left(v^{(k+1)},\, v^{(k+1)}\right)}. \end{aligned} \tag{1.4}$$

За счет обновления приближения $\lambda^{(k)}$ в ходе итераций метод итераций Рэлея имеет более быструю сходимость по сравнению с обратным степенным методом. Однако метод Рэлея не всегда сходится к собственному значению, ближайшему к начальному приближению.

Важное замечание! Когда в формулах встречается запись $A^{-1}v$, в программе не следует вычислять обратную матрицу. Вычисление обратной

матрицы — крайне дорогостоящая операция и не всегда осуществимая. Выражения вида $A^{-1}v$ следует понимать так: «найти решение системы линейных уравнений $Ax = v$». Поэтому вместо выражений вроде `x := := inverse(A)*v` следует писать `x := solve(A, v)`.

Если матрица A трехдиагональная, то и матрица $A - \mu E$ также является трехдиагональной. Следовательно, вектор $(A - \mu E)^{-1}v$ является решением системы уравнений с трехдиагональной матрицей. Таким образом, самым оптимальным способом вычисления выражения является *метод прогонки*.

1.2. Сеточные операторы

Методы поиска собственных векторов и собственных значений рассмотрим на примере дифференциальных операторов, которые встречаются в уравнениях математической физики. Численно можно работать только с конечномерными пространствами, поэтому сначала необходимо провести дискретизацию области, а затем выписать сеточные (дискретные) аппроксимации дифференциальных операторов.

Далее рассматриваются одномерные задачи. Разобьем отрезок $[0, l]$ на n сегментов равной длины, которые будем называть ячейками. Длину ячейки обозначим как $h = l/n$. Ячейки нумеруем целыми индексами $i = \overline{1, n}$. Центры ячеек обозначим как x_i, а границы ячеек — полуцелыми индексами $x_{i\pm1/2}$. Внутри каждой ячейки задано значение сеточной функции u_i, которое можно рассматривать как усредненное значение некоторой функции $u(x)$ внутри ячейки. Аппроксимацию операторов будем проводить методом конечных объемов, у Самарского [1] подобный метод аппроксимации встречается под названием «интегро-интерполяционный метод».

Пример 1.1. Рассмотрим дифференциальный оператор $\mathcal{L}$ отрицательной второй производной, который встречается в курсе уравнений математической физики при решении задачи Штурма–Лиувилля:

$$\mathcal{L}u = -\frac{d^2}{dx^2}u, \qquad u(x) : \begin{cases} (\alpha_L\, u - \beta_L\, u_x)\big|_{x=0} = 0, \\ (\alpha_R\, u + \beta_R\, u_x)\big|_{x=l} = 0. \end{cases}$$

Дифференциальный оператор $\mathcal{L}$ является самосопряженным и неотрицательно определенным, все его собственные значения положительны, за исключением случая $\alpha_L = \alpha_R = 0$, когда оператор имеет нулевое собственное значение.

Сеточный аналог оператора:

$$(Lu)_i = \frac{1}{h^2}\begin{cases} \dfrac{2\alpha_L h}{2\beta_L + \alpha_L h}\, u_1 + (u_1 - u_2), & i = 1, \\ (u_i - u_{i-1}) + (u_i - u_{i+1}), & i = \overline{2,\, n-1}, \\ (u_n - u_{n-1}) + \dfrac{2\alpha_R h}{2\beta_R + \alpha_R h}\, u_n, & i = n. \end{cases}$$

Сеточный оператор можно представить в виде трехдиагональной матрицы. Оператор L, как и $\mathcal{L}$, является самосопряженным (матрица симметрична) и неотрицательно определенным. Все собственные значения L являются положительными, за исключением случая $\alpha_L = \alpha_R = 0$, когда есть нулевое собственное значение.

Пример 1.2. Полиномы Лежандра можно определить как собственные функции дифференциального оператора $\mathcal{L}$:

$$\mathcal{L}u = -\frac{d}{dx}\left(1 - x^2\right)\frac{du}{dx}, \qquad \lim_{x \to \pm 1} \left(1 - x^2\right) u_x = 0.$$

Сеточный аналог оператора

$$(Lu)_i = \frac{1}{h^2}\begin{cases} \left(1 - (1-h)^2\right)(u_1 - u_2), & i = 1, \\ \xi_{i-1/2}\,(u_i - u_{i-1}) + \xi_{i+1/2}\,(u_i - u_{i+1}), & i = \overline{2,\, n-1}, \\ \left(1 - (1-h)^2\right)(u_n - u_{n-1}), & i = n. \end{cases}$$

Здесь введено обозначение $\xi_{i\pm 1/2} = 1 - x^2_{i\pm 1/2}$. Оба оператора $\mathcal{L}$ и L являются самосопряженными и неотрицательно определенными. Все собственные значения, кроме одного, положительны. Также существует нулевое собственное значение, которому соответствует вектор из одинаковых чисел.

Пример 1.3. Рассмотрим радиальную часть оператора Лапласа в полярных координатах. Дифференциальный оператор $\mathcal{L}$ определен на множестве ограниченных функций $u(r)$ на отрезке $r \in [0,\, R]$, удовлетворяющих однородным граничным условиям:

$$\mathcal{L}u = -\frac{1}{r}\frac{d}{dr} r \frac{du}{dr}, \qquad u(r) : \begin{cases} \lim\limits_{r \to 0} r u_r = 0, \\ (\alpha u + \beta u_r)\big|_{r=R} = 0. \end{cases}$$

Оператор $\mathcal{L}$ является положительно определенным, кроме случая $\alpha = 0$, когда имеется нулевое собственное значение. Также оператор $\mathcal{L}$ является самосопряженным при интегрировании с весом $\rho(r) = r$.

Для дискретизации воспользуемся методом конечных объемов. Разобьем круг $r < R$ на n концентрических колец, которые будем называть ячейками. Ячейки нумеруем целыми индексами, а границы ячеек — полуцелыми, $r_{i\pm1/2}$ — границы i-й ячейки. Ширину каждого кольца можно выбрать одинаковой: $h = R/n = r_{i+1/2} - r_{i-1/2}$. Введем «объем ячейки» v_i и «центр ячейки» r_i через интегралы по кольцу:

$$v_i = \int_{r_{i-\frac{1}{2}}}^{r_{i+\frac{1}{2}}} r\,dr = \frac{1}{2}\Big(r_{i+1/2}^2 - r_{i-1/2}^2\Big), \quad r_i = \frac{1}{v_i}\int_{r_{i-\frac{1}{2}}}^{r_{i+\frac{1}{2}}} r^2\,dr = \frac{1}{3v_i}\Big(r_{i+1/2}^3 - r_{i-1/2}^3\Big).$$

Интегрирование по углу $\int_0^{2\pi} d\varphi$ можно опустить, поскольку множитель 2π в результате сокращается во всех выражениях. Сеточная функция, как и ранее, задается путем усреднения по ячейке:

$$u_i = \frac{1}{v_i}\int_{r_{i-\frac{1}{2}}}^{r_{i+\frac{1}{2}}} u(r) r\,dr.$$

При линейной реконструкции функции внутри ячейки ее следует представлять в форме $u_j(r) = u_j + s\big(r - r_j\big)$, где s — произвольный наклон. В этом случае усреднение по j-й ячейке будет давать значение u_j, независимо от наклона s. При оценке производной на границе ячейки $r_{j+1/2}$ требуем, чтобы наклоны в соседних ячейках j и $j+1$ совпадали, тогда $s = u'(r_{j+1/2}) = \big(u_{j+1} - u_j\big)/\big(r_{j+1} - r_j\big)$. В ячейках на границе области требуем выполнения граничных условий.

Сеточный аналог оператора

$$(Lu)_i = \frac{1}{v_i}\begin{cases} r_{1+1/2}\,\dfrac{u_1 - u_2}{r_2 - r_1}, & i = 1, \\ r_{i-1/2}\,\dfrac{u_i - u_{i-1}}{|r_i - r_{i-1}|} + r_{i+1/2}\,\dfrac{u_i - u_{i+1}}{|r_{i+1} - r_i|}, & i = \overline{2,\, n-1}, \\ r_{n-1/2}\,\dfrac{u_n - u_{n-1}}{|r_n - r_{n-1}|} + \dfrac{\alpha R\, u_n}{\beta + \alpha(R - r_n)}, & i = n. \end{cases}$$

Сеточный оператор L является положительно определенным, кроме случая $\alpha = 0$, когда имеется нулевое собственное значение. Также оператор L является самосопряженным относительно скалярного произведения, введенного с билинейной формой $B = \operatorname{diag}(v_1,\, v_2,\, \ldots,\, v_n)$. Для оператора естественным образом ставится обобщенная задача на собственные значения (1.5).

Пример 1.4. Рассмотрим радиальную часть оператора Лапласа в сферических координатах. Дифференциальный оператор $\mathcal{L}$ определен на множестве ограниченных функций $u(r)$ на отрезке $r \in [0, R]$, удовлетворяющих однородным граничным условиям:

$$\mathcal{L}u = -\frac{1}{r^2}\frac{d}{dr}r^2\frac{du}{dr}, \qquad u(r) : \begin{cases} \lim\limits_{r\to 0} ru_r = 0, \\ (\alpha u + \beta u_r)\Big|_{r=R} = 0. \end{cases}$$

Оператор $\mathcal{L}$ является неотрицательно определенным и самосопряженным при интегрировании с весом $\rho(r) = r^2$.

Дискретизацию проводим на ячейки в виде сферических слоев. Аналогично плоскому случаю в полярных координатах вводим «объем ячейки» v_i и «центр ячейки» r_i через объемные интегралы по сферическому слою (угловой интеграл опускается):

$$v_i = \int_{r_{i-\frac{1}{2}}}^{r_{i+\frac{1}{2}}} r^2\,dr = \frac{1}{3}\left(r^3_{i+1/2} - r^3_{i-1/2}\right), \quad r_i = \frac{1}{v_i}\int_{r_{i-\frac{1}{2}}}^{r_{i+\frac{1}{2}}} r^3\,dr = \frac{1}{4v_i}\left(r^4_{i+1/2} - r^4_{i-1/2}\right).$$

Сеточный аналог оператора

$$(Lu)_i = \frac{1}{v_i}\begin{cases} r^2_{1+1/2}\dfrac{u_1 - u_2}{r_2 - r_1}, & i = 1, \\ r^2_{i-1/2}\dfrac{u_i - u_{i-1}}{|r_i - r_{i-1}|} + r^2_{i+1/2}\dfrac{u_i - u_{i+1}}{|r_{i+1} - r_i|}, & i = \overline{2,\, n-1}, \\ r^2_{n-1/2}\dfrac{u_n - u_{n-1}}{|r_n - r_{n-1}|} + \dfrac{\alpha R^2 u_n}{\beta + \alpha(R - r_n)}, & i = n. \end{cases}$$

Сеточный оператор L является положительно определенным, кроме случая $\alpha = 0$, когда имеется нулевое собственное значение. Также оператор L является самосопряженным относительно скалярного произведения, введенного с билинейной формой $B = \operatorname{diag}(v_1, v_2, \ldots, v_n)$. Для оператора естественным образом ставится обобщенная задача на собственные значения (1.5).

1.3. Обобщенная задача на собственные значения

Пример 1.5. Рассмотрим дифференциальный оператор $\mathcal{P}$ отрицательной второй производной с весовой функцией $\rho(x) > 0$:

$$\mathcal{P}u = -\frac{1}{\rho(x)}\frac{d^2}{dx^2}u, \qquad u(x) : \begin{cases} (\alpha_L\, u - \beta_L\, u_x)\Big|_{x=0} = 0, \\ (\alpha_R\, u + \beta_R\, u_x)\Big|_{x=l} = 0. \end{cases}$$

Помимо классической задачи на собственные значения $\mathcal{P}u = \lambda u$, можно поставить эквивалентную ей *обобщенную задачу на собственные значения*:

$$\mathcal{L}u = \lambda \rho u,$$

где $\mathcal{L}$ — дифференциальный оператор второй производной, введенный в примере 6.1. Обобщенная постановка корректна, даже если $\rho(x)$ — обобщенная функция; к примеру, если весовая функция $\rho(x)$ включает δ-функцию Дирака. Дифференциальный оператор $\mathcal{P}$ является самосопряженным при интегрировании с весовой функцией $\rho(x)$ и положительно определенным, кроме случая $\alpha_L = \alpha_R = 0$, когда существует нулевое собственное значение.

Построим сеточный аналог оператора $\mathcal{P}$. Для этого усредним весовую функцию $\rho(x)$ в ячейках:

$$\rho_i = \frac{1}{h} \int_{x_{i-\frac{1}{2}}}^{x_{i+\frac{1}{2}}} \rho(x)\, dx.$$

Теперь поделим каждую строку оператора второй производной L из примера 6.1 на плотность ρ_i:

$$(Pu)_i = \frac{1}{\rho_i h^2} \begin{cases} \dfrac{2\alpha_L h}{2\beta_L + \alpha_L h} u_1 + (u_1 - u_2), & i = 1, \\ (u_i - u_{i-1}) + (u_i - u_{i+1}), & i = \overline{2,\, n-1}, \\ (u_n - u_{n-1}) + \dfrac{2\alpha_R h}{2\beta_R + \alpha_R h} u_n, & i = n. \end{cases}$$

При более строгом выводе все функции и уравнения следует интегрировать с весом $\rho(x)$. Тогда в формулах появятся центры масс ячеек $x_i = \frac{1}{h}\int x\rho(x)dx$. Но не будем чрезмерно усложнять. Оператор P неотрицательно определен, все его собственные значения положительны, за исключением случая $\alpha_L = \alpha_R = 0$, когда имеется нулевое собственное значение.

Как это можно использовать? Введем матрицу с весами на диагонали: $B = \mathrm{diag}(\rho_1,\, \rho_2,\, \ldots,\, \rho_n)$. Тогда матрица оператора $P = B^{-1}L$, где L — сеточный оператор второй производной из примера 6.1. Матрица L симметричная и неотрицательно определенная. Обобщенная задача на собственные значения для сеточного оператора:

$$Lu = \lambda B u. \tag{1.5}$$

Матрица B симметричная и положительно определенная, а значит, можно использовать матрицу B как билинейную форму и ввести скалярное произведение: $(x,\, y)_B = x^T B y$. В линейном пространстве с таким скалярным произведением оператор P является самосопряженным $(Px,\, y)_B = (x,\, Py)_B$.

Собственные значения оператора P вещественны и неотрицательны, а собственные векторы, соответствующие различным собственным значениям, ортогональны: $(x, y)_B = 0$.

Это хорошо, что задачу удалось свести к работе с симметричными матрицами. Многие алгоритмы эффективнее работают с симметричными матрицами. В данном случае, вместо решения классической задачи на собственные значения $Pu = \lambda u$, где P — матрица общего вида, а спектр матрицы ищется в поле комплексных чисел, можно перейти к обобщенной задаче на собственные значения. Для обобщенной задачи с симметричными матрицами спектр является вещественным.

Если говорить о практическом применении данных знаний: многие библиотеки включают специализированные функции для работы с эрмитовыми матрицами. К примеру, модуль `scipy.sparse.linalg` в Python содержит отдельные функции для поиска собственных значений для матриц общего вида (функция `eigs`) и для поиска собственных значений для эрмитовых матриц, в том числе для обобщенной задачи на собственные значения (функция `eigsh`). При известных ограничениях на матрицу и ее спектр целесообразно использовать специализированную функцию `eigsh`.

1.4. Спектр трехточечного оператора второй производной

Рассмотрим задачу на СВ и СЗ для трехточечного сеточного оператора отрицательной второй производной $\mathcal{L} = -d^2/dx^2$. Полагаем, что сеточный оператор L получен методом конечных объемов на равномерной сетке в области $0 \le x \le l$. Область разделена на n ячеек длиной $h = l/n$. Ячейки пронумерованы целыми индексами $i = \overline{1, n}$, центры ячеек $x_i = (i - \frac{1}{2})h$, а значения сеточной функции v_i соответствуют ячейкам.

На границах области заданы граничные условия одного из трех типов: условия Дирихле (D), условия Неймана (N) или периодические граничные условия (P). Матрица оператора L имеет размер $n \times n$ и для различных типов граничных условий принимает вид:

$$L_D = \frac{1}{h^2}\begin{pmatrix} 3 & -1 & 0 & 0 & \cdots & 0 \\ -1 & 2 & -1 & 0 & \cdots & 0 \\ 0 & -1 & 2 & -1 & \cdots & 0 \\ & \cdots & & & \cdots & \\ 0 & \cdots & 0 & -1 & 2 & -1 \\ 0 & \cdots & 0 & 0 & -1 & 3 \end{pmatrix}, \qquad L_N = \frac{1}{h^2}\begin{pmatrix} 1 & -1 & 0 & 0 & \cdots & 0 \\ -1 & 2 & -1 & 0 & \cdots & 0 \\ 0 & -1 & 2 & -1 & \cdots & 0 \\ & \cdots & & & \cdots & \\ 0 & \cdots & 0 & -1 & 2 & -1 \\ 0 & \cdots & 0 & 0 & -1 & 1 \end{pmatrix},$$

$$L_P = \frac{1}{h^2}\begin{pmatrix} 2 & -1 & 0 & 0 & \cdots & -1 \\ -1 & 2 & -1 & 0 & \cdots & 0 \\ 0 & -1 & 2 & -1 & \cdots & 0 \\ & \cdots & & & \cdots & \\ 0 & \cdots & 0 & -1 & 2 & -1 \\ -1 & \cdots & 0 & 0 & -1 & 2 \end{pmatrix}.$$

При действии любым из перечисленных операторов на сеточную функцию для внутренних ячеек получаем аппроксимацию отрицательной второй про-

изводной на трех точках:

$$(Lv)_i = -\frac{v_{i+1} - 2v_i + v_{i-1}}{h^2}, \qquad i = \overline{2,\, n-1}.$$

Все три матрицы являются симметричными. Матрица L_D — положительно определенная. Матрицы L_N и L_D неотрицательно определены, вектор из одинаковых чисел соответствует нулевому собственному значению. Для всех трех матриц можно выписать полный набор собственных значений и собственных векторов.

СВ и СЗ для задачи Дирихле (для матрицы L_D):

$$\lambda_k = \frac{4}{h^2}\sin^2\frac{\pi k h}{2l}, \qquad v_i^{(k)} = \sin\frac{\pi k x_i}{l}, \qquad k = \overline{1,\, n}. \tag{1.6}$$

Если дискретизация содержит хотя бы три ячейки ($n \geq 3$), то минимальное собственное значение $\lambda_{\min}$ можно оценить снизу, а для спектра оператора справедливы оценки:

$$\lambda_{\min} = \frac{4}{h^2}\sin^2\frac{\pi h}{2l}, \qquad \frac{9}{l^2} \leq \lambda_{\min} \leq \lambda_k \leq \lambda_{\max} = \frac{4}{h^2}. \tag{1.7}$$

СВ и СЗ для задачи Неймана (для матрицы L_N):

$$\lambda_k = \frac{4}{h^2}\sin^2\frac{\pi k h}{2l}, \qquad v_i^{(k)} = \cos\frac{\pi k x_i}{l}, \qquad k = \overline{0,\, n-1}. \tag{1.8}$$

Если дискретизация содержит хотя бы три ячейки ($n \geq 3$), то минимальное положительное собственное значение λ_1 можно оценить снизу, а для спектра справедливы оценки:

$$\lambda_0 = \lambda_{\min} = 0, \qquad \frac{9}{l^2} \leq \lambda_1 \leq \lambda_k \leq \lambda_{\max} < \frac{4}{h^2}. \tag{1.9}$$

СВ и СЗ для задачи с периодическими граничными условиями (для матрицы L_P):

$$\begin{aligned} \lambda_k &= \frac{4}{h^2}\sin^2\frac{\pi(k+b)h}{2l}, \qquad b = k \bmod 2, \\ v_i^{(k)} &= \cos\left(\frac{\pi x_i (k+b)}{l} - \frac{\pi b}{2}\right), \end{aligned} \tag{1.10}$$

где $k = \overline{0,\, n-1}$, и $k \bmod 2$ — остаток от деления на два. Положительные собственные значения имеют кратность равную двум, а среди собственных векторов чередуются четные (cos) и нечетные (sin) функции. Использование операции $k \bmod 2$ позволяет объединить две серии собственных векторов.

Если дискретизация содержит хотя бы три ячейки ($n \geq 3$), то минимальное положительное собственное значение λ_1 можно оценить снизу, а для спектра оператора справедливы оценки (1.9). В целом, независимо от гранич-

ных условий, спектры трех операторов практически полностью совпадают. Для вырожденных операторов L_N и L_P добавляются нулевые собственные значения.

1.5. Лабораторная работа

В лабораторной работе требуется численно найти 5 – 6 собственных функций для дифференциального оператора, которые соответствуют минимальным собственным значениям. Для поиска использовать обратный степенной метод или метод Рэлея, начальные приближения собственных значений подбираются вручную. При выполнении работы допускается использование встроенных решателей систем линейных уравнений для вычисления векторов $A^{-1}b$. Построить на одном графике приближенные собственные функции и аналитические. Список задач приведен в прил. П.1.

Необязательные пункты:

- вычислить погрешность и оценить порядок аппроксимации;
- использовать функцию `scipy.linalg.solve_banded` для решения системы уравнений с трёхдиагональной матрицей;
- использовать разреженные матрицы `scipy.sparse` и найти решение на сетке с размером от 5 тыс. ячеек;
- создать разреженные матрицы без циклов;
- получить собственные функции и собственные значения с использованием встроенной функции `scipy.linalg.sparse.eigsh`.

Лекция 2. Аппроксимация уравнения Пуассона

Уравнение Пуассона в многомерной области D имеет вид

$$-\Delta u = f(\boldsymbol{r}), \qquad \boldsymbol{r} \in D \subseteq \mathbb{R}^d,$$

где Δ — оператор Лапласа (или лапласиан). Важный частный случай $f(\boldsymbol{r}) = 0$ носит собственное название — уравнение Лапласа. Уравнение Пуассона или Лапласа обычно дополняется тем или иным условием на границе области D, которую будем обозначать как ∂D. К классическим граничным условиям можно отнести условие Дирихле, когда на границе задано значение функции, и условие Неймана, когда на границе задана производная функции.

Уравнение Пуассона встречается в различных областях физики. Собственно, уравнение Пуассона в электростатике, стационарные задачи теплопроводности, диффузии и фильтрации. Уравнение Пуассона также описывает натяжение двумерной мембраны, потенциальное течение несжимаемой жидкости. Общность всех этих физических процессов проявляется в интегральной формулировке уравнения

$$-\oint_{\partial D} \frac{\partial u}{\partial n}\, dS = \int_D f(\boldsymbol{r})\, dV, \tag{2.1}$$

которая выражает фундаментальные законы сохранения. Поверхностный интеграл равен «потоку» некоторой величины через границы области, а объемный интеграл — «источникам» этой величины внутри области. Интегральное соотношение описывает баланс между генерацией некоторой величины внутри области и потоком этой величины через границы.

Для электростатики $(-\boldsymbol{\nabla} u)$ — это поток электрической индукции, а источниками $f(\boldsymbol{r})$ являются электрические заряды. Для теплопроводности $(-\boldsymbol{\nabla} u)$ — это поток тепла (закон Фурье). Для диффузии и фильтрации — поток вещества (закон Дарси). Для движения несжимаемой жидкости — это скорость $\boldsymbol{v} = -\boldsymbol{\nabla} u$, т.е. поток жидкости.

Уравнение Пуассона относится к дифференциальным уравнениям эллиптического типа. Численная аппроксимация краевых задач для уравнения Пуассона приводит к системам линейных уравнений. Если задача Коши для эволюционных гиперболических или параболических уравнений допускает решение по явным схемам, т.е. численное решение можно находить, последовательно продвигаясь по временным слоям, то численное решение краевых задач для эллиптических уравнений описывается исключительно неявными схемами.

Выпишем конечно-объемную аппроксимацию уравнения Пуассона в прямоугольной области:

$$-\Delta u = f(x,\, y), \qquad 0 < x < l_x, \quad 0 < y < l_y. \tag{2.2}$$

Разобьем вычислительную область на одинаковые прямоугольные ячейки: n_x ячеек вдоль оси x и n_y ячеек вдоль оси y, общее число ячеек $n = n_x \times n_y$. Размеры ячеек обозначим как $h_x = l_x/n_x$ и $h_y = l_y/n_y$. Для обозначения ячеек и значений в них введем двумерную индексацию, индекс ячейки по оси x обозначим как $i = \overline{1,\, n_x}$, вдоль оси y как $j = \overline{1,\, n_y}$. Координаты центров ячеек обозначим как x_{ij} и y_{ij}, а координаты границ ячеек будем обозначать полуцелыми индексами $x_{i\pm1/2}$ и $y_{j\pm1/2}$.

Интегральное соотношение (2.1) справедливо для любой подобласти области D, поэтому его можно выписать отдельно для каждой ячейки построенной сетки:

$$-\oint_{S_{ij}} \frac{\partial u}{\partial n}\, dS = \int_{V_{ij}} f(x,\, y)\, dV,$$

здесь V_{ij} — объем ячейки, S_{ij} — поверхность ячейки ij, а производные вычисляются по направлению внешней нормали к границе ячейки. В нашем случае каждая ячейка имеет ровно по четыре грани, выпишем поверхностный интеграл отдельно для каждой грани:

$$-\int_{S^L_{ij}} \frac{\partial u}{\partial n}\, dS - \int_{S^R_{ij}} \frac{\partial u}{\partial n}\, dS - \int_{S^B_{ij}} \frac{\partial u}{\partial n}\, dS - \int_{S^T_{ij}} \frac{\partial u}{\partial n}\, dS = \int_{V_{ij}} f(x, y)\, dV. \tag{2.3}$$

В данной формуле S^L_{ij}, S^R_{ij}, S^B_{ij} и S^T_{ij} — поверхности левой, правой, нижней и верхней граней ячейки соответственно. Уравнение (2.3) выполняется **точно**, на данном этапе никакие приближения не использовались.

Введем операцию проекции функции $u(x,\, y)$ на сетку. Сеточным значением функции u_{ij} в ячейке будем считать усредненное значение функции в объеме ячейки V_{ij}, т.е.

$$u_{ij} = \frac{1}{V_{ij}} \int_{V_{ij}} u(x,\, y)\, dV.$$

Такой способ проекции со вторым порядком точности совпадает с выбором значений сеточной функции в центре ячейки, т.е.

$$u_{ij} = u\big(x_{ij},\, y_{ij}\big) + O(h^2),$$

где h — характерный линейный размер ячейки. Для ячеек произвольной формы значение функции выбирается в барицентре ячейки. Таким образом, при использовании схем с порядком точности не выше второго, можно считать оба варианта проекции эквивалентными: значения u_{ij} можно рас-

сматривать как значение функции в центре ячейки или как усредненное по ячейке. После введения сеточной функции можно заменить объемный интеграл в правой части (2.3) на значение $h_x h_y f_{ij}$, где $h_x h_y$ — объем ячейки.

Далее необходимо аппроксимировать потоковые интегралы в левой части уравнения (2.3). Для примера, рассмотрим поток через левую грань ячейки:

$$\int_{S_{ij}^L} \frac{\partial u}{\partial n}\, dS = \frac{u_{i-1,j} - u_{ij}}{h_x}\, h_y + O(h^2),$$

здесь аппроксимация производной на грани умножается на длину грани h_y. После записи всех интегралов получаем следующую формулу для внутренних ячеек области:

$$\frac{u_{ij} - u_{i-1,j}}{h_x}\, h_y + \frac{u_{ij} - u_{i+1,j}}{h_x}\, h_y + \frac{u_{ij} - u_{i,j-1}}{h_y}\, h_x + \frac{u_{ij} - u_{i,j+1}}{h_y}\, h_x = h_x h_y f_{ij}.$$

После преобразований получаем формулу, в которой без труда различимы аппроксимации вторых производных в разных направлениях:

$$-\frac{u_{i+1,j} - 2u_{ij} + u_{i-1,j}}{h_x^2} - \frac{u_{i,j+1} - 2u_{ij} + u_{i,j-1}}{h_y^2} = f_{ij}. \tag{2.4}$$

Формула (2.4) справедлива для внутренних ячеек области $1 < i < n_x$ и $1 < j < n_y$. Для записи уравнений для граничных ячеек области требуется задание граничных условий.

2.1. Периодические граничные условия

Уравнение Пуассона задано в прямоугольной области, левая граница связана с правой, а нижняя граница замыкается на верхнюю, будто уравнение решается в бесконечной области.

$$\begin{cases} \Delta u = -f(x,\, y), \qquad 0 < x < l_x, \quad 0 < y < l_y, \\ u(x + l_x,\, y) = u(x,\, y), \\ u\big(x,\, y + l_y\big) = u(x,\, y). \end{cases}$$

Решение уравнения Пуассона с периодическими граничными условиями существует при выполнении условия

$$\int_D f(x,\, y)\, dx\, dy = 0. \tag{2.5}$$

Чтобы в этом убедиться, достаточно проинтегрировать уравнение по всей области. Потоки через левую и правую границы области различаются по знаку и взаимно вычитаются, аналогично потоки через верхнюю и нижнюю границы. Отсюда следует, что сумма источников внутри области должна быть

равна нулю. Если решение уравнения Пуассона с периодическими граничными условиями существует, то оно определено с точностью до произвольной аддитивной постоянной.

Конечно-объемная аппроксимация для всех ячеек определяется по формуле (2.4). Необходимо только учесть, что индексы ячеек также требуется периодически замкнуть. К примеру, справа от самой правой ячейки $u_{n_x,j}$ располагается ячейка с левой границы $u_{1,j}$. Снизу от нижнего ряда ячеек $u_{i,1}$ располагаются ячейки с верхнего ряда u_{i,n_y}.

Итоговую систему линейных уравнений можно представить в операторном виде $Lu = F$, где L — сеточный оператор, а F — вектор правых частей. Количество уравнений в системе равно числу ячеек n, соответственно, размер оператора L в матричном представлении $n \times n$. Оператор L является вырожденным, поскольку вектор из одинаковых чисел удовлетворяет однородной системе уравнений $Lu = 0$, в этом смысле оператор L повторяет свойства дифференциального оператора.

Условие совместности для систем линейных уравнений (теорема Кронекера – Капелли) гласит, что решение системы существует в том случае, когда ранг матрицы совпадает с рангом расширенной матрицы. Поскольку ранг матрицы L меньше числа неизвестных n, для разрешимости должно быть наложено условие и на правую часть системы — на вектор F. По невероятному стечению обстоятельств условие совместности полученной системы линейных уравнений совпадает с условием существования решения (2.5) дифференциальной задачи[1]:

$$h_x h_y \sum_{i,\,j} F_{ij} = \sum_{i,\,j} f_{ij}\, h_x h_y = 0.$$

Если значения сеточной функции f_{ij} получены путем усреднения в ячейках, то условие совместности для системы уравнений полностью совпадает с условием разрешимости дифференциальной задачи. Но чаще используется аппроксимация $f_{ij} = f(x_i,\, y_j)$, в этом случае вектор правых частей требуется отнормировать, чтобы сумма стала равна нулю, иначе система уравнений не будет иметь решения. Для этого достаточно вычесть из правых частей среднее значение:

$$\tilde{F}_{ij} = F_{ij} - \frac{1}{n} \sum_{i,\,j} F_{ij}.$$

Поскольку решение определяется с точностью до произвольной постоянной, решение также рекомендуется нормировать аналогичным образом, в том числе в ходе итераций, при использовании итерационных методов решения систем уравнений.

---

[1] Вектор из одинаковых чисел удовлетворяет однородной системе уравнений ⇒ сумма столбцов матрицы равна нулю. Матрица симметрична ⇒ сумма строк матрицы равна нулю ⇒ сумма строк расширенной матрицы должна быть равна нулю, т.е. $\sum_{ij} F_{ij} = 0$.

2.2. Граничные условия Неймана

В задаче Неймана для уравнения Пуассона на границах области задаются потоки (нормальные производные):

$$\begin{cases} \Delta u = -f(x,\, y), & 0 < x < l_x, \quad 0 < y < l_y, \\ u_x(0,\, y) = -\varphi_L(y), & u_x(l_x,\, y) = \varphi_R(y), \\ u_y(x,\, 0) = -\varphi_B(x), & u_y\big(x,\, l_y\big) = \varphi_T(x). \end{cases}$$

В данной постановке перед функцией φ_L указан знак минус, поскольку на левой границе производная по внешней нормали равна $-u_x$, по аналогичной причине минус стоит перед функцией φ_B. Решение задачи Неймана существует при выполнении условия

$$\int_D f\, dV + \oint_{\partial D} \varphi\, dS = 0, \tag{2.6}$$

т.е. сумма источников плюс сумма потоков через границу области должна быть равна нулю. Условие совместности имеет самую наглядную интерпретацию для задачи движения *несжимаемой* жидкости. Если источники жидкости в области отсутствуют, то сумма втекающих и вытекающих потоков жидкости должна быть равна нулю. Если в области есть источники жидкости, то вся создаваемая жидкость должна вытекать через границу. Решение задачи Неймана определено с точностью до аддитивной постоянной.

Выпишем точное равенство (2.3). Пусть теперь одна из сторон ячейки располагается на границе области, в этом случае условия Неймана напрямую определяют поток через внешнюю границу. К примеру, пусть левая сторона ячейки лежит на границе области

$$\int_{S^L_{ij}} \frac{\partial u}{\partial n}\, dS = \int_{S^L_{ij}} \varphi_L\, dS = \int_{y_{j-\frac{1}{2}}}^{y_{j+\frac{1}{2}}} \varphi_L(y)\, dy = \varphi_{Lj} h_y.$$

Опять же по умолчанию, величину φ_{Lj} вводим как усредненное значение функции φ_L на грани ячейки, т.е., по определению,

$$\varphi_{Lj} = \frac{1}{h_y} \int_{y_{j-\frac{1}{2}}}^{y_{j+\frac{1}{2}}} \varphi_L(y)\, dy = \varphi_L\big(y_j\big) + O(h_y^2),$$

но при работе со схемами не выше второго порядка можно ограничиться аппроксимацией функции φ_L на центре грани.

Выпишем полученную схему целиком:

$$[i > 1]\, \frac{u_{ij} - u_{i-1,j}}{h_x}\, h_y - [i = 1]\, \varphi_{Lj}\, h_y + [i < n_x]\, \frac{u_{ij} - u_{i+1,j}}{h_x}\, h_y -$$

$$- [i = n_x]\,\varphi_{Rj}\,h_y + [j > 1]\,\frac{u_{ij} - u_{i,j-1}}{h_y}\,h_x - [j = 1]\,\varphi_{Bi}\,h_x +$$
$$+ \left[j < n_y\right]\frac{u_{ij} - u_{i,j+1}}{h_y}\,h_x - \left[j = n_y\right]\varphi_{Ti}\,h_x = h_x h_y f_{ij}.$$

Здесь использовано обозначение $[\,.\,]$ — скобка Айверсона. Если в скобках указано истинное выражение, скобка принимает значение 1, иначе — 0. Такая запись справедлива для всех ячеек и позволяет учесть граничные условия для ячеек, для которых они существуют.

Перенесем неоднородность в правую часть и поделим выражение на объем ячейки $h_x h_y$:

$$[i > 1]\,\frac{u_{ij} - u_{i-1,j}}{h_x^2} + [i < n_x]\,\frac{u_{ij} - u_{i+1,j}}{h_x^2} +$$
$$+ [j > 1]\,\frac{u_{ij} - u_{i,j-1}}{h_y^2} + \left[j < n_y\right]\frac{u_{ij} - u_{i,j+1}}{h_y^2} =$$
$$= f_{ij} + [i = 1]\,\frac{\varphi_{Lj}}{h_x} + [i = n_x]\,\frac{\varphi_{Rj}}{h_x} + [j = 1]\,\frac{\varphi_{Bi}}{h_y} + \left[j = n_y\right]\frac{\varphi_{Ti}}{h_y}.$$

Здесь в левой части формулы записан результат действия сеточного оператора Лапласа на сеточную функцию, справа — сумма всех неоднородностей уравнения. Можно заметить, что потоки с границы входят в уравнение так же, как объемные источники.

Итоговую систему линейных уравнений представим в операторном виде $Lu = F$, где L — сеточный оператор , а F — вектор правых частей. Количество уравнений в системе равно числу ячеек n, соответственно, размер оператора L в матричном представлении $n \times n$. Оператор L является вырожденным, поскольку вектор из одинаковых чисел удовлетворяет однородной системе уравнений $Lu = 0$, в этом смысле оператор L повторяет свойства дифференциального оператора.

Условие совместности для систем линейных уравнений (теорема Кронекера – Капелли) гласит, что решение системы существует в том случае, когда ранг матрицы совпадает с рангом расширенной матрицы. Поскольку ранг матрицы L меньше числа неизвестных n, для разрешимости должно быть наложено условие и на правую часть системы — на вектор F. Решение системы линейных уравнений существует в том случае, если сумма значений F_{ij} равна нулю. По невероятному стечению обстоятельств условие совместности полученной системы уравнений совпадает с условием существования решения (2.6) дифференциальной задачи:

$$h_x h_y \sum_{i,\,j} F_{ij} = \sum_{i,\,j} f_{ij} h_x h_y + \sum_j \varphi_{Lj} h_y + \sum_j \varphi_{Rj} h_y + \sum_i \varphi_{Bi} h_x + \sum_i \varphi_{Ti} h_x = 0.$$

Если значения сеточной функции f_{ij} получены путем усреднения в ячейках, а значения φ_{Lj}, φ_{Rj} и другие путем усреднения потоков на гранях, то условие существования решения для системы линейных уравнений полностью совпадает с условием разрешимости дифференциальной задачи. Но чаще используется аппроксимация в центрах ячеек и граней, поэтому перед решением СЛАУ правую часть следует нормировать, чтобы сумма правых частей была равна нулю, иначе система уравнений не будет иметь решения. Для этого достаточно вычесть среднее значение:

$$\tilde{F}_{ij} = F_{ij} - \frac{1}{n}\sum_{i,\,j} F_{ij}.$$

Поскольку решение определяется с точностью до произвольной постоянной, решение также рекомендуется нормировать аналогичным образом, в том числе в ходе итераций, при использовании итерационных методов решения систем уравнений.

2.3. Граничные условия Дирихле

В задаче Дирихле на границе области задаются точные значения:

$$\begin{cases} \Delta u = -f(x,\,y), & 0 < x < l_x, \quad 0 < y < l_y, \\ u(0,\,y) = \mu_L(y), & u(l_x,\,y) = \mu_R(y), \\ u(x,\,0) = \mu_B(x), & u\big(x,\,l_y\big) = \mu_T(x). \end{cases}$$

Вернемся к точному равенству (2.3), требуется оценить производные на гранях, которые являются границей области. Пусть значение в самой ячейке равно u_k, значение на грани в соответствии с условием Дирихле задано и равно μ_k, тогда производную по внешней нормали можно оценить по формуле

$$\frac{\partial u}{\partial n} \approx \frac{\mu_k - u_k}{h/2} = \frac{2\mu_k - 2u_k}{h},$$

здесь $h/2$ — расстояние от центра ячейки до соответствующей грани. Подставим выражения в аппроксимации производных и получим формулу

$$\begin{aligned} [i>1]\,\frac{u_{ij}-u_{i-1,j}}{h_x}\,h_y + [i=1]\,\frac{u_{ij}-\mu_{Lj}}{h_x/2}\,h_y + \\ + [i<n_x]\,\frac{u_{ij}-u_{i+1,j}}{h_x}\,h_y + [i=n_x]\,\frac{u_{ij}-\mu_{Rj}}{h_x/2}\,h_y + \\ + [j>1]\,\frac{u_{ij}-u_{i,j-1}}{h_y}\,h_x + [j=1]\,\frac{u_{ij}-\mu_{Bi}}{h_y/2}\,h_x + \\ + \big[j<n_y\big]\,\frac{u_{ij}-u_{i,j+1}}{h_y}\,h_x + \big[j=n_y\big]\,\frac{u_{ij}-\mu_{Ti}}{h_y/2}\,h_x = h_x h_y f_{ij}. \end{aligned}$$

Разделим выражение на объем ячейки $h_x h_y$, также перенесем все неоднородные слагаемые в правую часть, итоговая аппроксимация

$$[i>1]\frac{u_{ij}-u_{i-1,j}}{h_x^2}+[i=1]\frac{2u_{ij}}{h_x^2}+[i<n_x]\frac{u_{ij}-u_{i+1,j}}{h_x^2}+[i=n_x]\frac{2u_{ij}}{h_x^2}+$$
$$+[j>1]\frac{u_{ij}-u_{i,j-1}}{h_y^2}+[j=1]\frac{2u_{ij}}{h_y^2}+\big[j<n_y\big]\frac{u_{ij}-u_{i,j+1}}{h_y^2}+\big[j=n_y\big]\frac{2u_{ij}}{h_y^2}=$$
$$=f_{ij}+[i=1]\frac{2\mu_{Lj}}{h_x^2}+[i=n_x]\frac{2\mu_{Rj}}{h_x^2}+[j=1]\frac{2\mu_{Bi}}{h_y^2}+\big[j=n_y\big]\frac{2\mu_{Ti}}{h_y^2}.$$

Как и в прошлых задачах, уравнение можно представить в операторном виде $Lu=F$, где L — сеточный оператор, а F — вектор правых частей. Количество уравнений в системе равно числу ячеек n, соответственно, размер оператора A в матричном представлении $n\times n$. Решение задачи Дирихле для уравнения Пуассона всегда существует и единственно. Система линейных уравнений для аппроксимации также всегда совместна, т.е. матрица оператора L не вырождена.

Несколько слов о том, как понимать постановки выше. Введем n-мерное линейное пространство сеточных функций $\mathbb{U}=\mathbb{R}^n$. Размерность пространства совпадает с числом ячеек нашей сетки. Все операции в линейном пространстве $\mathbb{U}$ можно определить стандартным образом, сеточные функции можно складывать, умножать на число, считать их скалярное произведение и норму. Выше мы записывали дискретные задачи в «операторной» форме $Lu=F$. Здесь u и F — сеточные функции из $\mathbb{U}$, а L — линейный оператор на пространстве сеточных функций $L:\ \mathbb{U}\to\mathbb{U}$.

В матричном представлении сеточные функции — это просто векторы длины n, а оператор L — квадратная матрица $n\times n$. Для перевода уравнения $Lu=F$ в матричное представление требуется всего лишь пронумеровать ячейки сетки. В дискретных уравнениях выше используется двойная индексация u_{ij} и F_{ij} для сеточной функции в прямоугольнике, если же для ячеек определить единственный индекс, то уравнения будут в точности выглядеть, как система линейных уравнений. Понятно, что свойства оператора L не меняются при различной нумерации, сеточный оператор для $-\Delta$ останется самосопряженным и неотрицательно определенным, а в матричном представлении матрица L будет симметричной.

Пример 2.1. Рассмотрим уравнение Пуассона в прямоугольнике со всеми видами граничных условий: условие Дирихле слева, условие Неймана справа, периодическое замыкание снизу/сверху:

$$\begin{cases}\Delta u=-f(x,\,y), \quad 0<x<l_x, \quad 0<y<l_y,\\ u(0,\,y)=\mu(y), \quad u_x(l_x,\,y)=\varphi(y),\\ u(x,\,y+l_y)=u(x,\,y).\end{cases}$$

Разобьем расчетную область на ячейки, наиболее близкие к квадратам со стороной h_0. Тогда число ячеек по осям должно быть равно $n_x = \mathrm{round}(l_x/h_0)$, $n_y = \mathrm{round}(l_y/h_0)$, размеры получившихся ячеек $h_x = l_x/n_x$, $h_y = l_y/n_y$. Ячейки нумеруем парой индексов $(i,\ j)$, координаты центров ячеек:

$$x_{ij} = \left(i - \tfrac{1}{2}\right)h_x, \quad y_{ij} = \left(j - \tfrac{1}{2}\right)h_y, \quad i = \overline{1,\ n_x},\ j = \overline{1,\ n_y}. \tag{2.7}$$

В листинге 2.1 приведен пример создания двумерной сетки с использованием функций `numpy`. Функция `numpy.meshgrid` создает из пары одномерных массивов двумерные массивы `X` и `Y`, которые соответствуют переменным x_{ij} и y_{ij} в математической записи выше (2.7). Разница лишь в том, что для массивов в Python используется индексация с нуля, в то время как в математической записи используется нумерация с единицы.

Листинг 2.1. Создание двумерной сетки `X`, `Y`

```
import numpy as np

# characteristic size
h0 = 0.01

# mesh sizes
nx = int(np.round(lx / h0))
ny = int(np.round(ly / h0))

# cell sizes
hx = lx / nx
hy = ly / ny

# cell centers
x = np.linspace(0.5 * hx, lx - 0.5 * hx, nx)
y = np.linspace(0.5 * hy, ly - 0.5 * hy, ny)

X, Y = np.meshgrid(x, y, indexing='ij')
```

Выпишем конечно-объемную аппроксимацию уравнения в форме $Lu = F$. Оператор L имеет следующую форму:

$$\begin{aligned}(Lu)_{ij} = [i > 1]\,\frac{u_{ij} - u_{i-1,j}}{h_x^2} + [i < n_x]\,\frac{u_{ij} - u_{i+1,j}}{h_x^2} + \\ + [j > 1]\,\frac{u_{ij} - u_{i,j-1}}{h_y^2} + [j < n_y]\,\frac{u_{ij} - u_{i,j+1}}{h_y^2} + \\ + [i = 1]\,\frac{2u_{ij}}{h_x^2} + [j = 1]\,\frac{u_{ij} - u_{i,n_y}}{h_y^2} + [j = n_y]\,\frac{u_{ij} - u_{i,1}}{h_y^2},\end{aligned} \tag{2.8}$$

последние два слагаемых отвечают за периодическое замыкание.

Сеточная функция F:

$$F_{ij} = f_{ij} + [i = 1]\,\frac{2\mu_j}{h_x^2} + [i = n_x]\,\frac{\varphi_j}{h_x}. \tag{2.9}$$

Пусть сеточные функции в программе задаются в виде двумерных numpy массивов $n_x \times n_y$. В листинге 2.2 приведены две функции: функция RHS (right-hand side) возвращает сеточную функцию правых частей F, функция OpL возвращает результат действия оператора L на сеточную функцию u. Синтаксис срезов позволяет выполнять быстрые векторные операции над двумерными массивами. Математическая запись операторов через скобки Айверсона естественным образом переходит в код на Python, каждое слагаемое в формулах (2.8) и (2.9) соответствует одной строке кода.

Листинг 2.2. Правые части F; действие оператора L на сеточную функцию u

```
def RHS(X, Y):
    res = f(X, Y)
    res[ 0, :] += 2*mu (Y[ 0, :]) / hx**2
    res[-1, :] +=   phi(Y[-1, :]) / hx
    return res

def OpL(u):
    res = np.zeros_like(u)
    res[+1:, :] += (u[+1:, :] - u[:-1, :]) / hx**2
    res[:-1, :] += (u[:-1, :] - u[+1:, :]) / hx**2
    res[:, +1:] += (u[:, +1:] - u[:, :-1]) / hy**2
    res[:, :-1] += (u[:, :-1] - u[:, +1:]) / hy**2

    # Dirichlet boundary condition
    res[0, :] += 2.0 * u[0, :] / hx**2

    # Periodic boundary condition
    res[:,  0] += (u[:, 0] - u[:, -1]) / hy**2
    res[:, -1] += (u[:, -1] - u[:, 0]) / hy**2
    return res
```

Лекция 3. Итерационные методы решения СЛАУ

Данная лекция носит исключительно теоретический характер. Ее можно прочитать студентам ближе к концу курса, после лекций 4 и 5, которые содержат прямые указания к выполнению лабораторных работ. В лекции представлено сокращенное изложение теории из учебника Самарского и Гулина [1].

3.1. Прямые и итерационные методы решения СЛАУ

Рассмотрим систему линейных алгебраических уравнений (СЛАУ) в матричной форме:

$$Ax = f. \tag{3.1}$$

Будем полагать, что x и f — векторы длины n, а матрица A — квадратная невырожденная матрица $n \times n$.

Прямые методы решения СЛАУ позволяют получить решение системы за конечное количество арифметических операций, при этом, если все операции выполняются точно (без ошибок округления), то решение системы также получается точным. К прямым методам решения СЛАУ относятся методы Крамера, Гаусса, Холецкого, метод прогонки и др. Большинство прямых методов подобны методу Гаусса и основаны на приведении матрицы A к треугольному виду. Многие методы адаптированы или содержат оптимизации для матриц специального вида, к примеру, разложение Холецкого применяется к симметричным положительно определенным матрицам, а метод прогонки — к трехдиагональным матрицам.

К недостаткам прямых методов относится большая вычислительная сложность. Так, для решения системы n уравнений классическим методом Гаусса требуется выполнить $O(n^3)$ арифметических операций. Другие прямые методы имеют сопоставимую вычислительную сложность. Также прямые методы могут быть вычислительно неустойчивыми. При последовательном исключении переменных из большой системы уравнений могут накапливаться значительные погрешности.

Итерационные методы решения СЛАУ позволяют приблизиться к точному решению системы уравнений с некоторой точностью путем выполнения некоторой итерационной процедуры. *Одношаговый итерационный метод* при вычислении нового приближения использует только значение предыдущего приближения. *Канонической формой одношагового итерационного метода* для системы (3.1) является его запись в форме

$$B_{k+1}\frac{x_{k+1} - x_k}{\tau_{k+1}} + Ax_k = f, \tag{3.2}$$

где τ_{k+1} — итерационный параметр, B_{k+1} — невырожденная матрица, задающая тот или иной итерационный метод. Предполагается, что задано начальное приближение x_0, тогда формула (3.2) позволяет последовательно вычислять новые приближения x_{k+1}. Для определения x_{k+1} по известным x_k и f достаточно решить систему уравнений

$$B_{k+1}x_{k+1} = F_{k+1}, \quad F_{k+1} = (B_{k+1} - \tau_{k+1}A)x_k + \tau_{k+1}f.$$

Итерационный метод (3.2) называют *явным* (*неявным*), если матрица $B_k = E$ ($B_k \neq E$), где E — единичная матрица. Как правило, неявные итерационные методы имеет смысл применять лишь в том случае, когда каждую матрицу B_k обратить легче, чем исходную матрицу A. К примеру, легко обратить диагональную матрицу или треугольную. Неявные методы позволяют ускорить сходимость.

Итерационный метод (3.2) называют *стационарным*, если $B_{k+1} = B$ и $\tau_{k+1} = \tau$ не зависят от номера итерации, и *нестационарным* — в противоположном случае. Приведем некоторые примеры итерационных методов.

Метод простой итерации: $$\frac{x_{k+1} - x_k}{\tau} + Ax_k = f - \tag{3.3}$$

явный стационарный метод, существует способ выбора оптимального параметра τ.

Метод Якоби: $$D\,(x_{k+1} - x_k) + Ax_k = f, \tag{3.4}$$

здесь D — диагональ матрицы A. Неявный стационарный метод, поскольку матрица D диагональная, выражения $D^{-1}y$ легко вычисляются.

Метод верхней релаксации: $$(D + \omega L)\,\frac{x_{k+1} - x_k}{\omega} + Ax_k = f, \tag{3.5}$$

здесь D — диагональ матрицы A, матрица L — часть матрицы A ниже диагонали, $\omega > 0$ — скалярный параметр. Неявный стационарный метод, матрица $D + \omega L$ — нижняя треугольная матрица, поэтому выражения $(D + \omega L)^{-1}y$ легко вычисляются. При значении параметра $\omega = 1$ метод переходит в метод Зейделя.

При исследовании различных итерационных методов нас интересуют два основных вопроса: при каких условиях итерационная процедура сходится и какова скорость сходимости. Выпишем несколько теорем о сходимости стационарных итерационных методов.

3.2. Условия сходимости стационарных методов

Теорема 3.1 (необходимое и достаточное условие сходимости стационарных итерационных методов). Стационарный итерационный метод сходится при любом начальном приближении x_0 тогда и только тогда, когда все собственные значения матрицы перехода $S = E - \tau B^{-1}A$ по модулю меньше единицы.

Доказательство. Строгое доказательство теоремы занимает несколько страниц и использует много нетривиальной линейной алгебры [1, с. 90]. Здесь приведем только основную идею доказательства. Пусть x — точное решение системы линейных уравнений (3.1), введем вектор отклонений на k-й итерации: $z_k = x_k - x$. Погрешность на k-й итерации равна $\|z_k\|$. При выполнении итераций векторы x_k и z_k изменяются по следующим формулам:

$$x_{k+1} = Sx_k + \tau B^{-1} f, \qquad z_{k+1} = Sz_k.$$

Тогда оценка погрешности на k-й итерации:

$$\|x_k - x\| \le \left\|S^k\right\| \cdot \|x_0 - x\| \le q^k \|x_0 - x\|.$$

Постоянное уменьшение погрешности $\|x_k - x\|$ со скоростью геометрической прогрессии с показателем q для любых начальных данных x_0 гарантируется только в том случае, когда норма матрицы перехода S меньше единицы, или, что эквивалентно, спектральный радиус матрицы $\rho(S)$ меньше единицы. При этом скорость сходимости метода q пропорциональна $\rho(S)$. Самая быстрая сходимость достигается при $q \ll 1$. Хороший выбор параметров τ и B позволяет уменьшить $\rho(S)$ и ускорить сходимость. □

Данная теорема имеет принципиальное теоретическое значение и накладывает минимальные ограничения на матрицы A и B. Однако на практике ее результаты редко применимы, поскольку оценка спектра матрицы S является очень сложной задачей. При некоторых ограничениях на матрицы A и B удается получить более простые условия сходимости.

Теорема 3.2 (сходимость стационарного итерационного метода). Пусть A — симметричная положительно определенная матрица, параметр $\tau > 0$, и пусть выполнено неравенство

$$B - \frac{1}{2}\tau A > 0;$$

тогда стационарный итерационный метод сходится. Матричное неравенство $M > 0$ понимается в том смысле, что матрица M — положительно определена. Теорема имеет следствия, которые позволяют получить простые условия сходимости для некоторых методов.

Следствие 1. Пусть A — симметричная положительно определенная матрица, а $\lambda_{\max}$ — максимальное собственное значение матрицы. Тогда метод простой итерации (3.3) сходится при выборе параметра $\tau < 2/\lambda_{\max}$.

Следствие 2. Пусть A — симметричная положительно определенная матрица с диагональным преобладанием, т.е. для каждого диагонального элемента $a_{ii} \geq \sum_{j \neq i} |a_{ij}|$, где по крайней мере одно неравенство строгое, тогда метод Якоби (3.4) сходится.

Следствие 3. Пусть A — симметричная положительно определенная матрица, тогда метод верхней релаксации (3.5) сходится при выборе итерационного параметра $0 < \omega < 2$. В частности, метод Зейделя ($\omega = 1$) сходится.

3.3. Скорость сходимости итерационных методов

Теорема 3.3 (о скорости сходимости стационарных методов для симметричных положительно определенных матриц).

Пусть A и B — симметричные положительно определенные матрицы, и пусть справедливы соотношения

$$\gamma_1 B \le A \le \gamma_2 B, \tag{3.6}$$

где γ_1, γ_2 — положительные постоянные, $\gamma_1 < \gamma_2$. При

$$\tau = \frac{2}{\gamma_1 + \gamma_2}$$

стационарный итерационный метод сходится, и для погрешности справедливы оценки:

$$\begin{aligned} \|x_k - x\|_A &\le q^k \|x_0 - x\|_A, \\ \|x_k - x\|_B &\le q^k \|x_0 - x\|_B, \end{aligned} \quad q = \frac{\kappa - 1}{\kappa + 1}, \quad \kappa = \frac{\gamma_2}{\gamma_1} > 1,$$

где $\|v\|_A = \sqrt{(Av, v)}$, $\|v\|_B = \sqrt{(Bv, v)}$.

Самая быстрая сходимость (минимум q) достигается при выборе минимального значения параметра κ, наиболее близкого к единице. Таким образом, для достижения наилучшей сходимости требуется выбрать максимальную из возможных оценку параметра γ_1 и минимальную из возможных оценку параметра γ_2.

Рассмотрим обобщенную задачу на собственные значения

$$Av = \lambda Bv.$$

Если для матриц A и B выполнены неравенства (3.6), то для любого собственного вектора v выполнены неравенства

$$\gamma_1 (Bv, v) \le (Av, v) = \lambda (Bv, v) \le \gamma_2 (Bv, v),$$

откуда следует, что $\gamma_1 \le \lambda_{\min}(B^{-1}A) < \lambda_{\max}(B^{-1}A) \le \gamma_2$, где $\lambda_{\min}(B^{-1}A)$ и $\lambda_{\max}(B^{-1}A)$ — минимальное и максимальное собственные значения для обобщенной задачи на собственные значения. Эти оценки являются самыми строгими для γ_1 и γ_2. При использовании таких оценок мы получаем оптимальный шаг τ_{opt} для стационарных итерационных методов:

$$\tau_{opt} = \frac{2}{\lambda_{\min}(B^{-1}A) + \lambda_{\max}(B^{-1}A)}, \quad q = \frac{\kappa - 1}{\kappa + 1}, \quad \kappa = \frac{\lambda_{\max}(B^{-1}A)}{\lambda_{\min}(B^{-1}A)}. \tag{3.7}$$

Следствие. Если A — симметричная положительно определенная матрица, то метод простой итерации (3.3) имеет максимальную скорость сходимости при выборе оптимального параметра τ_{opt}:

$$\begin{aligned} \tau_{\text{opt}} &= \frac{2}{\lambda_{\min}(A) + \lambda_{\max}(A)}, \\ q &= \frac{\kappa - 1}{\kappa + 1}, \\ \kappa &= \frac{\lambda_{\max}(A)}{\lambda_{\min}(A)}. \end{aligned} \tag{3.8}$$

3.4. Число обусловленности

Пусть условия теоремы о сходимости выполнены, и для погрешности справедлива оценка

$$\|x_k - x\| \le q^k \|x_0 - x\|, \qquad q < 1.$$

В этом случае говорят, что метод сходится со скоростью геометрической прогрессии со знаменателем q. На каждой итерации погрешность уменьшается в q раз. Зная показатель q, можно найти количество итераций n_0, требуемое для достижения заданной точности.

Пусть требуется уменьшить погрешность до значения ε от изначальной, тогда необходимое число итераций:

$$k > n_0(\varepsilon) = \frac{\ln(1/\varepsilon)}{\ln(1/q)} = \log_{1/q} \frac{1}{\varepsilon}.$$

После n_0-й итерации погрешность $\|x_k - x\| < \varepsilon \|x_0 - x\|$. Выражение в знаменателе $\ln(1/q)$ называется *скоростью сходимости итерационного метода*. Скорость сходимости не зависит от начального приближения x_0, от правых частей f и номера итерации. Скорость сходимости определяется используемым итерационным методом и матрицей A.

Наихудшая скорость сходимости наблюдается при значениях $\kappa \gg 1$. Действительно, из формул (3.7) и (3.8) следует, что в этом случае параметр $q \approx 1$, поэтому погрешность будет убывать крайне медленно.

Оценим скорость сходимости при $\kappa \gg 1$:

$$\ln \frac{1}{q} = \ln \frac{\kappa + 1}{\kappa - 1} = \ln\left(1 + \frac{2}{\kappa - 1}\right) \approx \frac{2}{\kappa - 1} \approx \frac{2}{\kappa}.$$

Таким образом, чем больше параметр κ, тем меньше скорость сходимости. А необходимое число итераций $n_0(\varepsilon)$ растет пропорционально числу κ:

$$n_0(\varepsilon) = \frac{\kappa}{2} \ln \frac{1}{\varepsilon}. \tag{3.9}$$

Число $\kappa(A) = \lambda_{\max}(A)/\lambda_{\min}(A)$ называется числом обусловленности матрицы A, оно играет важную роль в численных методах. Если число обусловленности $\kappa \gg 1$, то матрицу называют *плохо обусловленной*. Системы уравнений с плохо обусловленными матрицами часто встречаются в приложениях. Для таких матриц $q \approx 1$, а значит, явные итерационные методы (3.8) сходятся медленно. Этим объясняется выбор неявных итерационных схем.

Матрица B в неявных итерационных методах называется *предобуславливателем*. При хорошем выборе матрицы B (в идеале $B \approx A^{-1}$) уменьшается число обусловленности $\kappa = \lambda_{\max}\left(B^{-1}A\right)/\lambda_{\min}\left(B^{-1}A\right)$, а следовательно, увеличивается скорость сходимости.

Лекция 4. Простейшие явные методы

В лекции рассматриваются простейшие одношаговые явные итерационные методы решения СЛАУ: метод простой итерации/метод установления и метод минимальных невязок. Перед прочтением данного материала желательно ознакомиться с лекцией 3, которая содержит теоретические основы методов.

Для записи абстрактных систем линейных уравнений в данной лекции, как и в лекции 3, используются обозначения $Ax = f$. Аппроксимации уравнения Пуассона записываются в форме $Lu = F$.

4.1. Метод простой итерации

Метод простой итерации — явный стационарный одношаговый метод решения системы линейных уравнений $Ax = f$. Итерационная процедура в каноническом виде:

$$\frac{x_{k+1} - x_k}{\tau} + Ax_k = f.$$

Пусть A — симметричная положительно определенная матрица, тогда метод простой итерации сходится при выполнении условия на параметр $\tau < 2/\lambda_{\max}$, где $\lambda_{\max}$ — спектральный радиус матрицы A. Для метода простой итерации также известен оптимальный шаг τ_{opt}, при котором достигается максимальная скорость сходимости:

$$\tau_{opt} = \frac{2}{\lambda_{\min} + \lambda_{\max}},$$

здесь $\lambda_{\min}$ и $\lambda_{\max}$ — минимальное и максимальное собственные значения матрицы A.

В разделе 1.4 приведены собственные значения для трехточечного оператора второй производной (см. (1.6), (1.8), (1.10)). Пусть L — пятиточечная аппроксимация оператора Лапласа $-\Delta$ на прямоугольной сетке. Оператор L включает граничные условия, которые могут различаться по разным осям. Для оператора L также можно получить точные выражения для собственных значений и собственных векторов. Собственные значения L получаются в виде сумм СЗ для трехточечных операторов по каждой оси координат. Собственные векторы L получаются в виде произведения СВ для трехточечных операторов по каждой оси координат.

К примеру, пусть решается задача Дирихле для уравнения Пуассона в прямоугольной области $l_x \times l_y$. Область разбита на прямоугольные ячейки $n_x \times n_y$, размеры ячеек $h_x \times h_y$. Собственные значения трехточечного оператора вдоль каждой оси определяются формулой (1.6). Количество собственных значений пятиточечного оператора равно $n_x n_y$, пронумеруем собственные

значения парой индексов, тогда

$$\lambda_{ij} = \frac{4}{h_x^2}\sin^2\frac{\pi i h_x}{2l_x} + \frac{4}{h_y^2}\sin^2\frac{\pi j h_y}{2l_y}, \qquad i = \overline{1,\, n_x}, \quad j = \overline{1,\, n_y}.$$

Минимальное и максимальное собственные значения *для задачи Дирихле* в прямоугольнике:

$$\lambda_{\min} = \frac{4}{h_x^2}\sin^2\frac{\pi h_x}{2l_x} + \frac{4}{h_y^2}\sin^2\frac{\pi h_y}{2l_y}, \qquad \lambda_{\max} = \frac{4}{h_x^2} + \frac{4}{h_y^2}.$$

Оценка сверху $\lambda_{\max} \le \frac{4}{h_x^2} + \frac{4}{h_y^2}$ справедлива для произвольных граничных условий. Тогда условие сходимости метода простой итерации:

$$\tau < \frac{h_x^2 h_y^2}{2\left(h_x^2 + h_y^2\right)} \sim \frac{h^2}{4}. \tag{4.1}$$

Пусть u и F — сеточные функции, а L — линейный самосопряженный положительно определенный сеточный оператор. И пусть требуется решить уравнение на сеточную функцию u: $Lu = F$. Для использования метода простой итерации нет необходимости задавать оператор L в матричном виде. Достаточно определить действие оператора на сеточную функцию Lu. Пример действия оператора на сеточную функцию приведен в листинге 2.2.

Метод установления основан на том, что решение стационарных задач часто можно рассматривать как решение динамических задач на бесконечности. К примеру, распределение тепла $u(x,\, y)$ в области под действием стационарных источников тепла описывается уравнением Пуассона $\Delta u = -f(x,\, y)$. Решение данного уравнения можно получить, если устремить к бесконечности решение уравнения теплопроводности:

$$v_t - \Delta v = f(x, y), \qquad u(x,\, y) = \lim_{t\to\infty} v(x,\, y,\, t).$$

Получается, численное решение стационарной задачи можно получить, решая уравнение теплопроводности. Если выполнить достаточно много шагов по времени для уравнения теплопроводности, то можно приблизиться к решению стационарной задачи. Пусть L — сеточная аппроксимация оператора $-\Delta$, F — аппроксимация неоднородности в уравнении, u_k — сеточная функция, где k — временной слой, а τ — шаг интегрирования по времени. Выпишем явную схему для уравнения теплопроводности:

$$\frac{u_{k+1} - u_k}{\tau} + Lu_k = F,$$

это не что иное, как метод простой итерации для уравнения $Lu = F$. При $k \to +\infty$ решение u_k уравнения теплопроводности стремится к решению стационарной задачи. Получается, условие сходимости (4.1) — это условие устойчивости явной схемы для двумерного уравнения теплопроводности. Таким образом, метод установления и метод простой итерации эквивалентны.

4.2. Метод минимальных невязок

Пусть требуется решить систему линейных уравнений $Ax = f$, где A — симметричная положительно определенная матрица. Рассмотрим класс явных нестационарных итерационных методов:

$$\frac{x_{k+1} - x_k}{\tau_{k+1}} + Ax_k = f.$$

Введем обозначение для невязки на k-й итерации $r_k = Ax_k - f$. Метод минимальных невязок заключается в том, чтобы минимизировать невязку на новой итерации при известных значениях x_k и r_k на предыдущей итерации. Выпишем невязку r_{k+1} на новой итерации:

$$r_{k+1} = Ax_{k+1} - f = A(x_k - \tau_{k+1}(Ax_k - f)) - f = r_k - \tau_{k+1}Ar_k.$$

Введем норму векторов через скалярное произведение. Тогда квадрат нормы невязки

$$\|r_{k+1}\|^2 = \|r_k\|^2 - 2\tau_{k+1}(Ar_k, r_k) + \tau_{k+1}^2\|Ar_k\|^2.$$

Это квадратичная функция от переменной τ_{k+1}, которая достигает минимума при значении

$$\tau_{k+1} = \frac{(Ar_k, r_k)}{\|Ar_k\|^2}.$$

Параметр $\tau_{k+1} > 0$, поскольку матрица A — положительно определена. Метод минимальных невязок сходится со скоростью метода простой итерации с оптимальным параметром τ_{opt}.

Пусть u и F — сеточные функции, а L — линейный самосопряженный положительно определенный сеточный оператор. И пусть требуется решить уравнение на сеточную функцию u: $Lu = F$. Для решения уравнения можно применить метод минимальных невязок, при этом нет необходимости задавать оператор L в матричном виде. Достаточно определить действие оператора на сеточную функцию Lu, а также ввести операцию скалярного произведения в пространстве сеточных функций. Пример действия оператора на сеточную функцию приведен в листинге 2.2.

4.3. Лабораторная работа

Найти приближенное решение уравнения Пуассона в прямоугольной области. Для решения системы линейных уравнений использовать метод простой итерации или метод минимальных невязок. Список задач приведен в прил. П.2. Решить задачу из пункта (а), построить рядом точное решение, численное и погрешность. Не следует использовать разреженные матрицы и встроенные решатели СЛАУ.

Необязательные пункты:

- сравнить приближенное решение с аналитическим, построить график погрешности, оценить порядок аппроксимации;
- построить зависимость погрешности от номера итерации;
- построить зависимость невязки от номера итерации;
- оценить параметр q, определить, за сколько итераций погрешность/невязка уменьшается в 10 раз;
- найти число обусловленности κ матрицы оператора, определить, как число обусловленности связано с дискретизацией задачи;
- оценить параметр q, найти число обусловленности κ матрицы, сравнить теоретическую оценку параметра q и полученную на практике;
- для метода простой итерации: решить задачу при различных значениях параметра τ. Убедиться, что метод расходится при $\tau > \tau_{\max}$;
- использовать итерационный решатель `scipy.sparse.spsolve` без создания разреженных матриц, для этого создать обёртку `LinearOperator` с функцией умножения матрицы на вектор;
- ответить на фундаментальный вопрос: сколько итераций требуется для достижения предельной точности? Во сколько раз увеличится требуемое число итераций при увеличении числа ячеек в m раз по каждой оси координат?

Лекция 5. Простейшие неявные методы

В лекции рассматриваются простейшие одношаговые неявные итерационные методы решения СЛАУ: метод Якоби и метод Зейделя. Перед прочтением этой лекции желательно ознакомиться с лекцией 3, которая содержит теоретические основы методов.

Для записи абстрактных систем линейных уравнений в данной лекции, как и в лекции 3, используются обозначения $Ax = f$. Аппроксимации уравнения Пуассона записываются в форме $Lu = F$.

5.1. Метод Якоби

Выпишем систему уравнений $Ax = f$ в развернутом виде:

$$\sum_{j=1}^{n} a_{ij}\, x_j = f_i, \qquad i = \overline{1,\, n}.$$

Пусть диагональные коэффициенты в системе уравнений отличны от нуля ($a_{ii} \neq 0$). В этом случае можно из каждого i-го уравнения системы выразить неизвестную x_i:

$$x_i = \frac{1}{a_{ii}}\left(f_i - \sum_{j=1,\, j\neq i}^{n} a_{ij}\, x_j\right), \qquad i = \overline{1,\, n}.$$

Будем выполнять итерации по такой схеме, т.е. на каждой итерации будем пересчитывать переменную x_i из i-го уравнения системы. Пусть $x_i^{(0)}$ — начальное приближение. Тогда итерации метода Якоби можно записать в следующем виде:

$$x_i^{(k+1)} = \frac{1}{a_{ii}}\left(f_i - \sum_{j=1,\, j\neq i}^{n} a_{ij}\, x_j^{(k)}\right), \qquad i = \overline{1,\, n}. \tag{5.1}$$

Рассмотрим метод Якоби на примере задачи Дирихле для уравнения Пуассона:

$$\begin{cases} \Delta u = -f(x,\, y), & 0 < x < l_x, \quad 0 < y < l_y, \\ u(0,\, y) = \mu_L(y), & u(l_x,\, y) = \mu_R(y), \\ u(x,\, 0) = \mu_B(x), & u\big(x,\, l_y\big) = \mu_T(x). \end{cases}$$

Чтобы окончательно всех запутать, в этот раз выпишем конечно-разностную аппроксимацию, т.е. никаких ячеек. Введем равномерную сетку в прямоугольнике, для простоты полагаем, что прямоугольник можно разбить на в точности квадратные ячейки со стороной $h = l_x/n_x = l_y/n_y$, где n_x и n_y —

число отрезков разбиения по сторонам прямоугольника (количество узлов на единицу больше). Пронумеруем узлы сетки, начиная с нуля:

$$x_{ij} = hi, \quad y_{ij} = hj, \qquad i = \overline{0,\, n_x}, \quad j = \overline{0,\, n_y},$$

значения сеточной функции u_{ij} будем относить к узлам сетки. Значения u_{ij} на границах прямоугольника известны и задаются условиями Дирихле, к примеру, $u_{i,0} = \mu_B(x_{i,0})$. Для внутренних узлов сетки ($0 < i < n_x$, $0 < j < n_y$) получаем уравнения

$$\frac{-u_{i+1,j} + 2u_{ij} - u_{i-1,j}}{h^2} + \frac{-u_{i,j+1} + 2u_{ij} - u_{i,j-1}}{h^2} = f_{ij}.$$

В такой записи каждое уравнение соответствует одному узлу сетки, а каждому узлу соответствует значение функции u_{ij}. Чтобы получить итерационный метод Якоби, необходимо выразить значение u_{ij} в узле через значения в соседних узлах:

$$u_{ij}^{(k+1)} = \frac{1}{4}\left(u_{i+1,j}^{(k)} + u_{i-1,j}^{(k)} + u_{i,j+1}^{(k)} + u_{i,j-1}^{(k)}\right) + \frac{h^2}{4} f_{i,j},$$

т.е. на каждой итерации для узла $(i,\, j)$ усредняются значения в соседних узлах и прибавляется величина пропорциональная f_{ij}.

При решении уравнения Лапласа, т.е. при $f(x,\, y) = 0$, получается итерационная процедура

$$u_{ij}^{(k+1)} = \frac{u_{i+1,j}^{(k)} + u_{i-1,j}^{(k)} + u_{i,j+1}^{(k)} + u_{i,j-1}^{(k)}}{4},$$

которая называется сглаживанием Лапласа. Пусть на границах прямоугольника заданы произвольные (возможно, разрывные) функции, тогда сглаживание Лапласа позволяет получить функцию, которая гладким образом связывает функции на сторонах прямоугольника (рис. 5.1). Такую форму принимает эластичная мембрана, натянутая на некоторую форму. Очень красиво.

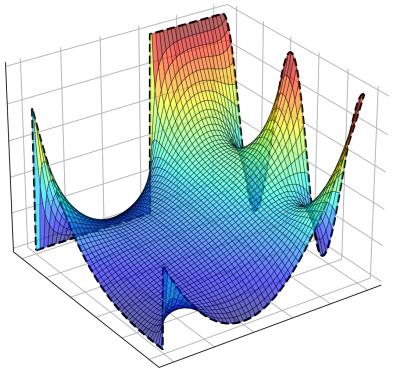

Рис. 5.1. Решение задачи Дирихле для уравнения Лапласа

«Выразить переменную x_i из i-го уравнения» звучит очень хитро, особенно это кажется целесообразно для систем уравнений, в которых коэффициенты на диагонали преобладают. Матрицей с диагональным преобладанием называют матрицу, диагональные элементы которой удовлетворяют условиям

$$\forall i \qquad |a_{ii}| \geq \sum_{j \neq i} |a_{ij}|. \tag{5.2}$$

При этом, если все неравенства строгие, то говорят, что матрица обладает *строгим диагональным преобладанием*, в обратном случае — слабым диагональным преобладанием. Матрицы для сеточного оператора Лапласа, которые мы рассматриваем, имеют ненулевые диагональные элементы и обладают, по крайней мере, слабым диагональным преобладанием. Метод Якоби гарантированно сходится для симметричных матриц и если по крайней мере одно из неравенств (5.2) является строгим. Это достаточное условие сходимости, но не необходимое, метод может сходиться и при более слабых условиях на матрицу. Так, в лабораторной работе будет предложено применить метод Якоби к вырожденным матрицам, которые получаются при дискретизации уравнения Пуассона с периодическими граничными условиями или условиями Неймана.

Пусть необходимо решить уравнение $Lu = F$ на сеточную функцию u, где L — сеточный оператор, F — сеточная функция. Для использования метода Якоби достаточно определить результат действия оператора $\tilde{L} = L - D$ на произвольную сеточную функцию, а также задать матрицу диагональных элементов D. Выпишем действие оператора $\tilde{L}$ для примера 2.1. Для этого достаточно вычеркнуть из формулы (2.8) все слагаемые с u_{ij}:

$$(\tilde{L}u)_{ij} = [i > 1]\frac{-u_{i-1,j}}{h_x^2} + [i < n_x]\frac{-u_{i+1,j}}{h_x^2} + [j > 1]\frac{-u_{i,j-1}}{h_y^2} + \\ + [j < n_y]\frac{-u_{i,j+1}}{h_y^2} + [j = 1]\frac{-u_{i,n_y}}{h_y^2} + [j = n_y]\frac{-u_{i,1}}{h_y^2}.$$

Если u и F — двумерные массивы $n_x \times n_y$, то D также можно представить в виде двумерной матрицы $n_x \times n_y$, которая содержит коэффициенты сеточного оператора L при u_{ij}:

$$D_{ij} = [i > 1]\frac{1}{h_x^2} + [i < n_x]\frac{1}{h_x^2} + [i = 1]\frac{2}{h_x^2} + \frac{2}{h_y^2}.$$

Исходная система уравнений $(Lu)_{ij} = D_{ij}u_{ij} + (\tilde{L}u)_{ij} = F_{ij}$. Выражаем отсюда u_{ij} и получаем метод Якоби:

$$u_{ij}^{(k+1)} = \left(F_{ij} - \left(\tilde{L}u^{(k)}\right)_{ij}\right)/D_{ij}.$$

5.2. Метод Зейделя

Если в итерационной процедуре (5.1) вычислять значения x_i в строгом порядке и при пересчете использовать значения $x_{j<i}$ с нового приближения,

то получится метод Зейделя:

$$x_i^{(k+1)} = \frac{1}{a_{ii}}\left(f_i - \sum_{j=1}^{i-1} a_{ij}\, x_j^{(k+1)} - \sum_{j=i+1}^{n} a_{ij}\, x_j^{(k)} \right).$$

Метод Зейделя имеет более слабые условия сходимости по сравнению с методом Якоби. Метод Зейделя (как и метод Якоби) сходится для матриц со строгим диагональным преобладанием, но он также сходится для симметричных положительно определенных матриц.

Метод Зейделя сходится быстрее метода Якоби, но имеет существенный недостаток: его сложнее распараллелить. Метод Якоби легко реализуется на массово-параллельной архитектуре, в многопоточном режиме и на системах с распределенной памятью. Метод Зейделя в классической версии не поддается параллелизации, поскольку требует последовательного вычисления значений $x_i^{(k+1)}$. По этой же причине метод Зейделя не получится изящно реализовать на срезах в `numpy`.

5.3. Лабораторная работа

Численно решить уравнение Пуассона в прямоугольнике с использованием метода Якоби для решения системы линейных уравнений. Список задач приведен в прил. П.2. Решить задачу из пункта (а), построить рядом точное решение, численное и погрешность. В решении избегать использования циклов, разреженных матриц и матриц с размерами больше числа ячеек. Пример действия оператора на сеточную функцию приведен в листинге 2.2.

Необязательные пункты:

- сравнить приближенное решение с аналитическим, построить график погрешности, оценить порядок аппроксимации;
- построить зависимость погрешности от номера итерации;
- построить зависимость невязки от номера итерации;
- оценить параметр q, определить, за сколько итераций погрешность и невязка уменьшаются в 10 раз;
- найти число обусловленности κ матрицы оператора, определить, как число обусловленности связано с дискретизацией задачи;
- оценить параметр q, найти число обусловленности κ матрицы, сравнить теоретическую оценку параметра q и полученную на практике;
- для метода простой итерации: решить задачу при различных значениях параметра τ. Убедиться, что метод расходится при $\tau > \tau_{\max}$;

- ответить на фундаментальный вопрос: сколько итераций требуется для достижения предельной точности? Во сколько раз увеличится требуемое число итераций при увеличении числа ячеек в m раз по каждой оси координат?

Лекция 6. Разреженные матрицы

6.1. Разреженные и плотные матрицы

Разре́женная/разрежённая матрица — матрица, в которой превалируют нулевые элементы. В противном случае, если бóльшая часть элементов матрицы ненулевая, матрица считается *плотной*.

Условия, при которых матрица считается разреженной, можно ввести по-разному. В рамках курса численных методов разреженной можно считать матрицу $n \times n$, число ненулевых элементов которой $O(n)$. То есть при изменении размера задачи число ненулевых элементов изменяется линейно. Другое близкое определение: считаем матрицу разреженной, если число ненулевых элементов в строке/столбце матрицы меньше некоторой небольшой константы m.

Системы уравнений с разреженными матрицами часто возникают при численном решении уравнений в частных производных. Для метода конечных разностей, для метода конечных объемов, для метода конечных элементов — во всех случаях неявные схемы приводят к возникновению систем уравнений с разреженными матрицами.

Пусть матрица A возникает при написании неявной схемы второго порядка методом конечных объемов. Тогда элементы $A_{kl} \neq 0$, если $k = l$, или если ячейки k и l смежные через грань. В этом случае число ненулевых элементов в k-й строке равно числу соседей k-й ячейки плюс один (элемент на диагонали). Для схем более высокого порядка ячейки могут быть связаны не только с соседними ячейками, но и с ячейками в большей окрестности.

Хорошо знакомый пример разреженной матрицы: трехдиагональная матрица, которая возникает при решении одномерных краевых задач. При размере $n \times n$ трехдиагональная матрица содержит около $3n$ ненулевых элементов. При размере задачи $n = 100$ такая матрица содержит 97 % нулей, при размере задачи $n = 10^5$ такая матрица содержит 99.97 % нулевых элементов. Для хранения и операций с разреженными матрицами целесообразно использовать специальные форматы, которые игнорируют нулевые элементы.

6.2. Форматы хранения разреженных матриц

Существует несколько форматов и соответствующих структур данных для хранения разреженных матриц. Разные форматы отличаются эффективностью выполнения различных операций: добавление нового элемента матрицы, добавление новой строки/столбца, проход по строке/столбцу матрицы, умножение матрицы на вектор, сложение двух матриц и т.д.

DOK — Dictionary of Keys, словарь по ключам. Матрица хранится в виде ассоциативного массива, где ключ — это пара индексов (строка, столбец), а значение — непосредственно значение элемента матрицы. Разреженные

матрицы в DOK-формате предоставляют удобный интерфейс, который позволяет работать с ними как с обычными двумерными массивами. Создание DOK-матрицы 12×12 и инициализация двух элементов на диагонали:

```
import scipy.sparse as sp

A = sp.dok_matrix((12, 12))
A[3, 3] = 10.0
A[4, 4] += 2.0
```

Очевидно, для хранения DOK-матриц используются хэш-таблицы. Хэш-таблицы обеспечивают быстрый доступ к элементам матрицы по индексам, быстрое добавление элементов, в среднем за $O(1)$.

К недостаткам формата относится дороговизна выполнения операций, которые требуют прохода по строкам или столбцам матрицы. Неэффективно выполняются операции сложения двух DOK-матриц, умножения DOK-матрицы на вектор и др.

Матрицы в DOK-формате подходят для создания и инициализации разреженных матриц. После полного заполнения DOK-матрицы ее следует преобразовать в один из сжатых типов хранения, которые более эффективно реализуют матричные операции.

LIL — List of Lists, список списков. Разреженная матрица хранится в виде списка строк, для каждой строки хранится список пар (номер столбца, значение). Каждая строка поддерживается отсортированной по номеру столбца.

Для LIL-матриц относительно быстро выполняется добавление новых элементов, быстро осуществляется доступ к элементу по индексам. Эффективно выполняются срезы по строкам, умножение матрицы на вектор-столбец. Поскольку каждая строка отсортирована по номеру столбца, добавление нового элемента в строку выполняется в худшем случае за $O(m)$, а поиск значения по номеру столбца — за $O(\log m)$, где m — число элементов в строке. Поскольку m — небольшая константа, можно считать, что эти операции выполняются за константное время. LIL-матрицы поддерживают синтаксис срезов и списков индексов. Создание LIL-матрицы 12×12, инициализация трех элементов около диагонали в 5-й (с нуля) строке:

```
import scipy.sparse as sp

A = sp.lil_matrix((12, 12))
A[5, [4, 5, 6]] = [-1.0, 2.0, -1.0]
```

Также допускается обращение к элементам по паре индексов аналогично DOK-матрицам: `A[3, 3] = 10.0`.

Матрицы в LIL-формате подходят для создания и инициализации разреженных матриц. После полного заполнения LIL-матрицы ее следует преобразовать в один из сжатых типов хранения, которые более эффективно реализуют матричные операции.

COO — Coordinate List, список координат. Разреженная матрица хранится в виде трех массивов равной длины: индексы строк r_k, индексы столбцов c_k и

значения v_k. Тройка (r_k, c_k, v_k) описывает ненулевой элемент разреженной матрицы. Допускается хранение троек с одинаковыми парами индексов (r_k, c_k). После приведения к канонической форме значения с одинаковыми индексами складываются, а индексы сортируются сначала по строкам, а затем по столбцам.

Преимущества формата: матрицы в COO-формате удобно использовать при построении, такие матрицы допускают хранение одинаковых элементов, что полезно при реализации различных численных методов, COO-матрицы эффективно преобразуются в сжатые форматы хранения.

Недостатки: формат не поддерживает срезы и арифметические операции. После создания и инициализации матрицу следует преобразовать в сжатый формат данных.

CSR — Compressed Sparse Row. Сжатый формат хранения разреженных матриц по строкам. Фактически — LIL-матрица, у которой зафиксировали структуру, выбросили все ненужное и записали данные в сплошной блок памяти.

Для CSR-матриц эффективно реализованы поэлементные арифметические операции CSR + CSR, CSR * CSR и т.д. Эффективно выполняются срезы по строкам. Быстрое произведение матрицы на вектор-столбец.

Недостатки: дорогая операция среза по столбцам. Дорогостоящее изменение структуры матрицы (по сравнению с LIL или DOK). Очевидно, добавление нового элемента в CSR-матрицу требует целиком пересобрать структуру данных. Библиотека `scipy.sparse` позволяет выполнять операции вида `A[3, 3] = 10.0` для CSR-матриц, но при этом выводит предупреждение.

CSC — Compressed Sparse Column. Формат аналогичен CSR с заменой строк на столбцы. Соответственно, эффективнее выполняются срезы по столбцам, умножение матрицы слева на вектор-строку. Быстро выполняются поэлементные операции CSC + CSC, CSC * CSC и т.д. Добавление элементов является дорогой операцией.

Библиотека `scipy` включает еще несколько дополнительных форматов разреженных матриц. **DIA** — Diagonal Storage, используется для хранения матриц из диагоналей. **BSR** — Block Sparse Row, используется для хранения блочных матриц с плотными блоками.

В целом `scipy` свободно жонглирует форматами для максимальной эффективности. По умолчанию чаще всего используется CSR-формат. В этот формат конвертируется результат большинства арифметических операций.

Рассмотрим следующий пример:

```
import scipy.sparse as sp

rows = [0, 1, 2]
cols = [1, 2, 3]
vals = [0.1, 0.2, 0.3]

A = sp.coo_matrix((vals, (rows, cols)), shape=(5, 5))
B = A + A.T
```

```
C = B + B.T
```

В программе создается COO-матрица A размером 5×5 с тремя ненулевыми элементами. Матрица после транспонирования A.T также имеет COO-формат, при этом результат сложения (матрица B) имеет CSR-формат. Матрица после транспонирования B.T приобретает CSC-формат, а результат сложения (матрица C) снова имеет CSR-формат.

Пример 6.1. Рассмотрим пример 2.1. Требуется решить уравнение Пуассона в квадратной области $[0, l]^2$ с несколькими видами граничных условий — условие Дирихле слева, условие Неймана справа, периодическое замыкание снизу/сверху:

$$\begin{cases} \Delta u = -f(x, y), \quad 0 < x < l, \quad 0 < y < l, \\ u(0, y) = \mu(y), \quad u_x(l, y) = \varphi(y), \\ u(x, y + l) = u(x, y). \end{cases}$$

Разобьем расчетную область на квадратные ячейки со стороной h. Число ячеек по осям равно, $n_x = n_y = l/h$. Выпишем конечно-объемную аппроксимацию уравнения в форме $Lu = F$. Если ячейки и значения сеточных функций нумеруются парой индексов (i, j), то для правых частей F получается выражение (2.9), а действие оператора L на сеточную функцию записывается в виде

$$\begin{aligned} (Lu)_{ij} = [i > 1] \frac{u_{ij} - u_{i-1,j}}{h^2} + [i < n_x] \frac{u_{ij} - u_{i+1,j}}{h^2} + \\ + [j > 1] \frac{u_{ij} - u_{i,j-1}}{h^2} + [j < n_y] \frac{u_{ij} - u_{i,j+1}}{h^2} + \\ + [i = 1] \frac{2u_{ij}}{h^2} + [j = 1] \frac{u_{ij} - u_{i,n_y}}{h^2} + [j = n_y] \frac{u_{ij} - u_{i,1}}{h^2}, \end{aligned}$$

где два последних слагаемых отвечают за периодическое замыкание. Как будет выглядеть матрица оператора L, если ввести нумерацию ячеек единственным индексом?

Каждая строка матрицы соответствует уравнению на одну ячейку, k-я строка матрицы содержит коэффициенты, связывающие k-ю ячейку с ее соседями. В k-й строке матрицы ненулевые элементы содержатся в столбцах с индексами, которые равны индексам соседей k-й ячейки. Также ненулевые элементы располагаются на диагонали матрицы. Пусть k-я ячейка имеет четырех соседей с индексами m_1, m_2, m_3, m_4, тогда k-я строка матрицы будет содержать пять ненулевых элементов в столбцах с индексами m_1, m_2, m_3, m_4 и k.

На диагонали матрицы L располагаются положительные числа. Элемент на диагонали будет равен числу соседей + 2, если у ячейки с одной из сторон задано условие Дирихле, и все поделить на h^2. В данной задаче для ячеек у левого края (условие Дирихле) на диагонали задается число $5h^{-2}$, для

ячеек у правого края (условие Неймана) задается $3h^{-2}$, остальные ячейки имеют ровно по четыре соседа, поэтому соответствующий им диагональный элемент будет равен $4h^{-2}$.

Недиагональный элемент матрицы L_{km} отличен от нуля только в том случае, если ячейки с индексами k и m являются смежными. В данном случае все ненулевые недиагональные элементы матрицы совпадают, $L_{km} = L_{mk} = -h^{-2}$. Для ячеек у левой и правой границы число соседей равно трем, значит, соответствующие строки матрицы содержат по три отрицательных элемента (и положительная диагональ), остальные ячейки имеют по четыре соседа, а значит, соответствующие строки матрицы содержат по четыре отрицательных элемента (и положительная диагональ).

Матрица оператора L симметрична и обладает слабым диагональным преобладанием. Для строк матрицы, которые соответствуют ячейкам у левой границы области (условие Дирихле), есть строгое преобладание диагональных элементов. В целом это все, что можно однозначно сказать о матрице оператора L. Конкретная форма матрицы зависит от введенной нумерации ячеек.

Рассмотрим матрицу L на примере небольшой сетки 6×6 ячеек. Индексация ячеек введена снизу вверх и слева направо, т.е. от левой нижней ячейки сетки. На рис. 6.1 приведена матрица h^2L. Матрица имеет размер 36×36. На осях рисунка подписаны номера строк и столбцов матрицы, цветами показаны значения элементов. Нулевым значениям соответствует белый цвет.

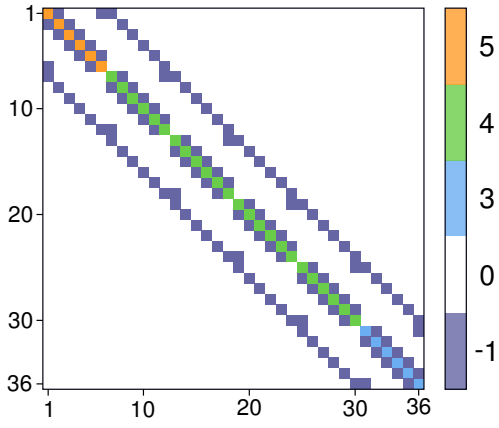

Рис. 6.1. Разреженная матрица

Ячейки с индексами 1–6 располагаются у левого края области, соответствующие диагональные элементы равны 5, в соответствующих строках по три элемента со значениями -1. Ячейки с индексами 31–36 располагаются у правого края области, соответствующие диагональные элементы равны 3, в соответствующих строках по три элемента со значениями -1. Во всех остальных строках на диагонали стоит 4, и имеется по 4 элемента со значениями -1.

Введем нумерацию ячеек двумерной сетки. Пусть координаты x_{ij} и y_{ij}, сеточные функции u_{ij} и F_{ij} хранятся в двумерных `numpy` массивах размера $n_x \times n_y$. Самая «естественная» нумерация ячеек получается при развертывании сеточных функций в одномерные массивы. Это можно сделать с помощью функций `numpy.ravel()` или `numpy.flatten()`. Различие в том, что функция `flatten` создает копию данных, а `ravel` позволяет ссылаться на исходные данные.

Обратной операцией является функция `numpy.reshape((nx, ny))`, которая приводит развернутый одномерный массив в исходную форму дву-

мерного массива. Используем следующий лайфхак для нумерации ячеек в прямоугольнике:

```
k = np.arange(nx * ny, dtype=int).reshape((nx, ny))
```

В результате получим двумерный массив `k` размером $n_x \times n_y$, элемент массива `k[i, j]` равен «естественному» индексу ячейки (i, j). Если $i > 1$, то индекс соседа слева равен `k[i - 1, j]`. Нижняя и верхняя границы периодически замыкаются, поэтому индекс ячейки сверху всегда можно получить как `k[i, (j + 1) % ny]`, соответственно, индекс ячейки снизу — `k[i, (j - 1) % ny]`. Теперь можем пройтись в цикле по ячейкам и заполнить матрицу оператора, код приведен в листинге 6.1.

Листинг 6.1. Инициализация разреженной LIL-матрицы

```
1  import numpy as np
2  import scipy.sparse as sp
3
4  # Cells indexing
5  k = np.arange(nx * ny, dtype=int).reshape((nx, ny))
6
7  # L = sp.dok_matrix((nx * ny, nx * ny))
8  L = sp.lil_matrix((nx * ny, nx * ny))
9
10 for i in range(nx):
11     for j in range(ny):
12         if i == 0:
13             # Has no left neighbor, Dirichlet
14             L[k[i, j], k[i, j]] += 3.0 / hx ** 2
15             L[k[i, j], k[i + 1, j]] = -1.0 / hx ** 2
16         elif i == nx - 1:
17             # Has no right neighbor
18             L[k[i, j], k[i, j]] += 1.0 / hx ** 2
19             L[k[i, j], k[i - 1, j]] = -1.0 / hx ** 2
20         else:
21             # Left and right neighbors
22             L[k[i, j], k[i, j]] += 2.0 / hx ** 2
23             L[k[i, j], k[i - 1, j]] = -1.0 / hx ** 2
24             L[k[i, j], k[i + 1, j]] = -1.0 / hx ** 2
25
26         # Bottom and top neighbors
27         L[k[i, j], k[i, j]] += 2.0 / hy ** 2
28         L[k[i, j], k[i, (j - 1) % ny]] = -1.0 / hy ** 2
29         L[k[i, j], k[i, (j + 1) % ny]] = -1.0 / hy ** 2
30
31 L = L.tocsr()
```

В листинге 6.1 используется LIL-матрица, но точно таким же образом можно инициализировать DOK-матрицу (необходимо раскомментировать строку 7 и вычеркнуть строку 8). Синтаксис доступа к элементам для LIL- и DOK-матриц совпадает. В тесте на производительность заполнение LIL-матриц оказалось эффективней. Возможно, это зависит от числа ненулевых элементов в строке. Заполнение матрицы L для сетки из 6 млн ячеек для LIL-матрицы заняло 141 с против 344 с для DOK-матрицы.

Вероятно, самый быстрый вариант инициализации разреженной матрицы получается через COO-формат. Массивы индексов строк, столбцов и значений можно задать без использования циклов, но для этого придется использовать векторизованные операции со срезами и массивами индексов. Пример приведен в листинге 6.2. Инициализация матрицы таким способом для сетки из 6 млн ячеек занимает 1,7 с, т.е. в 80 раз быстрее, чем при использовании LIL-матриц, и в 200 раз быстрее, чем при использовании DOK-матриц.

Листинг 6.2. Инициализация разреженной CSR-матрицы

```
import numpy as np
import scipy.sparse as sp

# Cells indexing
k = np.arange(nx * ny, dtype=int).reshape((nx, ny))

# Pair of indices (I, J)
I, J = np.meshgrid(np.arange(nx), np.arange(ny))
I, J = I.flatten(), J.flatten()

iC = k[I, J].flatten()  # has bottom and top neighbors
iB = k[I, (J - 1) % ny].flatten()  # bottom neighbors
iT = k[I, (J + 1) % ny].flatten()  # top neighbors

iHR = k[:-1, :].flatten()  # has right neighbors
iR  = k[1: , :].flatten()  # right cells for iHR

iNL = k[0  , :].flatten()  # has no left neighbors
iHL = k[1: , :].flatten()  # has left neighbors
iL  = k[:-1, :].flatten()  # left neighbors for iHL

R1, C1, V1 = iC, iC, np.full(iC.size, +2.0/hy**2)
R2, C2, V2 = iC, iB, np.full(iC.size, -1.0/hy**2)
R3, C3, V3 = iC, iT, np.full(iC.size, -1.0/hy**2)

R4, C4, V4 = iHL, iHL, np.full(iHL.size, +1.0/hx**2)
R5, C5, V5 = iHL, iL,  np.full(iHL.size, -1.0/hx**2)

R6, C6, V6 = iHR, iHR, np.full(iHR.size, +1.0/hx**2)
R7, C7, V7 = iHR, iR,  np.full(iHR.size, -1.0/hx**2)

R8, C8, V8 = iNL, iNL, np.full(iNL.size, +2.0/hx**2)

row = np.concatenate((R1, R2, R3, R4, R5, R6, R7, R8))
col = np.concatenate((C1, C2, C3, C4, C5, C6, C7, C8))
val = np.concatenate((V1, V2, V3, V4, V5, V6, V7, V8))

L = sp.csr_matrix((val, (row, col)))
```

После построения матрицы оператора L можно воспользоваться существующими решателями систем линейных уравнений. Пример приведен в листинге 6.3. Здесь используются два решателя: прямой решатель `spsolve`

и приближенный решатель — стабилизированный метод сопряженных градиентов `bicgstab`. Помимо метода `bcgstab` модуль линейной алгебры содержит и другие итерационные методы, к примеру, уже известный нам метод минимальных невязок `gmres` (generalized minimal residual).

Листинг 6.3. Решение СЛАУ встроенными решателями

```
import scipy.sparse as sp
import scipy.sparse.linalg

L = OpL()                 # csr-matrix
F = RHS(X, Y).flatten()   # right-hand side

# Direct solver
u1 = sp.linalg.spsolve(L, F)
u1 = u1.reshape((nx, ny))

# Iterative solver
u2, b = sp.linalg.bicgstab(L, F)
u2 = u2.reshape((nx, ny))
```

В листинге 6.3 подразумевается, что функция `OpL` возвращает заполненную матрицу в CSR-формате. Также полагаем, что функция `RHS` возвращает неоднородность в виде двумерного массива F_{ij}, поэтому его необходимо развернуть в одномерный. Массивы `u1` и `u2` изначально одномерные, а затем приводятся к двумерным.

6.3. Лабораторная работа

Численно решить уравнение Пуассона в прямоугольнике с использованием существующих решателей систем линейных уравнений. Список задач приведен в прил. П.2. Решить задачу из пункта (а), построить рядом точное решение, численное и погрешность. Решить задачу из пункта (б), тип уравнений и граничных условий в обоих пунктах совпадает, это должно упростить постановку.

Необязательные пункты:

- сравнить приближенное решение с аналитическим, построить график погрешности, оценить порядок аппроксимации;
- протестировать несколько приближенных решателей СЛАУ, сравнить их эффективность и погрешности. Также провести сравнение с прямым решателем;
- оценить погрешность аппроксимации; определить, какую погрешность следует требовать при расчетах с итеративными решателями;
- для расчетов с прямым решателем выяснить, как растет время выполнения задачи и объем используемой памяти при увеличении размеров сетки;
- задать предобуславливатель и протестировать, на сколько улучшится производительность итерационных алгоритмов;
- самый быстрый и изящный метод построения матрицы оператора L основан на произведении Кронекера ($\otimes$) и сумме Кронекера ($\oplus$). Если L_x и L_y — матрицы операторов вдоль направлений x и y, то матрица двумерного оператора L строится в виде суммы Кронекера:

$$A \oplus B = A \otimes E_m + E_n \otimes B,$$

где E_n и E_m — единичные матрицы. Восстановление точной формулы оставим отличникам.

Лекция 7. Уравнения параболического типа

7.1. Двумерное уравнение теплопроводности

Рассмотрим двумерное уравнение теплопроводности в прямоугольнике в довольно общей постановке. На границах области используются граничные условия разных типов: условие Дирихле слева, условие Неймана справа, периодическое замыкание снизу/сверху.

$$\begin{cases} c\rho u_t = \operatorname{div}(\kappa \boldsymbol{\nabla} u) + f(x, y, t), \quad 0 < x < l_x,\ 0 < y < l_y,\ 0 < t, \\ u(0, y, t) = \mu(y, t), \quad \kappa \cdot u_x(l_x, y, t) = \varphi(y, t), \\ u(x, y + l_y, t) = u(x, y, t). \end{cases}$$

В уравнении c — удельная теплоемкость, ρ — плотность, κ — коэффициент теплопроводности. Обратим внимание на наличие коэффициента κ в правом граничном условии, производная указана с коэффициентом теплопроводности κ, поскольку поток тепла пропорционален κ. Введем дифференциальный оператор $\mathcal{L}u = -\operatorname{div}(\kappa \boldsymbol{\nabla} u)$, который включает также граничные условия. Тогда уравнение можно записать в форме

$$c\rho u_t + \mathcal{L}u = f.$$

Выпишем конечно-объемную аппроксимацию для оператора $\mathcal{L}$ и для неоднородности. Подобные выражения уже неоднократно встречались, с такими же граничными условиями, к примеру (2.8) и (2.9). В данной задаче необходимо только верно учесть коэффициент κ.

Сеточный оператор обозначим как L:

$$\begin{aligned} (Lu)_{ij} = \kappa \cdot \Big(& [i > 1] \frac{u_{ij} - u_{i-1,j}}{h_x^2} + [i < n_x] \frac{u_{ij} - u_{i+1,j}}{h_x^2} + \\ & + [j > 1] \frac{u_{ij} - u_{i,j-1}}{h_y^2} + \big[j < n_y\big] \frac{u_{ij} - u_{i,j+1}}{h_y^2} + \\ & + [i = 1] \frac{2u_{ij}}{h_x^2} + [j = 1] \frac{u_{ij} - u_{i,n_y}}{h_y^2} + \big[j = n_y\big] \frac{u_{ij} - u_{i,1}}{h_y^2} \Big). \end{aligned}$$

Сеточная функция F зависит от времени:

$$F_{ij}(t) = f_{ij}(t) + [i = 1] \frac{2\kappa\mu_j(t)}{h_x^2} + [i = n_x] \frac{\varphi_j(t)}{h_x}.$$

В обоих выражениях предполагается, что коэффициент κ является константой, в противном случае необходимо аппроксимировать значение κ на гра-

нях ячеек. Также обратим внимание на наличие коэффициента κ перед $\mu_j(t)$ в неоднородности.

Выпишем теперь схему Кранка – Николсон для уравнения. Временной шаг обозначим как τ. Номер временного шага будем обозначать верхним индексом k, нижние индексы с указанием ячейки опустим:

$$P\frac{u^{k+1}-u^k}{\tau} + (1-\nu)Lu^k + \nu Lu^{k+1} = F\big(t^k + \nu\tau\big), \quad \nu \in [0,\,1].$$

Здесь P – диагональная матрица коэффициентов $c\rho$ в ячейках. Если c и ρ – константы, то P – тоже просто константа, $P = c\rho$. При $\nu = 0$ получается явная схема, условие устойчивости для явной схемы с точностью до констант (4.1). При $\nu = 1$ получается чисто неявная схема. Приведем запись к системе линейных уравнений на u^{k+1}:

$$\big(E + \tau\nu P^{-1}L\big)u^{k+1} = \big(E - \tau(1-\nu)P^{-1}L\big)u^k + \tau P^{-1}F\big(t^k + \nu\tau\big).$$

Если параметры c, ρ и κ не зависят от времени, то матрицы при u^{k+1} и u^k являются постоянными, их достаточно вычислить один раз. Введем для них обозначения A и B:

$$A = E + \tau\nu P^{-1}L, \qquad B = E - \tau(1-\nu)P^{-1}L.$$

После этого система уравнений

$$Au^{k+1} = Bu^k + \tau P^{-1}F\big(t^k + \nu\tau\big)$$

решается в основном цикле программы в одну строку.

Матрица A лучше обусловлена по сравнению с матрицей оператора L, поэтому для решения целесообразно использовать итерационные методы решения СЛАУ. Кроме того, значение u^k на предыдущем временном слое является хорошим начальным приближением для u^{k+1}. Также, если матрица A постоянная, то можно единственный раз в начале программы найти для нее предобуславливатель и использовать его в итерационном методе, тогда расчет будет просто «летать».

7.2. Двумерное уравнение Шрёдингера

Рассмотрим нестационарное уравнение Шрёдингера в прямоугольной области. На границах используются различные виды граничных условий: условие Дирихле слева, условие Неймана справа, периодическое замыкание снизу/сверху. Ограничимся рассмотрением только однородного уравнения с

однородными граничными условиями.

$$\begin{cases} i\hbar\dfrac{\partial\psi}{\partial t} = \mathcal{H}\psi, \quad 0 < x < l_x,\ 0 < y < l_y,\ 0 < t, \\ \psi(0,\ y,\ t) = 0, \quad \psi_x(l_x,\ y,\ t) = 0, \\ \psi(x + l,\ y,\ t) = \psi(x,\ y,\ t). \end{cases}$$

Здесь введен гамильтониан

$$\mathcal{H} = -\frac{\hbar^2}{2m}\Delta + U(x,\ y),$$

где $U(x,\ y)$ — потенциал. Выпишем конечно-объемную аппроксимацию для гамильтониана, сеточный оператор обозначим как H:

$$\begin{aligned}(H\psi)_{ij} = \frac{\hbar^2}{2m}\Bigg([i > 1]\,\frac{\psi_{ij} - \psi_{i-1,j}}{h_x^2} + [i < n_x]\,\frac{\psi_{ij} - \psi_{i+1,j}}{h_x^2} + \\ + [j > 1]\,\frac{\psi_{ij} - \psi_{i,j-1}}{h_y^2} + \big[j < n_y\big]\,\frac{\psi_{ij} - \psi_{i,j+1}}{h_y^2} + [i = 1]\,\frac{2\psi_{ij}}{h_x^2} + \\ + [j = 1]\,\frac{\psi_{ij} - \psi_{i,n_y}}{h_y^2} + \big[j = n_y\big]\,\frac{\psi_{ij} - \psi_{i,1}}{h_y^2}\Bigg) + U_{ij}.\end{aligned}$$

После этого выписываем схему Кранка – Николсон:

$$i\hbar\,\frac{\psi^{k+1} - \psi^k}{\tau} = (1 - \nu)H\psi^k + \nu H\psi^{k+1}, \quad \nu \in [0,\ 1].$$

Схема будет устойчивой при преобладании неявной части $\nu \geq \frac{1}{2}$. В квантовой механике гамильтониан является самосопряженным оператором, при корректной аппроксимации матрица H должна быть эрмитовой. В этом случае при $\nu = \frac{1}{2}$ матрица перехода на новый временной слой по схеме Кранка – Николсон будет унитарной. Таким образом, в численном решении будет сохраняться величина равная суммарной вероятности $\int \psi\bar{\psi}\,dV$, т.е. $(\psi^{k+1},\ \psi^{k+1}) = \big(\psi^k,\ \psi^k\big)$.

Соберем слагаемые при ψ^k и ψ^{k+1}, получим:

$$\left(E + i\frac{\tau\nu}{\hbar}H\right)\psi^{k+1} = \left(E - i\frac{(1 - \nu)\tau}{\hbar}H\right)\psi^k.$$

Матрицы при ψ^{k+1} и ψ^k являются постоянными. Введем для них обозначения A и B. Матрицы достаточно вычислить один раз, после чего система

уравнений решается в основном цикле программы в одну строку:

$$A\psi^{k+1} = B\psi^k, \qquad A = E + i\frac{\tau\nu}{\hbar}H, \quad B = E - i\frac{(1-\nu)\tau}{\hbar}H.$$

Матрица A лучше обусловлена по сравнению с матрицей оператора H, поэтому для решения целесообразно использовать итерационные методы решения СЛАУ. Кроме того, значение ψ^k на предыдущем временном слое является хорошим начальным приближением для ψ^{k+1}. Также, если матрица A постоянная, то можно единственный раз в начале программы найти для нее предобуславливатель, и использовать его в итерационном методе, тогда расчет будет просто «летать».

Красивые формулы для маленьких любителей математики. Схему Кранка – Николсон для уравнения Шрёдингера можно получить другим оригинальным способом. Пусть мы провели пространственную дискретизацию уравнения Шрёдингера, далее считаем значения в ячейках зависящими только от времени $\psi_j(t)$. Тогда получаем систему ОДУ в матричной форме:

$$i\hbar\frac{d\psi}{dt} = H\psi.$$

Решение системы ОДУ можно представить через матричную экспоненту, формула такая же, будто ψ и H скаляры:

$$\psi(t) = e^{-i\frac{t}{\hbar}H}\psi(0),$$

а точная формула продвижения на один временной шаг τ:

$$\psi^{k+1} = e^{-i\frac{\tau}{\hbar}H}\psi^k.$$

Если аппроксимация гамильтониана $\mathcal{H}$ выполнена корректно, тогда матрица H эрмитова. В этом случае матричная экспонента $U = \exp\left(-i\frac{\tau}{\hbar}H\right)$ – это унитарная матрица, а значит, и преобразование волновой функции $\psi^k \to \psi^{k+1}$ унитарное, как и должно быть. Разложим матричную экспоненту в ряд Тейлора по малому параметру и получим различные аппроксимации:

$$\psi^{k+1} = e^{-i\frac{\tau}{\hbar}H}\psi^k \quad \Rightarrow \quad \psi^{k+1} = \left(E - i\frac{\tau}{\hbar}H\right)\psi^k \text{ – явная схема;}$$

$$e^{i\frac{\tau}{\hbar}H}\psi^{k+1} = \psi^k \quad \Rightarrow \quad \left(E + i\frac{\tau}{\hbar}H\right)\psi^{k+1} = \psi^k \text{ – неявная схема.}$$

Для данных двух схем матрица перехода $\psi^k \to \psi^{k+1}$ не является унитарной. Из теории конформных отображений нам известно дробно-рациональное отображение Кэли, которое обобщается на случай произ-

вольных линейных операторов:

$$e^{-iz} = \frac{e^{-i\frac{z}{2}}}{e^{+i\frac{z}{2}}} \approx \frac{1 - i\frac{z}{2}}{1 + i\frac{z}{2}}, \qquad e^{-i\frac{\tau}{\hbar}H} \approx \left(E + \frac{i\tau}{2\hbar}H\right)^{-1}\left(E - \frac{i\tau}{2\hbar}H\right).$$

Отображение Кэли для эрмитовых матриц является унитарным [2, с. 265], как исходная матричная экспонента. Такая аппроксимация матричной экспоненты дает схему Кранка – Николсон:

$$\left(E + \frac{i\,\tau}{2\hbar}H\right)\psi^{k+1} = \left(E - \frac{i\,\tau}{2\hbar}H\right)\psi^{k}.$$

7.3. Лабораторная работа

Провести численное моделирование уравнения параболического типа. Условия задач приведены в прил. П.3. В условиях встречаются три варианта задач: задачи на уравнение теплопроводности, задачи на уравнение диффузии с распадом, задачи на уравнение Шрёдингера. При выполнении работы следует использовать неявные схемы.

Дополнительные вопросы:

1. Если схема абсолютно устойчива, то каким образом следует выбрать временной шаг τ?
2. Сравнить время расчета при использовании прямых и итерационных методов решения СЛАУ.
3. Для задач со стационарным оператором перехода найти предобуславливатель и ускорить расчет.
4. Оценить погрешность аппроксимации уравнения. Какую точность следует требовать при использовании итерационного решателя СЛАУ?

Лекция 8. Итерационный метод с чебышёвским набором параметров

Пафну́тий Льво́вич Чебышёв (1821 — 1894 гг.) — «величайший, наряду с Н.И. Лобачевским, русский математик XIX века» [3, с. 216]. Чебышёв получил фундаментальные результаты в теории чисел (распределение простых чисел) и теории вероятностей (центральная предельная теорема, закон больших чисел), построил общую теорию ортогональных многочленов, теорию равномерных приближений функций и многие другие.

В течение 40 лет, начиная с публикации статьи «Теория механизмов, известных под названием параллелограммов» в 1854 г., ученый занимался теорией наилучшего приближения функций. Теория приближений постепенно охватывала более широкий круг связанных с ней вопросов — теорию ортогональных многочленов, оценки интегралов, квадратурные формулы, интерполирование [3, с. 218], т. е. многие темы, лежащие в основе современных численных методов. Чебышёв разработал метод аппроксимации функций полиномами и рациональными дробями, который без существенных изменений используется в современных компьютерах [4].

Система ортогональных полиномов, носящих имя Чебышёва, появляется в различных разделах численных методов. Новые оригинальные применения многочленов Чебышёва были обнаружены в 1950-х гг. при решении систем линейных уравнений. Об этом красивом математическом методе речь пойдет в следующих параграфах.

Фамилию ученого — по его собственному указанию — следует произносить Чебышо́в. Различные словари [5, 6] и профильные академические издания [7], последовательно использующие букву ё при передаче имен и названий, фиксируют в качестве нормы написание и произношение Чебышёв. Не дадим в обиду букву ё, господа!

8.1. Многочлены Чебышёва

В данном разделе рассматриваются пространства непрерывных на отрезке функций $C[a,b]$. Норма функций в пространстве $C[a,b]$ вводится следующим образом:

$$\|f(x)\| = \max_{x\in[a,b]} |f(x)|,$$

эту норму также называют *нормой Чебышёва* или *равномерной нормой*, поскольку сходимость по этой норме эквивалентна равномерной сходимости. В функциональном анализе также $\|\,.\,\|_\infty$.

Многочлен Чебышёва степени n определяется по формуле

$$T_n(x) = \cos(n \arccos x). \tag{8.1}$$

Многочлены введены Чебышёвым при исследовании полиномов, наименее отклоняющихся от нуля на отрезке $[-1,\ 1]$. Это ключевое свойство полиномов $T_n(x)$ будет доказано ближе к концу параграфа.

Покажем, что тригонометрическая формула (8.1) действительно определяет полиномы. Первые две функции: $T_0(x) = 1$ и $T_1(x) = x$. Для определения следующих выпишем рекуррентную формулу:

$$\begin{aligned}\cos((n+1)\arccos x) + \cos((n-1)\arccos x) = \\ = 2\cos(n\arccos x)\cos(\arccos x) = 2x\cos(n\arccos x) \quad \Rightarrow\end{aligned}$$

$$\Rightarrow T_{n+1}(x) + T_{n-1}(x) = 2xT_n(x).$$

Следовательно, $T_{n+1}(x)$ также является многочленом. Докажем некоторые свойства многочленов Чебышёва.

1. Многочлены Чебышёва четной степени являются четными функциями, нечетной — нечетными:

$$\begin{aligned}T_n(-x) = \cos(n\arccos(-x)) = \cos(\pi n - n\arccos x) = \\ = \cos(\pi n)\cdot\cos(n\arccos x) = (-1)^n T_n(x).\end{aligned}$$

2. Старший коэффициент[1] многочлена Чебышёва степени $n \geq 1$ равен 2^{n-1}. Выпишем рекуррентную формулу:

$$T_{n+1}(x) = 2xT_n(x) - T_{n-1}(x).$$

Из формулы следует, что при увеличении n коэффициент при старшей степени x^n увеличивается в два раза (при $n \geq 1$).

[1] Коэффициент при одночлене с максимальной степенью.

3. Все корни x_k многочленов Чебышёва действительные и располагаются на интервале $(-1,\ 1)$:

$$\cos(n\arccos x) = 0 \quad \Rightarrow \quad n\arccos x = \frac{(2k+1)\pi}{2} \quad \Rightarrow$$
$$\Rightarrow \quad x_k = \cos\frac{(2k+1)\pi}{2n}, \qquad k = \overline{0,\ n-1}.$$

4. Экстремумы $\tilde{x}_k$ многочленов Чебышёва располагаются на отрезке $[-1,\ 1]$, значения в экстремумах равны по модулю и чередуются по знаку:

$$T_n'(x) = \frac{n}{\sqrt{1-x^2}}\sin(n\arccos x) = 0 \qquad \Rightarrow \qquad \tilde{x}_k = \cos\frac{\pi k}{n},$$
$$T_n(\tilde{x}_k) = \cos\left(n\arccos\left(\cos\frac{\pi k}{n}\right)\right) = (-1)^k, \qquad k = \overline{0,\ n}.$$

Следовательно, отклонение многочленов от нуля $\|T_n(x)\| = 1$.

Лемма 8.1 Рассмотрим многочлены степени n с фиксированным старшим коэффициентом равным c_n. Пусть существует упорядоченный набор точек $\tilde{x}_k$ на отрезке $[-1,\ 1]$:

$$-1 \le \tilde{x}_n \le \tilde{x}_{n-1} \le \cdots \le \tilde{x}_1 \le \tilde{x}_0 \le 1,$$

в которых некоторый полином $Q_n(x)$ принимает максимальные по модулю значения, т.е. $|Q_n(\tilde{x}_k)| = \|Q_n\|$, при этом знаки $Q_n(\tilde{x}_k)$ чередуются. Тогда многочлен $Q_n(x)$ наименее отклоняется от нуля на отрезке $[-1,\ 1]$ среди многочленов с коэффициентом при старшей степени равным c_n.

Доказательство. Докажем от противного. Пусть полином $Q_n(x)$ удовлетворяет условиям леммы, т.е. существует упорядоченный набор точек $\tilde{x}_k \in$ $\in [-1,\ 1]$, в которых Q_n достигает экстремумов, и при этом знаки $Q_n(\tilde{x}_k)$ чередуются. Без ограничения общности можно считать, что $Q_n(\tilde{x}_k) = (-1)^k\|Q_n\|$, для $k = \overline{0,\ n}$.

Предположим, что существует отличный от него полином $\tilde{Q}_n(x)$, который имеет такой же старший коэффициент, но меньше отклоняется от нуля, т.е. $\forall x \quad |\tilde{Q}_n(x)| < \|Q_n\|$. Рассмотрим разность $R(x) = Q_n(x) - \tilde{Q}_n(x)$, это полином степени $n-1$. Рассмотрим значения $R(x)$ в узлах $\tilde{x}_k$:

$$R(\tilde{x}_k) = (-1)^k\|Q_n\| - \tilde{Q}_n(\tilde{x}_k), \qquad k = \overline{0,\ n}.$$

При четных k значение $R(\tilde{x}_k) > 0$, при нечетных $-R(\tilde{x}_k) < 0$, т.е. значения функции $R(x)$ в узлах $\tilde{x}_k$ чередуются. Отсюда следует, что на отрезке $[-1, 1]$ полином $R(x)$ n раз меняет знак, т.е. имеет n корней. Приходим к противоречию, поскольку $R(x)$ — это полином степени $n-1$. □

Следствие. Среди многочленов степени $n \geq 1$ со *старшим коэффициентом равным* 2^{n-1} наименее отклоняются от нуля многочлены Чебышёва. В силу доказанных ранее свойств многочлены $T_n(x)$ удовлетворяют всем условиям леммы ⇒ следствие доказано.

Теорема 8.1 (теорема Чебышёва 1854 г.). Для того чтобы некоторый многочлен $P(x)$ степени не выше n был многочленом, наименее уклоняющимся от функции $f \in C[a, b]$, необходимо и достаточно, чтобы на отрезке $[a, b]$ нашлась по крайней мере одна система из $n+2$ точек $x_i : a \leq x_0 \leq x_1 \leq \cdots \leq \leq x_{n+1} \leq b$, в которых разность $f(x) - P(x)$:
1) поочередно принимает значения разных знаков;
2) достигает по модулю наибольшего на $[a, b]$ значения.

Советский математик Яков Евгеньевич Ремез в 1934 г. ввел термин «чебышёвский альтернанс» (от французского alternance — чередование) для обозначения такой системы точек, а также предложил итерационный алгоритм поиска этого самого альтернанса [8]. Алгоритм Ремеза позволяет построить лучшую полиномиальную аппроксимацию для произвольной непрерывной функции. Данная теорема лежит в основе теории аппроксимации [4].

8.2. Многочлены Чебышёва на отрезке $[a, b]$

Многочлены Чебышёва можно ввести на произвольном отрезке. Выполним линейное отображение $t \in [-1, 1]$ на отрезок $x \in [a, b]$:

$$x = \frac{a+b}{2} + \frac{b-a}{2}t, \qquad t = \frac{2x-(b+a)}{b-a}.$$

Многочлены Чебышёва на отрезке $[a, b]$:

$$T_n(x) = \cos\left(n \arccos \frac{2x-(a+b)}{b-a}\right), \qquad x \in [a, b].$$

Корни x_k многочленов также линейно отображаются на отрезок:

$$x_k = \frac{a+b}{2} + \frac{b-a}{2}\cos\frac{(2k+1)\pi}{2n}, \qquad k = \overline{0, n-1}.$$

Отклонение многочленов от нуля не изменяется, $\|T_n\| = 1$. Для полиномов на отрезке также справедлива лемма 8.1. Многочлены Чебышёва наименее отклоняются от нуля на отрезке $[a, b]$ среди полиномов со старшим коэффициентом $\frac{2^{2n-1}}{(b-a)^n}$.

Предположим, что отрезок $[a, b]$ расположен на положительной полуоси, т.е. $0 < a < b$. Рассмотрим систему полиномов $P_n(x)$ с другой нормировкой. Все корни полиномов $T_n(x)$ располагаются на интервале (a, b), значение $T_n(0) \neq 0$, поэтому мы можем ввести полиномы $P_n(x)$ с условием $P_n(0) = 1$:

$$P_n(x) = \frac{T_n(x)}{T_n(0)}.$$

Для многочленов $P_n(x)$ свободный член равен единице. Поскольку многочлены $P_n(x)$ получены из $T_n(x)$ путем умножения на константу, для них выполняются свойства многочленов Чебышёва, связанные с корнями и экстремумами.

Найдем $P_n(x)$ и отклонение $\|P_n\|$. Фактически требуется вычислить значение $T_n(0)$, введем величину $\kappa = b/a$, значение $\kappa > 1$:

$$T_n(0) = \cos\left(n \arccos\left(-\frac{a+b}{b-a}\right)\right) = (-1)^n \cos\left(n \arccos\frac{\kappa+1}{\kappa-1}\right).$$

Величина под $\arccos$ больше единицы. Воспользуемся формулой Эйлера, а также выражением для арккосинуса через натуральный логарифм $\arccos z = -i \ln\left(z + \sqrt{z^2-1}\right)$. Получим формулу

$$\cos(n \arccos z) = \frac{\left(z + \sqrt{z^2-1}\right)^n + \left(z - \sqrt{z^2-1}\right)^n}{2}.$$

Формула справедлива для комплексных z, а также для действительных $z > 1$. Применим формулу для вычисления $T_n(0)$:

$$T_n(0) = \frac{(-1)^n}{2}\left[\left(\frac{\kappa+1}{\kappa-1} + \sqrt{\left(\frac{\kappa+1}{\kappa-1}\right)^2 - 1}\right)^n + \left(\frac{\kappa+1}{\kappa-1} - \sqrt{\left(\frac{\kappa+1}{\kappa-1}\right)^2 - 1}\right)^n\right].$$

Поскольку $\kappa > 1$, при раскрытии выражений можно не задумываться о комплексных числах и ветвлениях:

$$\frac{\kappa+1}{\kappa-1} \pm \sqrt{\left(\frac{\kappa+1}{\kappa-1}\right)^2 - 1} = \frac{\kappa+1}{\kappa-1} \pm \sqrt{\frac{4\kappa}{(\kappa-1)^2}} = \frac{\kappa \pm 2\sqrt{\kappa} + 1}{\kappa-1} = \frac{(\sqrt{\kappa} \pm 1)^2}{(\sqrt{\kappa}+1)(\sqrt{\kappa}-1)}.$$

В результате получаем выражение

$$T_n(0) = \frac{(-1)^n}{2}\left[\left(\frac{\sqrt{\kappa}+1}{\sqrt{\kappa}-1}\right)^n + \left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)^n\right].$$

Введем величину $\rho_1 = \frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}$, где $0 < \rho_1 < 1$, тогда

$$T_n(0) = \frac{(-1)^n}{2}\left(\rho_1^n + \frac{1}{\rho_1^n}\right) = \frac{(-1)^n\,(1+\rho_1^{2n})}{2\rho_1^n}.$$

Итоговое выражение для полиномов P_n:

$$\begin{gathered} P_n(x) = (-1)^n q_n \cos\left(n \arccos\frac{2x-(a+b)}{b-a}\right), \\ q_n = \frac{2\rho_1^n}{1+\rho_1^{2n}}, \qquad \rho_1 = \frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}, \qquad \xi = \frac{b}{a}. \end{gathered} \tag{8.2}$$

Полиномы $P_n(x)$ понадобятся в дальнейшем, в первую очередь нас интересует максимальное отклонение от нуля: $\|P_n\| = q_n$.

Лемма 8.2 Рассмотрим многочлены степени n с фиксированным свободным членом c_0. Пусть существует упорядоченный набор точек $\tilde{x}_k$ на отрезке $[a,\, b]$, где $a > 0$:

$$a \le \tilde{x}_n \le \tilde{x}_{n-1} \le \cdots \le \tilde{x}_1 \le \tilde{x}_0 \le b,$$

в которых некоторый полином $Q_n(x)$ принимает максимальные по модулю значения, т.е. $|Q_n(\tilde{x}_k)| = \|Q_n\|$, при этом знаки $Q_n(\tilde{x}_k)$ чередуются. Тогда многочлен $Q_n(x)$ наименее отклоняется от нуля на отрезке $[a,\, b]$ среди многочленов со свободным членом равным c_0.

Доказательство. Докажем от противного. Пусть полином $Q_n(x)$ удовлетворяет условиям леммы. То есть существует упорядоченный набор точек $\tilde{x}_k \in$ $\in [a,\, b]$, в которых Q_n достигает экстремумов, и при этом знаки $Q_n(\tilde{x}_k)$ чередуются. Без ограничения общности можно считать, что $Q_n(\tilde{x}_k) = (-1)^k \|Q_n\|$, для $k = \overline{0,\, n}$.

Предположим, что существует отличный от него полином $\tilde{Q}_n(x)$, который имеет такой же свободный член, но меньше отклоняется от нуля, т.е. $\forall x \quad |\tilde{Q}_n(x)| < \|Q_n\|$. Рассмотрим разность:

$$R(x) = Q_n(x) - \tilde{Q}_n(x) = xQ_{n-1}(x),$$

здесь удалось вынести переменную x, поскольку свободный член сокращается, и остается некоторый полином $Q_{n-1}(x)$ степени $n-1$. Полином $R(x)$ имеет степень n и как минимум один корень в точке $x = 0$. Рассмотрим значения $R(x)$ в узлах $\tilde{x}_k$:

$$R(\tilde{x}_k) = (-1)^k \|Q_n\| - \tilde{Q}_n(\tilde{x}_k), \qquad k = \overline{0,\, n}.$$

При четных k значение $R(\tilde{x}_k) > 0$, при нечетных $-$ $R(\tilde{x}_k) < 0$, т.е. значения функции $R(x)$ в узлах $\tilde{x}_k$ чередуются. Отсюда следует, что на отрезке $[a,\, b]$ полином $R(x)$ n раз меняет знак, т.е. имеет n корней. Приходим к противоречию, поскольку $R(x)$ имеет на отрезке не более $n-1$ корней. □

Следствие. Среди полиномов со свободным членом равным единице, полиномы $P_n(x)$ наименее отклоняются от нуля на отрезке $[a,\, b]$. Лемма справедлива, поскольку многочлены $P_n(x)$ получены из многочленов Чебышёва путем умножения на константу. Для многочленов $P_n(x)$ выполняются все свойства многочленов Чебышёва, связанные с экстремумами и чередованием знаков на отрезке $[a,\, b]$.

Задача. Пусть полином $f(\lambda)$ задан в форме

$$f(\lambda) = (1 - \tau_1\lambda) \cdot (1 - \tau_2\lambda) \cdot \ldots \cdot (1 - \tau_n\lambda),$$

требуется минимизировать отклонение $f(\lambda)$ на отрезке $\lambda \in [a,\, b]$ путем выбора параметров $\tau_1,\, \tau_2,\, \ldots,\, \tau_n$. Заметим, что $f(0) = 1$, полиномы $P_n(x)$ наименее отклоняются от нуля среди многочленов со свободным членом равным единице. Требуем выполнения $f(\lambda) \equiv P_n(\lambda)$: это возможно, если выбрать $\tau_k = x_k^{-1}$, где x_k $-$ корни многочлена Чебышёва на отрезке $[a,\, b]$. В этом случае отклонение $\|f(\lambda)\|$ минимально и определяется (8.2).

8.3. Итерации с чебышёвским набором параметров

Пусть требуется решить систему линейных уравнений $Ax + f$, где A $-$ симметричная положительно определенная матрица. Рассмотрим явный нестационарный итерационный метод в канонической форме:

$$\frac{x_{k+1} - x_k}{\tau_{k+1}} + Ax_k = f.$$

Каким образом выбрать итерационные параметры $\tau_1,\, \tau_2,\, \ldots,\, \tau_n$, чтобы на n-й итерации погрешность $\|x_n - x\|$ была минимальна?

Теорема 8.2 Пусть $A = A^T > 0$, $\lambda_{\min}, \lambda_{\max}$ — минимальное и максимальное собственные значения A. Среди явных нестационарных итерационных методов решения СЛАУ минимальную погрешность $\|x_n - x\|$ на n-м шаге имеет метод с набором параметров:

$$\kappa = \frac{\lambda_{\max}}{\lambda_{\min}}, \qquad \rho_0 = \frac{\kappa - 1}{\kappa + 1}, \qquad \tau_0 = \frac{2}{\lambda_{\min} + \lambda_{\max}},$$

$$\tau_k = \frac{\tau_0}{1 + \rho_0 t_k}, \qquad t_k = \cos\frac{(2k-1)\pi}{2n}, \qquad k = \overline{1,\, n}.$$

При этом, для погрешности справедлива оценка:

$$\begin{gathered} \|x_n - x\| \le q_n \|x_0 - x\|, \\ q_n = \frac{2\rho_1^n}{1 + \rho_1^{2n}}, \\ \rho_1 = \frac{\sqrt{\kappa} - 1}{\sqrt{\kappa} + 1}. \end{gathered} \tag{8.3}$$

Доказательство. Для погрешности $z_k = x_k - x$ итерационная процедура имеет вид $z_{k+1} = (E - \tau_{k+1}A)z_k$, следовательно,

$$z_n = (E - \tau_n A) \cdot (E - \tau_{n-1}A) \cdot \ldots \cdot (E - \tau_1 A)z_0 = R_n z_0.$$

Матрица R_n симметрична, а значит, $\|R_n\| = |\nu|$, где $|\nu|$ — спектральный радиус R_n, и оценка не улучшаема:

$$\|z_n\| \le \|R_n\| \cdot \|z_0\| = |\nu| \cdot \|z_0\|.$$

Пусть $\lambda_1, \lambda_2, \ldots, \lambda_m$ — собственные значения матрицы A, а v_k — соответствующие собственные векторы. Они же являются СВ для матрицы R_n:

$$\begin{aligned} R_n v_k = (E - \tau_n A) \cdot (E - \tau_{n-1}A) \cdot \ldots \cdot (E - \tau_1 A)v_k = \\ = (E - \tau_n \lambda_k) \cdot (E - \tau_{n-1}\lambda_k) \cdot \ldots \cdot (E - \tau_1 \lambda_k)v_k. \end{aligned}$$

В последней строке приведено СЗ матрицы R_n, которое соответствует v_k. Тогда спектральный радиус матрицы R_n:

$$\begin{aligned} |\nu| = \max_{\lambda_k} \left|(E - \tau_n\lambda_k) \cdot (E - \tau_{n-1}\lambda_k) \cdot \ldots \cdot (E - \tau_1\lambda_k)\right| \le \\ \le \max_{\lambda \in [\lambda_{\min}, \lambda_{\max}]} \left|(E - \tau_n\lambda) \cdot (E - \tau_{n-1}\lambda) \cdot \ldots \cdot (E - \tau_1\lambda)\right|. \end{aligned}$$

Требуется минимизировать величину $|\nu|$. Решение такой задачи нам известно. Для этого в качестве τ_k^{-1} следует выбрать корни многочлена Чебышёва на отрезке $[\lambda_{\min}, \lambda_{\max}]$:

$$\tau_k^{-1} = \frac{\lambda_{\min} + \lambda_{\max}}{2} + \frac{\lambda_{\max} - \lambda_{\min}}{2} \cos \frac{(2k-1)\pi}{2n}, \quad k = \overline{1,\, n}.$$

При этом будет справедлива оценка:

$$|\nu| \le \frac{2\rho_1^n}{1 + \rho_1^{2n}}, \qquad \rho_1 = \frac{\sqrt{\kappa} - 1}{\sqrt{\kappa} + 1}, \qquad \kappa = \frac{\lambda_{\max}}{\lambda_{\min}}.$$

После минимальных преобразований получается выражение из условия теоремы. Что и требовалось доказать. □

Рассмотрим простейший случай $n = 1$. В этом случае мы получаем метод простой итерации с оптимальным параметром:

$$\tau = \frac{2}{\lambda_{\min} + \lambda_{\max}}.$$

То есть метод простой итерации с оптимальным параметром обеспечивает минимальную погрешность на каждом шаге. Показатель геометрической прогрессии, с которым убывает погрешность, равен

$$q = \frac{2\rho_1}{1 + \rho_1^2} = \frac{2(\sqrt{\kappa} - 1)}{(\sqrt{\kappa} + 1)\left(1 + \frac{(\sqrt{\kappa}-1)^2}{(\sqrt{\kappa}+1)^2}\right)} = \frac{2(\sqrt{\kappa} - 1)}{\sqrt{\kappa} + 1 + \frac{(\sqrt{\kappa}-1)^2}{\sqrt{\kappa}+1}} =$$

$$= \frac{2(\kappa - 1)}{(\sqrt{\kappa} + 1)^2 + (\sqrt{\kappa} - 1)^2} = \frac{\kappa - 1}{\kappa + 1},$$

данная формула уже была получена ранее другим способом (см. (3.8)).

8.4. Скорость сходимости

Оценим количество итераций n_0 метода с чебышёвским набором параметров, которое требуется для достижения точности ε. Погрешность метода после n-й итерации $\|x_n - x\| \le q_n \|x_0 - x\|$. Погрешность менее ε от начальной достигается при $q_n < \varepsilon$, отсюда следуют условия:

$$q_n = \frac{2\rho_1^n}{1 + \rho_1^{2n}} < \varepsilon \quad \Rightarrow \quad \frac{1}{\rho_1^n} + \rho_1^n > \frac{2}{\varepsilon} \quad \Rightarrow \quad \frac{1}{\rho_1^n} > \frac{1}{\varepsilon} + \sqrt{\frac{1}{\varepsilon^2} - 1}.$$

Последний переход справедлив при $\frac{1}{\rho_1^n} > 1$. Последнее неравенство заведомо выполняется при следующих условиях:

$$\frac{1}{\rho_1^n} > \frac{2}{\varepsilon} \qquad \Rightarrow \qquad n \ln \frac{1}{\rho_1} > \ln \frac{2}{\varepsilon}.$$

Необходимое число итераций для достижения точности ε:

$$n_0(\varepsilon) = \frac{\ln(2/\varepsilon)}{\ln(1/\rho_1)}.$$

Самый неблагоприятный случай достигается для плохо обусловленных матриц, когда параметр $\kappa \gg 1$ и $\rho_1 \approx 1$. В этом случае

$$\ln \frac{1}{\rho_1} = \ln \frac{\sqrt{\kappa}+1}{\sqrt{\kappa}-1} = \ln\left(1 + \frac{2}{\sqrt{\kappa}-1}\right) \approx \frac{2}{\sqrt{\kappa}-1} \approx \frac{2}{\sqrt{\kappa}}.$$

Оценка числа итераций до заданной точности:

$$n_0(\varepsilon) \approx \frac{\sqrt{\kappa}}{2} \ln \frac{2}{\varepsilon}, \qquad n_0 = O(\sqrt{\kappa}).$$

Ранее для метода простой итерации с оптимальным параметром была получена оценка (3.9):

$$n_0(\varepsilon) \approx \frac{\kappa}{2} \ln \frac{1}{\varepsilon}, \qquad n_0 = O(\kappa).$$

Ускорение метода с чебышёвским набором параметров огромное. Грубо говоря, если метод простой итерации требует $n_0 = 10^6$ итераций для достижения точности ε, то метод с чебышёвскими параметрами достигнет такой точности за $\sqrt{n_0} = 1000$ итераций.

8.5. Порядок чебышёвских параметров

Не смотря на невероятное ускорение сходимости, которое предполагает метод, он не лишен недостатков при реализации на ЭВМ. Оказалось, что на практике метод неустойчив. Наличие машинной погрешности при представлении чисел с плавающей точкой в памяти компьютера чаще всего приводит к накоплению большой погрешности и расходимости метода.

Заметим, что с математической точки зрения порядок использования итерационных параметров $\tau_1, \tau_2, \ldots, \tau_n$ не имеет значения. Независимо от порядка параметров τ_k для финальной погрешности справедлива оценка (8.3). В отсутствие машинной погрешности, при использовании любого порядка параметров τ_k на n-й итерации мы придем к одной погрешности. При

этом на промежуточных итерациях $k < n$ погрешность может быть очень велика, т.е. на промежуточных итерациях x_k никак не приближено к точному решению.

Метод оптимального упорядочивания чебышёвских параметров предложили В.И. Лебедев и С.А. Финогенов [9]. Способ выбора оптимального порядка итерационных параметров исчерпывающе описан у А.А. Самарского [10, с. 572]. Здесь он не приводится, продемонстрируем на примере.

На рис. 8.1 приведены графики погрешностей от итерации при использовании различных итерационных методов решения СЛАУ. Тестируется метод простой итерации с оптимальным параметром и методы с чебышёвскими параметрами. В качестве тестовой задачи выбрано уравнение Пуассона на небольшой двумерной сетке 40×60 ячеек. На рисунке приведены графики для метода простой итерации с оптимальным параметром и метода с чебышёвскими параметрами с различными способами упорядочивания параметров. Начальная погрешность равна единице.

Выполняется 256 итераций каждым методом. Метод с простейшим упорядочиванием параметров Чебышёва по возрастанию оказался неустойчивым. Упорядочивание в случайном порядке оказывается устойчивым, однако при этом в ходе итераций погрешность принимает значения порядка 10^{11} (на грани расходимости). В методе с оптимальным упорядочиванием параметров максимальная погрешность на итерациях порядка 10^3. Два сходящихся метода сходятся к одной погрешности $\varepsilon = 4,2 \cdot 10^{-3}$, как и предполагает теория. Метод простой итерации завершается с погрешностью $\varepsilon = 0,6$.

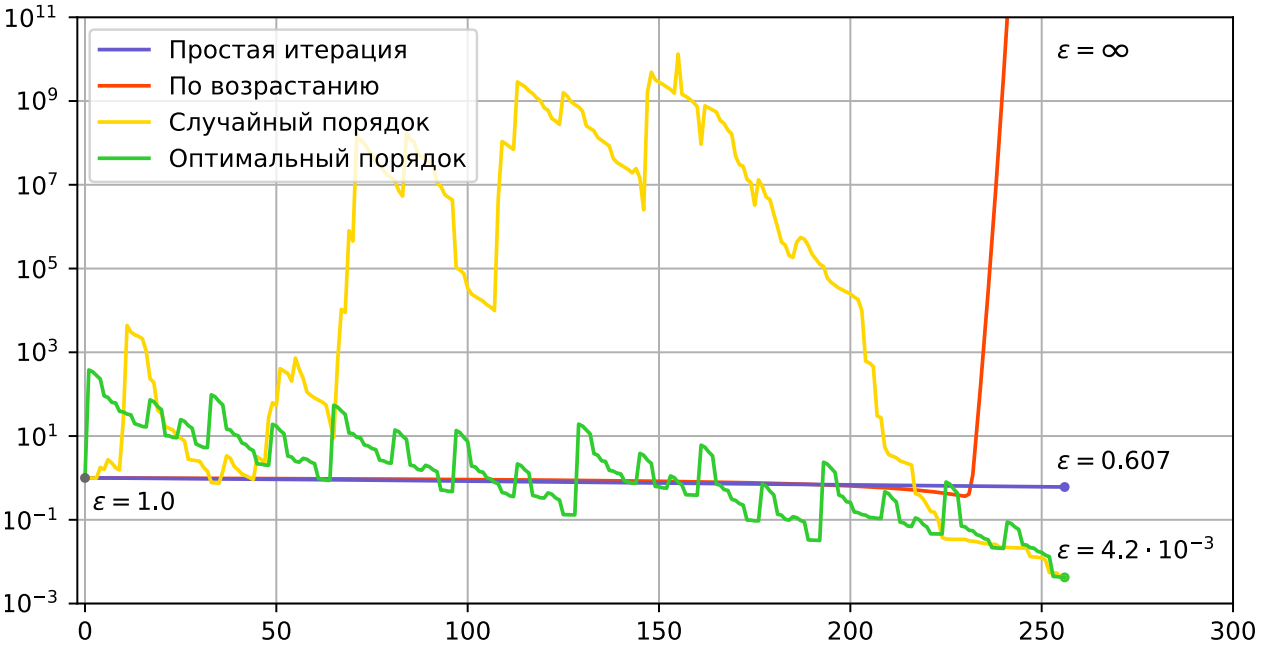

Рис. 8.1. Погрешность итерационных методов при различных способах упорядочивания чебышёвских параметров

Лекция 9. Аппроксимация уравнений на неструктурированных сетках

Триангуляция — разбиение плоского объекта на треугольники. В более высоких размерностях — разбиение на симплексы. *Си́мплекс* — обобщение треугольника и тетраэдра в n-мерном пространстве. Симплекс это простейший многогранник, который строится в n-мерном пространстве на $n+1$ вершине. Триангуляция строится для выпуклой оболочки множества точек, которые становятся вершинами симплексов. Триангуляция для множества точек не является единственной. Даже четырехугольник можно разбить на треугольники двумя способами.

Наиболее распространенный вид триангуляции — *триангуляция Делоне*[1]. Триангуляция Делоне множества точек на плоскости обладает тем свойством, что окружность, описанная вокруг любого треугольника, не включает других точек множества. Если все точки множества лежат на одной прямой, тогда триангуляции Делоне не существует. Если более трёх точек лежат на одной окружности, то триангуляция Делоне не единственна. Простейший пример — квадрат, обе триангуляции квадрата являются триангуляциями Делоне. В остальных случаях триангуляция Делоне существует и единственна.

Аналогичным образом триангуляция Делоне определяется в более высоких размерностях. Вокруг n-мерного симплекса всегда можно описать гиперсферу, при этом требуется, чтобы внутри данной гиперсферы не было других точек множества. Вырождение триангуляции происходит в том случае, если точки множества лежат в одной гиперплоскости n-мерного пространства. При этом триангуляция единственна, если никакие $n+2$ точки не лежат на одной гиперсфере.

В расчетных задачах хотелось бы иметь дело с триангуляцией, которая состоит преимущественно из «хороших» треугольников, близких к правильным. Очевидно, что замостить расчетную область правильными треугольниками удается редко. При триангуляции со сплюснутыми треугольниками страдает точность аппроксимации производных, и как следствие — точность многих численных схем. Триангуляция Делоне *максимизирует минимальный угол* триангуляции, то есть позволяет избежать самых сплюснутых треугольников. Если худший треугольник это «слабое звено» триангуляции, то триангуляция Делоне характеризуется самым крепким слабым звеном среди других триангуляций. На рисунке 9.1 синими линиями показана триангуляция Делоне для некоторого множества точек.

Пусть в области D задано множество из n различных точек $\{\boldsymbol{x}_i\}_{i=1}^n$. *Диаграммой Вороного*[2] данного множества точек называется декомпозиция

[1] Борис Николаевич Делоне́ (1890 – 1980 гг.) — русский и советский математик, профессор МГУ, член-корреспондент Академии наук СССР.

[2] Георгий Феодосьевич Вороной (1868 – 1908 гг.) — русский математик, член-корреспондент Петербургской академии наук.

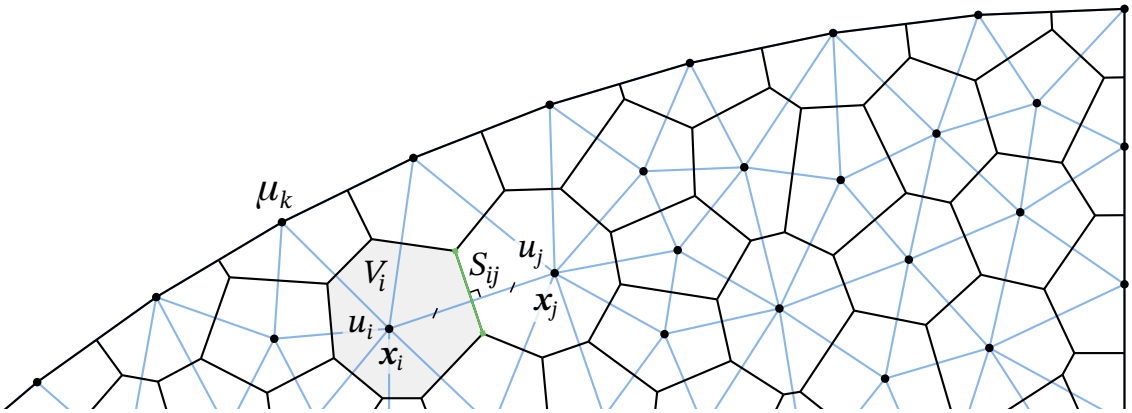

Рис. 9.1. Триангуляция Делоне и соответствующая диаграмма Вороного.

области D на n подобластей $\{\Omega_i\}_{i=1}^n$, каждая из которых содержит точки, которые ближе к $\boldsymbol{x}_i$, чем к остальным:

$$\Omega_i = \left\{ \boldsymbol{x} \in D : \quad \forall j \;\; \|\boldsymbol{x} - \boldsymbol{x}_i\| < \left\|\boldsymbol{x} - \boldsymbol{x}_j\right\| \right\},$$

полученные подобласти Ω_i называются ячейками Вороного, точки $\boldsymbol{x}_i$ — генераторами диаграммы или центрами ячеек Вороного.

Диаграммы Вороного имеют обширную область применений. К примеру, если генераторы диаграммы это вышки телефонной связи, тогда ячейки диаграммы это области покрытия ближайшей вышки. Концепция диаграмм Вороного легко обобщается на произвольные метрические пространства. Интересные диаграммы Вороного получаются на сфере или в манхэттенской метрике[3], при этом последние имеют очевидный смысл для городской навигации. К примеру, карту города можно представить в виде диаграммы Вороного, используя в качестве генераторов станции метро. В этом случае ячейки диаграммы будут показывать районы города, ближайшие к заданной станции метро.

Триангуляции Делоне взаимно однозначно соответствует диаграмма Вороного для того же множества точек. Пример приведен на Рис. 9.1, ячейки Вороного показаны черными линиями, а триангуляция синими. Если соединить центры смежных ячеек Вороного отрезками, то получится триангуляция Делоне. То есть смежность узлов в триангуляции эквивалентна смежности соответствующих ячеек Вороного. Ячейки диаграммы являются выпуклыми многоугольниками. Грань между двумя ячейками диаграммы является серединным перпендикуляром к отрезку, соединяющему центры ячеек. Из этого следует, что вокруг каждой ячейки можно описать окружность с центром в центре ячейки.

[3] Манхэттенская метрика или расстояние городских кварталов — метрика, порождаемая нормой $\|\boldsymbol{x}\| = \sum_i |x_i|$, она же метрика L_1. Так измеряется расстояние между точками в городах с прямоугольной планировкой.

Численные методы для сеток, полученных путем триангуляции, рассмотрим на примере решения задачи Дирихле для уравнения Пуассона в произвольной двумерной области D:

$$\mathcal{P}u = -\frac{1}{\rho(\boldsymbol{x})}\operatorname{div}(\kappa(\boldsymbol{x})\boldsymbol{\nabla}u), \qquad \begin{cases}\mathcal{P}u = f(\boldsymbol{x}), & \boldsymbol{x}\in D,\\ u(\boldsymbol{x}) = \mu(\boldsymbol{x}), & \boldsymbol{x}\in \partial D.\end{cases} \tag{9.1}$$

Заметим, что ключевой задачей здесь является аппроксимация эллиптического оператора $\mathcal{P}$. Аппроксимацию данного оператора затем можно использовать для интегрирования по времени волнового уравнения, уравнения теплопроводности/диффузии/Шрёдингера, или для поиска собственных значений оператора $\mathcal{P}$.

Вспомним некоторые свойства данного оператора, известные из курса математической физики. Оператор является самосопряженным для скалярного произведения с весом $\rho(\boldsymbol{x})$:

$$(\mathcal{P}u,\, v)_\rho = (u,\, \mathcal{P}v)_\rho, \qquad (u,\, v)_\rho = \int_D u(\boldsymbol{x})\, v(\boldsymbol{x})\, \rho(\boldsymbol{x})\, d\boldsymbol{x}.$$

Оператор является положительно определенным, оператор не вырожденный (при наличии условий Дирихле). В силу самосопряженности оператор имеет полный набор собственных функций, собственные функции для разных собственных значений ортогональны с весом $\rho(\boldsymbol{x})$.

9.1. Метод конечных объемов

Рассмотрим подход, описанный в статье Самарского и Вабищевича для уравнений конвекции-диффузии [11]. Пусть построена триангуляция и соответствующая диаграмма Вороного. Обозначим за N количество внутренних узлов сетки, M — число всех узлов. Пронумеруем узлы сетки и соответствующие ячейки. Для внутренних узлов сетки используем индексы $i = \overline{1,\, N}$. Для узлов, лежащих на границе, используем индексы $i > N$.

Значения неизвестной функции $u(\boldsymbol{x})$ задаются в узлах сетки (в центрах ячеек Вороного). Значения сеточной функции в граничных узлах определяются условиями Дирихле:

$$u_i = \mu(\boldsymbol{x}_i), \qquad i > N.$$

Умножим дифференциальное уравнение (9.1) на $\rho(\boldsymbol{x})$ и проинтегрируем его по объему ячейки Вороного Ω_i. Объемный интеграл от дивергенции преобразуется в поверхностный интеграл по границе ячейки:

$$\int_{\Omega_i} f(\boldsymbol{x})\rho(\boldsymbol{x})\, d\boldsymbol{x} = -\oint_{\partial\Omega_i} \kappa\frac{\partial u}{\partial n}\, dS \approx -\sum_{j\in\omega(i)} \kappa_{ij}\frac{u_j - u_i}{|\boldsymbol{x}_j - \boldsymbol{x}_i|}S_{ij}, \quad i = \overline{1,\, N},$$

здесь $\omega(i)$ — множество индексов ячеек, смежных с i-ой ячейкой через грань. Векторы $\boldsymbol{x}_i$ и $\boldsymbol{x}_j$ — центры ячеек, S_{ij} — площадь/длина грани, V_i — объем/площадь ячейки. Выберем коэффициент $\varkappa_{ij} = \varkappa(\boldsymbol{c}_{ij})$, где точка $\boldsymbol{c}_{ij}$ — центр грани. Переменные отмечены на Рис. 9.1.

Введем такой параметр как «вес» ячейки b_i, обозначим как f_i — среднее значение функции $f(\boldsymbol{x})$ в ячейке при интегрировании с весом $\rho(x)$:

$$b_i = \int_{\Omega_i} \rho(\boldsymbol{x})\, d\boldsymbol{x} \approx \rho(\boldsymbol{x}_i)\, V_i, \quad f_i = \frac{1}{b_i} \int_{\Omega_i} f(\boldsymbol{x})\rho(\boldsymbol{x})\, d\boldsymbol{x} \approx f(\boldsymbol{x}_i). \tag{9.2}$$

Приближенные равенства (9.2) выполняются со вторым порядком точности от линейного размера ячейки, если точка $\boldsymbol{x}_i$ является центром масс ячейки с весом $\rho(\boldsymbol{x})$. Если сетка достаточно гладкая, а функция $\rho(\boldsymbol{x})$ непрерывна, тогда точка $\boldsymbol{x}_i$ будет по крайней мере близка к этому центру масс.

Получаем систему линейных уравнений:

$$\xi_{ij} = \frac{\varkappa_{ij} S_{ij}}{|\boldsymbol{x}_i - \boldsymbol{x}_j|}, \qquad \sum_{j \in \omega(i)} \xi_{ij}(u_i - u_j) = b_i f_i, \qquad i = \overline{1,\, N}.$$

Коэффициент ξ_{ij} характеризует «взаимодействие» двух смежных ячеек. Для уравнения теплопроводности эта величина пропорциональна потоку тепла, который перетекает между смежными ячейками. Симметрия $\xi_{ij} = \xi_{ji}$ гарантирует, что количество тепла, которое вытекает из одной ячейки, в точности равно количеству тепла, которое втекает в соседнюю ячейку.

Множество индексов $\omega(i)$ включает как узлы на границе, так и внутренние узлы сетки, разделим их. Обозначим как $\omega_1(i)$ множество соседних узлов, которые являются внутренними, а $\omega_2(i)$ — множество соседних узлов на границе. Множество $\omega_2(i)$ будет пустым, если i-ый узел не связан ни с одним граничным узлом. Для граничных узлов известно точное значение, если $j \in \omega_2(i)$, тогда $u_j = \mu(\boldsymbol{x}_j)$, поэтому

$$\sum_{j \in \omega_1(i)} \xi_{ij}(u_i - u_j) + \sum_{j \in \omega_2(i)} \xi_{ij}(u_i - \mu_j) = b_i f_i, \quad i = \overline{1,\, N}.$$

Перенесем неоднородность в правую часть и получим систему линейных уравнений на внутренние узлы u_i:

$$\Big(\sum_{j \in \omega(i)} \xi_{ij}\Big) \cdot u_i - \sum_{j \in \omega_1(i)} \xi_{ij} \cdot u_j = b_i f_i + \sum_{j \in \omega_2(i)} \xi_{ij} \cdot \mu_j, \quad i = \overline{1,\, N}.$$

Перепишем систему уравнений в матричной форме. Определим диагональную матрицу $B = \operatorname{diag}(b_1,\, b_2,\, \ldots,\, b_N)$. Слагаемые, обусловленные неод-

нородными условиям Дирихле, выделим в вектор g.

$$Lu = Bf + g, \qquad g_i = \sum_{j \in \omega_2(i)} \xi_{ij} \cdot \mu_j, \qquad i = \overline{1, N}, \tag{9.3}$$

$$(Lu)_i = \Big(\sum_{j \in \omega(i)} \xi_{ij} \Big) \cdot u_i - \sum_{j \in \omega_1(i)} \xi_{ij} \cdot u_j, \qquad i = \overline{1, N}. \tag{9.4}$$

Все векторы в системе имеют длину N, а матрицы размерность $N \times N$. Матрица $P = B^{-1}L$ является сеточным аналогом оператора $\mathcal{P}$. Сеточный оператор P повторяет основные свойства эллиптического оператора $\mathcal{P}$. Если ввести скалярное произведение через билинейную форму B: $(u, v)_B = u^T B v$, тогда оператор P является самосопряженным, а собственные векторы ортогональны в пространстве с таким скалярным произведением.

Докажем свойства сеточного оператора P. Матрица L симметрична в силу симметрии коэффициентов ξ_{ij}. Элементы на диагонали матрицы положительны, а недиагональные элементы — отрицательны. Матрица L обладает слабым диагональным преобладанием, при этом есть строгое диагональное преобладание для части строк, которые соответствуют узлам, смежным с граничными узлами. Далее воспользуемся теоремой.

Теорема о кругах Гершгорина[4]. Пусть A — комплексная матрица $N \times N$ с элементами a_{ij}. Вычислим сумму модулей недиагональных элементов в строке $R_i = \sum_{j \neq i} |a_{ij}|$. Построим на комплексной плоскости множество кругов Гершгорина с центрами в токах $a_{ii} \in \mathbb{C}$ и с радиусами R_i. Все собственные числа матрицы A расположены внутри кругов Гершгорина.

Теорема о кругах Гершгорина позволяет локализовать спектр матрицы. Из теоремы, в частности, следует, что симметричная матрица с положительными диагональными элементами со слабым диагональным преобладанием неотрицательно определена, а с сильным диагональным преобладанием — положительно определена. Матрица L относится к промежуточному классу: она обладает строгим диагональным преобладанием только для части строк, которые соответствуют узлам, смежным с граничными.

Матрицы подобные L часто встречаются при аппроксимации эллиптических операторов, они выделены в отдельный класс матриц с *цепным диагональным преобладанием* (weakly chained diagonally dominant matrix, WCDD). Если поставить WCDD матрице в соответствие направленный граф, со связями при $L_{ij} \neq 0$, то от любой вершины со слабым преобладанием в строке в этом графе можно перейти к вершине с сильным преобладанием в строке. Матрица L удовлетворяет этому условию, поскольку граф, который соответствует матрице L, это триангуляция области. Граф триангуляции позволяет дойти по ребрам от любого узла до границы области. Известно, что WCDD матрицы не вырождены [12], а значит матрица L положительно определена.

[4] Семён Аронович Гершгорин (1901 – 1933 гг.) — советский математик, работал в области прикладной математики.

Задача на собственные значения для оператора P эквивалентна обобщенной задаче на собственные значения для пары матриц L и B:

$$Pv = \lambda v \qquad \Leftrightarrow \qquad Lv = \lambda Bv,$$

Диагональная матрица B также является симметричной и положительно определенной, поэтому собственные значения для оператора P являются положительными: $\forall v \quad (Lv,\, v) = \lambda(Bv,\, v) \quad \Rightarrow \quad \lambda > 0$.

При исследовании устойчивости схем нам понадобится оценка для спектрального радиуса оператора P. Оценку сверху также можно получить из теоремы о кругах Гершгорина:

$$\lambda_{\max}(P) \le \max_i \sum_j |P_{ij}| \le \max_i \frac{2}{b_i} \sum_{j \in \omega(i)} \frac{\varkappa_{ij} S_{ij}}{|\boldsymbol{x}_i - \boldsymbol{x}_j|}. \tag{9.5}$$

То есть спектральный радиус матрицы меньше суммы модулей элементов в любой строке. На практике такая оценка оказывается довольно точной.

9.2. Метод конечных элементов

Рассмотрим уравнение Пуассона (9.1) в интегральной форме, называемой также *слабой формой*. Введем следующие функциональные пространства:

- $H^1_\mu(D)$ — множество достаточно гладких функций, принимающих значения $\mu(\boldsymbol{x})$ на границе области ∂D.
- $H^1_0(D)$ — линейное пространство функций, обращающихся в ноль на границе области, то есть удовлетворяющих однородным граничным условиям Дирихле.

Для произвольной тестовой функции $\varphi \in H^1_0(D)$ вычислим скалярное произведение с оператором $\mathcal{P}$, применим формулу Грина:

$$(\mathcal{P}u,\, \varphi)_\rho = -\int_D \varphi \operatorname{div}(\varkappa \boldsymbol{\nabla} u)\, d\boldsymbol{x} = \oint_{\partial D} \varphi \varkappa \frac{\partial u}{\partial n}\, dS + \int_D \varkappa\, (\boldsymbol{\nabla} u,\, \boldsymbol{\nabla}\varphi)\, d\boldsymbol{x}.$$

Поверхностный интеграл равен нулю, поскольку тестовая функция φ обращается в ноль на границе области. Таким образом, для произвольной тестовой функции выполнено интегральное соотношение:

$$\forall \varphi \in H^1_0(D) \qquad \int_D \varkappa\, (\boldsymbol{\nabla} u,\, \boldsymbol{\nabla}\varphi)\, d\boldsymbol{x} = \int_D \varphi(\boldsymbol{x}) f(\boldsymbol{x}) \rho(\boldsymbol{x})\, d\boldsymbol{x}. \tag{9.6}$$

Такая постановка называется слабой формой. Функция $u \in H^1_\mu$ является решением дифференциальной задачи (9.1) тогда и только тогда, когда для любой тестовой функции $\varphi \in H^1_0$ выполнено (9.6).

Основой метода конечных элементов (МКЭ) является метод Галёркина[5], идея которого заключается в следующем:

1. Приближенное решение ищется в конечномерном подпространстве $\Phi^1_\mu(D) \subset H^1_\mu(D)$.
2. Интегральное соотношение (9.6) проверяется на тестовых функциях из конечномерного подпространства $\Phi^1_0(D) \subset H^1_0(D)$.

В качестве таких подпространств выберем множества непрерывных кусочно-линейных функций, линейных на каждом конечном элементе (треугольнике). Такие функции однозначно определяются значениями в узлах триангуляции, а внутри элементов используется линейная интерполяция[6].

Пусть построена триангуляция. Обозначим за N количество внутренних узлов сетки, M — число всех узлов. Пронумеруем узлы сетки. Для внутренних узлов сетки используем индексы $i = \overline{1, N}$. Для узлов, лежащих на границе, используем индексы $i > N$.

Произвольная тестовая функция определяется значениями в N внутренних узлах сетки. Следовательно, Φ^1_0 — N-мерное линейное пространство. Введем базисные функции $\varphi_i(\boldsymbol{x})$ следующим образом: пусть $\varphi_i(\boldsymbol{x}_i) = 1$ и $\varphi_i(\boldsymbol{x}_j) = 0$, $i \neq j$, внутри элементов функция линейна. Базисные функции в методе конечных элементов также называются функциями формы. Аналогичным образом определим функции формы для узлов на границе сетки. В двумерном случае базисные функции $\varphi_i(\boldsymbol{x})$ можно представить в виде пирамидок с единичной высотой.

Любая тестовая функция $v \in \Phi^1_0(D)$ может быть представлена в виде линейной комбинации базисных функций. Для представления функций общего вида потребуется использовать базисные функции граничных узлов:

$$u(\boldsymbol{x}) = \sum_{j=1}^{N} u_j \varphi_j(\boldsymbol{x}) + \sum_{j=N+1}^{M} \mu_j \varphi_j(\boldsymbol{x}), \qquad f(\boldsymbol{x}) = \sum_{j=1}^{M} f_j \varphi_j(\boldsymbol{x}),$$

здесь u_j, μ_j и f_j — значения в узлах сетки.

Выполним подстановку сумм в слабую форму (9.6). В качестве тестовых функций используем функции формы $\varphi_i(\boldsymbol{x})$ при $i < N$. Получаем систему N линейных уравнений на величины u_i:

$$\sum_{j=1}^{N} \left(\int_D \kappa \left(\nabla\varphi_i, \nabla\varphi_j \right) d\boldsymbol{x} \right) \cdot u_j = \sum_{j=1}^{N} \left(\int_D \varphi_i \varphi_j \rho \, d\boldsymbol{x} \right) \cdot f_j -$$

[5] Борис Григорьевич Галёркин (1871 – 1945 гг.) — русский и советский механик и математик, известный своими результатами в области теории упругости.

[6] Верно для любого n-мерного симплекса. Если в вершинах симплекса заданы произвольные значения, то существует единственная линейная функция, которая через них проходит.

$$- \sum_{j=N+1}^{M} \left(\int_D \kappa \left(\nabla \varphi_i, \nabla \varphi_j \right) d\boldsymbol{x} \right) \cdot \mu_j.$$

Введем коэффициенты для всех $i = \overline{1,\, M}$, $j = \overline{1,\, M}$:

$$\xi_{ij} = \int_D \kappa \left(\nabla \varphi_i, \nabla \varphi_j \right) d\boldsymbol{x}, \qquad b_{ij} = \int_D \varphi_i \varphi_j \rho \, d\boldsymbol{x}. \tag{9.7}$$

Коэффициенты ξ_{ij}, b_{ij} отличны от нуля только для смежных пар узлов $(i,\, j)$. В этом случае функции формы $\varphi_i(\boldsymbol{x})$ и $\varphi_j(\boldsymbol{x})$ накладываются друг на друга.

Составим из коэффициентов ξ_{ij} квадратную матрицу L размера $N \times N$. Составим из коэффициентов b_{ij} матрицу B размера $N \times M$. Вынесем в вектор g слагаемые, обусловленные неоднородными граничными условиями. Система уравнений в матричной форме:

$$Lu = Bf + g, \qquad g_i = - \sum_{j=N+1}^{M} \xi_{ij} \mu_j, \qquad i = \overline{1,\, N}. \tag{9.8}$$

Сеточные функции вычисляются в узлах сетки $f_i = f(\boldsymbol{x}_i)$ и $\mu_i = \mu(\boldsymbol{x}_i)$. Матрица L называется матрицей жесткости (*stiffness matrix*), матрица B называется матрицей масс. Такие названия сложились исторически, поскольку исходный метод применялся для решения задач упругости.

Матрица масс B содержит только положительные элементы. Для решения однородных задач следует использовать квадратную матрицу B размера $N \times N$, в этом случае матрица масс симметричная и положительно определенная. Матрица жесткости L также симметричная и положительно определенная. Элементы на диагонали матрицы L положительны, а недиагональные элементы отрицательны. При этом матрица L обладает цепным диагональным преобладанием. Строгое диагональное преобладание есть для строк, которые соответствуют узлам, смежным с граничными. По этой причине матрица жесткости L является невырожденной.

9.3. Интегрирование уравнений по времени

Волновое уравнение

Рассмотрим волновое уравнение с граничными условиями Дирихле. В частности, оно описывает колебания мембраны, закрепленной по краю:

$$\begin{cases} \rho(\boldsymbol{x})\, u_{tt} = \operatorname{div}(\kappa \nabla u), & \boldsymbol{x} \in D, \quad 0 < t, \\ u(\boldsymbol{x},\, t) = 0, & \boldsymbol{x} \in \partial D, \\ u(\boldsymbol{x},\, 0) = \varphi(\boldsymbol{x}), \quad u_t(\boldsymbol{x},\, 0) = \psi(\boldsymbol{x}). \end{cases}$$

Аналог схемы крест для волнового уравнения:

$$B\,\frac{u^{k+1}-2u^k+u^{k-1}}{\tau^2}+Lu^k=0,$$

где τ — шаг интегрирования по времени, u^k — вектор значений во внутренних узлах на k-ом временном шаге. Матрицы для метода конечных объемов (9.2–9.4), для метода конечных элементов (9.7–9.8), следует использовать квадратную матрицу масс B.

Условия устойчивости для трёхслойных схем общего вида изложены у Самарского [1]. При существующих ограничениях на матрицы (матрицы L и B симметричные и положительно определенные), схема будет устойчива при ограничении на шаг по времени:

$$\tau\le\frac{2}{\sqrt{\lambda_{\max}(B^{-1}L)}}.$$

Поскольку для спектра эллиптического оператора справедлива оценка $\lambda_{\max}(B^{-1}L)=O\big(h^{-2}\big)$, для устойчивости потребуется использовать временной шаг $\tau\sim h$.

Уравнение теплопроводности

Рассмотрим однородное уравнение теплопроводности/диффузии:

$$\begin{cases}\rho(\boldsymbol{x})\,u_t=\operatorname{div}(\kappa\boldsymbol{\nabla}u), & \boldsymbol{x}\in D,\quad 0<t,\\ u(\boldsymbol{x},\,t)=0, & \boldsymbol{x}\in\partial D,\\ u(\boldsymbol{x},\,0)=\varphi(\boldsymbol{x}).\end{cases}$$

Выпишем схему Кранка – Николсон:

$$B\,\frac{u^{k+1}-u^k}{\tau}+(1-\nu)Lu^k+\nu Lu^{k+1}=0,\quad \nu\in[0,\,1],$$

в формуле используются те же матрицы для метода конечных объемов (9.2–9.4) или метода конечных элементов (9.7–9.8).

Условия устойчивости для двухслойных схем изложены у Самарского [1]. Преимущественно явная схема ($\nu<\frac{1}{2}$) устойчива при серьезных ограничениях на временной шаг. Для симметричных и положительно определенных матриц L и B получается условие устойчивости:

$$\tau\le\frac{1}{(1/2-\nu)\,\lambda_{\max}(B^{-1}L)}\ \sim\ h^2,\qquad\text{при}\quad\nu<\frac{1}{2}.$$

При преобладании неявной части ($\nu\ge\frac{1}{2}$) схема *абсолютно устойчива*.

Уравнение Шрёдингера

Рассмотрим нестационарное уравнение Шрёдингера с граничным условием Дирихле. Такое граничное условие ставится для частицы в бесконечно глубокой потенциальной яме.

$$\begin{cases} i\psi_t = \left[-\dfrac{1}{2m}\nabla^2 + U(\boldsymbol{x},\, t)\right]\cdot\psi, & \boldsymbol{x}\in D, \quad 0 < t, \\ \psi(\boldsymbol{x},\, t) = 0, & \boldsymbol{x}\in \partial D, \\ \psi(\boldsymbol{x},\, 0) = \psi_0(\boldsymbol{x}). \end{cases}$$

Аппроксимация гамильтониана выполняется аналогично предыдущим пунктам. Для метода конечных объемов используются матрицы (9.2–9.4), для метода конечных элементов матрицы (9.7–9.8). Потенциал $U(\boldsymbol{x},\, t)$ добавляется на диагональ матрицы L. Затем выпишем схему Кранка – Николсон:

$$iB\,\frac{u^{k+1}-u^k}{\tau} + (1-\nu)Lu^k + \nu Lu^{k+1} = 0, \quad \nu\in\left[\tfrac{1}{2},\, 1\right],$$

схема не устойчива при $\nu < \frac{1}{2}$, но абсолютно устойчива при $\nu \geq \frac{1}{2}$.

9.4. Лабораторная работа

Рассмотрим практический пример построения неструктурированной сетки и работы с ней, а также решения уравнения Пуассона и других уравнений. Для выполнения работы предлагается использовать язык python и библиотеки `scipy` и `matplotlib`. Полный код решения не приводится, чтобы дать студентам возможность подумать самостоятельно, но советую следовать общему порядку выполнения и рекомендациям.

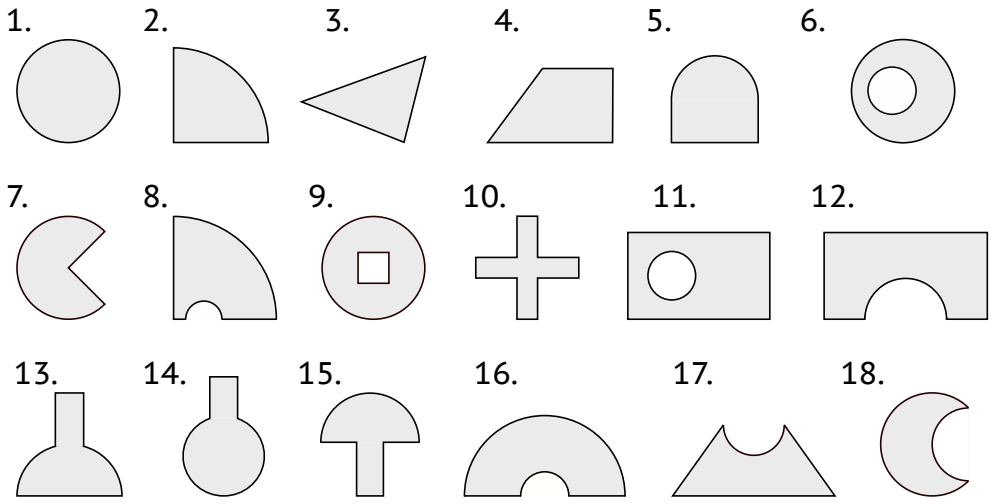

Рис. 9.2. Варианты для выполнения работы.

Задана нестандартная расчетная область (Рис. 9.2). Для примера будем рассматривать область в виде полукруга с отверстием (вариант 16). Границы области — концентрические окружности с радиусами $r_{in} = 0.3$, $r_{out} = 1.0$. Обозначим характерный размер элементов дискретизации как h — желаемая длина стороны треугольников при триангуляции.

Этап 1. Создадим узлы сетки на границе расчетной области. Если границы области это простые параметрические кривые, вроде прямых и окружностей, то не составит труда разбить границу на отрезки с длиной порядка h. В нашем примере следует взять число отрезков на внешней окружности около $\pi r_{out}/h$, а на внутренней около $\pi r_{in}/h$. Число отрезков на прямых участках границы по $(r_{out} - r_{in})/h$.

Координаты узлов на каждом из четырех участков границы можно создать функцией `linspace`. Для объединения нескольких `numpy` массивов удобно использовать функцию `concatenate`, которая последовательно соединяет массивы. Координаты узлов поместим в массивы `xs_b`, `ys_b` (boundary):

```
xs_b = np.concatenate((xs1, xs2, xs3, xs4))
ys_b = np.concatenate((ys1, ys2, ys3, ys4))
```

Следует убедиться, что узлы в массивах не дублируются. Граничные узлы показаны на Рис. 9.3а.

Этап 2. Создадим внутренние узлы сетки. Сначала определим ограничивающий прямоугольник (bounding box) для расчетной области. В рассматриваемом примере это $[-r_{out}, r_{out}] \times [0, r_{out}]$. Равномерно заполним точками ограничивающий прямоугольник, а затем исключим точки, которые не попадают в расчетную область.

Какое количество точек следует создать в ограничивающем прямоугольнике? При замощении области правильными треугольниками число узлов в два раза меньше числа треугольников[7]. Таким образом, чтобы замостить ограничивающий прямоугольник площади S_{bb} правильными треугольниками со стороной h, понадобится порядка N_{bb} узлов:

$$N_{bb} = \frac{S_{bb}}{2S_{tr}}, \qquad S_{tr} = \frac{h^2\sqrt{3}}{4},$$

здесь S_{tr} — площадь правильного треугольника со стороной h. Остается нерешенным вопрос, каким образом сгенерировать узлы в прямоугольнике.

Вариант 1. Использовать генераторы случайных чисел с равномерным распределением. Теоретически, если выбрать равномерное распределение по x и y, то точки равномерно заполнят прямоугольник. Но это справедливо только при стремлении числа точек к бесконечности. Для конечного числа точек будут заметны места сгущения и разрежения точек (Рис. 9.3б).

[7] Следует рассмотреть периодический кусочек замощения, в нем на два треугольника приходится один уникальный узел.

Рис. 9.3. Генерация узлов сетки в области и на границе.

Вариант 2. Использовать квазислучайные последовательности с низким расхождением. Такие последовательности находят множество применений: в методе Монте – Карло, в компьютерной графике, для численного интегрирования, для построения сеток [13]. Первой предложенной последовательностью такого рода является последовательность Соболя[8]. Помимо последовательности Соболя есть также квазислучайные последовательности Кронекера, Хальтона, Нидеррайтера и другие. Последовательность R_d Мартина Робертса на основе золотого сечения имеет хорошие свойства и пишется в несколько строк [13]. Последовательность Соболя реализована в новых версиях библиотеки `scipy`:

```
from scipy.stats.qmc import Sobol
```

На Рис. 9.3в показаны точки, созданные с использованием последовательности Соболя. Они заполняют область более равномерно, по сравнению со случайными точками.

Вариант 3. Использование плотных упаковок. В двумерном случае для генерации точек можно использовать замощение области правильными треугольниками. Однако такой вариант не обобщается на трёхмерный случай, поскольку правильные тетраэдры не замощают пространство. В трёхмерном случае можно использовать узлы плотных упаковок: гранецентрированной кубической (ГЦК) или гексагональной плотной упаковки (ГПУ). Триангуляция на основе упаковок шаров исследуется, к примеру, в работе [14].

После генерации точек в ограничивающем прямоугольнике оставим только те точки, которые находятся внутри расчетной области и поместим их в массивы `xs_i`, `ys_i` (inner). В рассматриваемом примере:

```
r = np.sqrt(x ** 2 + y ** 2)
inside = np.logical_and(r_in < r, r < r_out)
xs_i, ys_i = np.array(x[inside]), np.array(y[inside])
```

Затем совместим массивы внутренних и граничных узлов, граничные узлы расположим после внутренних. Запомним количество внутренних узлов N.

```
n_inner = xs_i.size
xs = np.concatenate((xs_i, xs_b))
ys = np.concatenate((ys_i, ys_b))
```

[8] Илья Меерович Соболь (1926 г.) — советский и российский математик. Ввёл в практику метода Монте – Карло использование квазислучайных последовательностей. Разработал класс последовательностей многомерных точек, носящих его имя.

Этап 3. Триангуляция полученного множества точек. Удобные функции для построения и визуализации триангуляции есть в библиотеке `matplotlib` (менее удобные в `scipy.spatial`).

Листинг 9.1. Триангуляция и визуализация.

```
1  from matplotlib import tri
2
3  # Create triangulation
4  mesh = tri.Triangulation(xs, ys)
5
6  # Mask off unwanted triangles
7  mesh.set_mask(np.hypot(
8      xs[mesh.triangles].mean(axis=1),
9      ys[mesh.triangles].mean(axis=1)) < r_in)
10
11 # Create axes
12 ax = plt.gca()
13 ax.set_aspect('equal')
14
15 # Plot some function
16 u = foo(xs, ys)
17 tcf = ax.tricontourf(mesh, u, levels=30, cmap='jet')
18 plt.colorbar(tcf, ax=ax)
19
20 # Plot triangulation
21 ax.triplot(mesh, color='black')
```

Триангуляция строится для выпуклой оболочки множества узлов. Полученная сетка сохраняется в переменную `mesh`. Поскольку в рассматриваемом примере расчетная область не является выпуклой, то внутренняя окружность также заполняется треугольниками. Лишние треугольники исключаются из рассмотрения. В строках 7-9 в Листинге 9.1 скрываются треугольники, центр которых расположен внутри окружности с радиусом r_{in}. В строке 17 строится цветовая карта для функции u, которая задана в узлах триангуляции. В строке 21 строится сама сетка. Примеры сеток приведены на Рис. 9.4.

а) Случайные узлы б) Посл-сть Соболя в) После сглаживания

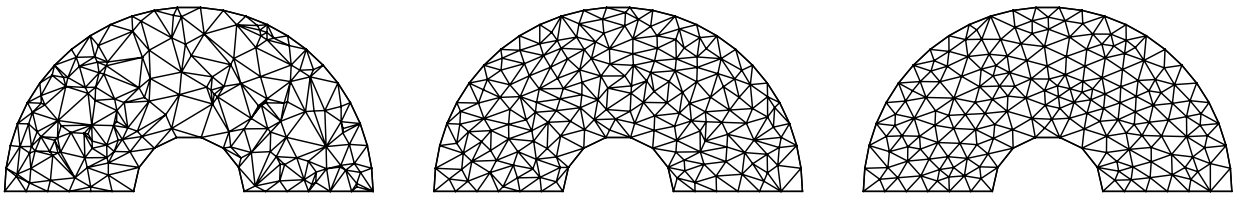

Рис. 9.4. Триангуляция области.

Этап 4. Сглаживание сетки. Если нас не устраивает качество сетки на Рис. 9.4б, то можно провести дополнительное сглаживание. На практике часто применяется сглаживание Ллойда [15]. На каждой итерации сглаживания Ллойда строится диаграмма Вороного, а затем центр каждой ячейки Вороного смещается к барицентру ячейки Вороного. В результате итераций

генераторы ячеек Вороного становятся также центрами масс ячеек. Алгоритм Ллойда является довольно трудоемким, поскольку требует построения диаграммы Вороного на каждой итерации. Другим простым и эффективным методом улучшения сетки (не только из треугольников) является сглаживание Лапласа. В ходе итераций сглаживания каждый узел перемещается на среднее арифметическое смежных с ним узлов.

В Листинге 9.2 приведена простая реализация сглаживания Лапласа. Итерацию сглаживания Лапласа можно представить как действие линейного оператора S на вектор координат:

$$x^{k+1} = Sx^k, \qquad y^{k+1} = Sy^k, \qquad (Su)_i = \frac{1}{|\omega(i)|} \sum_{j \in \omega(i)} u_j, \quad i = \overline{1, N},$$

здесь $\omega(i)$ — множество узлов, смежных с i-ым, $|\omega(i)|$ — количество узлов, смежных с i-ым. Каждая строка матрицы содержит одинаковые элементы, сумма которых равна единице. При сглаживании только внутренние узлы сетки изменяют своё положение, поэтому используется матрица S размера $N \times M$, где N — количество внутренних узлов и M — полное число узлов.

Листинг 9.2. Сглаживание Лапласа для триангуляции.

```
import numpy as np
import scipy.sparse as sp
from matplotlib import tri

def smoothing(xs, ys, n_inner):
    n_points = xs.size

    mesh = tri.Triangulation(xs, ys)

    idx_1  = mesh.edges[:, 0] < n_inner
    rows_1 = mesh.edges[idx_1, 0]
    cols_1 = mesh.edges[idx_1, 1]

    idx_2  = mesh.edges[:, 1] < n_inner
    rows_2 = mesh.edges[idx_2, 1]
    cols_2 = mesh.edges[idx_2, 0]

    rows = np.concatenate((rows_1, rows_2))
    cols = np.concatenate((cols_1, cols_2))
    data = np.ones(rows.size)

    S = sp.csr_array((data, (rows, cols)),
                     shape=(n_inner, n_points))
    S = sp.diags([1.0 / M.sum(axis=1)], [0]) * M

    for k in range(15):
        xs[:n_inner] = S @ xs
        ys[:n_inner] = S @ ys

    return xs, ys
```

Массив `mesh.edges` хранит пары индексов смежных вершин. При этом пары хранятся в единственном экземпляре: если вершины i и j смежные, то массив содержит только одну пару $[i, j]$ или $[j, i]$. Поэтому в Листинге 9.2 используются пары массивов `rows_1`, `rows_2` и `cols_1`, `cols_2`. Объединенный массив `rows` содержит индексы внутренних узлов, массив `cols` — индексы смежных узлов, в том числе узлов на границе области. Массив `data` состоит только из единиц. В строке 23 создается матрица смежности размера $N \times M$, затем каждая строка делится на сумму элементов в строке, то есть на величину $|\omega(i)|$.

Триангуляция Делоне и матрица смежности могут измениться в ходе итераций сглаживания. В Листинге 9.2 сглаживание проводится 15 раз, но для ускорения счета триангуляция не пересчитывается, а матрица S не меняется. После выполнения сглаживания триангуляцию необходимо провести заново. Результат сглаживания Лапласа показан на Рис. 9.4в.

Этап 5а. Для метода конечных объемов. Требуется построить разреженные матрицы L и B размера $N \times N$, векторы f и g размера N. Всю необходимую геометрию можно получить из диаграммы Вороного.

Листинг 9.3. Построение и визуализация диаграммы Вороного.

```
import matplotlib.pyplot as plt
from scipy.spatial import Voronoi, voronoi_plot_2d

vor = Voronoi(np.column_stack((xs, ys)))

ax = plt.gca()
voronoi_plot_2d(vor, ax)
plt.show()
```

Класс `Voronoi` содержит следующие важные атрибуты:

`points`	Массив центров ячеек Вороного, на Рис. 9.5 отмечены кругами.
`vertices`	Массив точек, которые образуют границы ячеек Вороного, на Рис. 9.5 отмечены треугольниками.
`ridge_points`	Массив пар индексов смежных ячеек (смежных вершин триангуляции). На Рис. 9.5 показаны пунктирными линиями.
`ridge_vertices`	Массив пар индексов точек, которые составляют границы ячеек Вороного, то есть связи между `vertices`. Показаны черным цветом на Рис. 9.5. Содержат индекс -1 для бесконечных границ.

В массиве `ridge_vertices` перечислены уникальные пары индексов, а массив `ridge_points` содержит удвоенные пары индексов. Таким образом, размер массива `ridge_points` в два раза больше, при этом первая

половина элементов соответствует элементам в массиве `ridge_points`. Рассмотрим Рис. 9.5, пусть `ridge_points[r]` содержит индексы $(i,\ j)$, которые указывают на точки $(\boldsymbol{x}_i,\ \boldsymbol{x}_j)$ — центры смежных ячеек Вороного, тогда `ridge_vertices[r]` содержит пару индексов $(l,\ k)$, которые указывают на точки $(\boldsymbol{v}_l,\ \boldsymbol{v}_k)$ — точки общей грани между соответствующим ячейками Вороного.

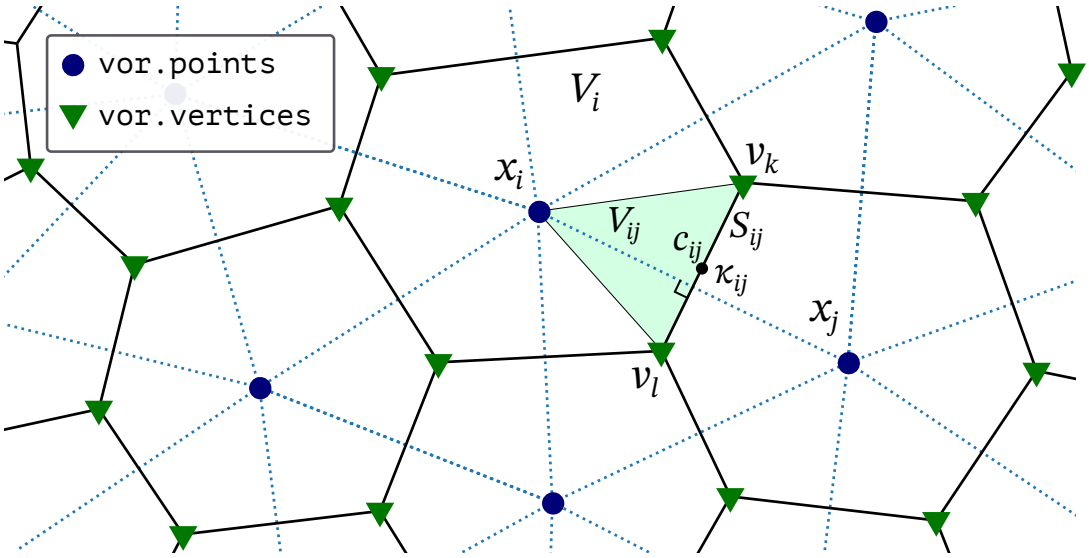

Рис. 9.5. Сетка из ячеек Вороного.

Данные атрибуты класса достаточны для заполнения необходимых матриц и векторов. Центр грани находится как полусумма $\boldsymbol{c}_{ij} = \frac{1}{2}(\boldsymbol{v}_l + \boldsymbol{v}_k)$. Коэффициент $\varkappa_{ij}$ вычисляется в центре грани. Длина грани $S_{ij} = |\boldsymbol{v}_l - \boldsymbol{v}_k|$. Расстояние между центрами ячеек $d_{ij} = |\boldsymbol{x}_i - \boldsymbol{x}_j|$. Площадь ячейки Вороного составляется из площадей треугольников $V_i = \sum_j V_{ij}$, где $V_{ij} = \frac{1}{4} S_{ij} \cdot d_{ij}$.

Параметры $\varkappa_{ij}$, S_{ij}, d_{ij}, V_{ij} удобно хранить в разреженной матрице размера $N \times M$, где N — число внутренних узлов, M — число всех узлов. После этого заполняются разреженная матрица L, диагональная матрица B, векторы неоднородностей f и g.

Этап 5б. Для метода конечных элементов. Треугольный конечный элемент Δ_{ijk} определяется индексами трёх вершин i, j, k. Вычислим барицентр $\boldsymbol{x}_c$ и площадь S_{ijk} треугольника Δ_{ijk}:

$$\boldsymbol{x}_c = \frac{1}{3}(\boldsymbol{x}_i + \boldsymbol{x}_j + \boldsymbol{x}_k), \quad S_{ijk} = \frac{1}{2}\Big|(x_i - x_k)(y_j - y_k) - (x_j - x_k)(y_i - y_k)\Big|.$$

В листинге 9.4 приведен пример вычисления центров и площадей конечных элементов. В коде `mesh` — триангуляция, `x`, `y` — массивы координат узлов, `area` — массив площадей треугольников.

Листинг 9.4. Вычисление параметров конечных элементов.

```
# Get actual triangles
elems = mesh.get_masked_triangles()

# Array of indices of triangles' nodes
i, j, k = elems[:, 0], elems[:, 1], elems[:, 2]

# Centers of the triangles
xc = (x[i] + x[j] + x[k]) / 3.0
yc = (y[i] + y[j] + y[k]) / 3.0

# Areas of the triangles
area = 0.5 * np.abs((x[i] - x[k]) * (y[j] - y[k]) -
                    (x[j] - x[k]) * (y[i] - y[k]))
```

Далее вычислим коэффициенты ξ_{ij} и b_{ij} для всех $i = \overline{1, M}$, $j = \overline{1, M}$, коэффициенты запишем в разреженные матрицы. Интегрирование по каждому конечному элементу вносит вклад в коэффициенты матриц ξ_{ij} и b_{ij}. Для формирования матрицы масс используются следующие интегралы по треугольнику Δ_{ijk}:

$$\int_{\Delta_{ijk}} \varphi_i^2 \rho \, d\boldsymbol{x} \approx \frac{1}{6}\rho(\boldsymbol{x}_c) S_{ijk}, \qquad \int_{\Delta_{ijk}} \varphi_i \varphi_j \rho \, d\boldsymbol{x} \approx \frac{1}{12}\rho(\boldsymbol{x}_c) S_{ijk}.$$

Для одного треугольника Δ_{ijk} можно вычислить 9 таких интегралов, которые добавляются к коэффициентам матрицы b_{ii}, b_{jj}, b_{kk}, b_{ij}, b_{ji}, b_{ik}, b_{ki}, b_{jk}, b_{kj}. В силу симметрии достаточно вычислить только 6 интегралов. Аналогичным образом вычисляются 9 интегралов от градиентов базисных функций

$$\int_{\Delta_{ijk}} \kappa \, (\boldsymbol{\nabla}\varphi_i)^2 d\boldsymbol{x} \approx \frac{\kappa(\boldsymbol{x}_c)}{4S_{ijk}}\big[(x_j - x_k)^2 + (y_j - y_k)^2)\big].$$

$$\int_{\Delta_{ijk}} \kappa \, (\boldsymbol{\nabla}\varphi_i, \boldsymbol{\nabla}\varphi_j) d\boldsymbol{x} \approx -\frac{\kappa(\boldsymbol{x}_c)}{4S_{ijk}}\big[(x_i - x_k)(x_j - x_k) + (y_i - y_k)(y_j - y_k)\big].$$

Значения интегралов добавляются к коэффициентам ξ_{ii}, ξ_{jj}, ξ_{kk}, ξ_{ij}, ξ_{ji}, ξ_{ik}, ξ_{ki}, ξ_{jk}, ξ_{kj}. Точные равенства в формулах выше достигаются в том случае,

если $\varkappa(\boldsymbol{x}) = \text{const}$ и $\rho(\boldsymbol{x}) = \text{const}$. В остальных случаях используется аппроксимация: значения $\varkappa$ и ρ выбираются в барицентрах треугольников.

Листинг 9.5. Сборка расширенной матрицы масс.

```
b_ii = area / 6.0
b_ij = area / 12.0

rows = np.concatenate((i, j, k, i, j, j, k, i, k))
cols = np.concatenate((i, j, k, j, i, k, j, k, i))
data = np.concatenate((b_ii, b_ii, b_ii,
                       b_ij, b_ij, b_ij,
                       b_ij, b_ij, b_ij))

B = sp.csr_matrix((data, (rows, cols)), shape=(x.size, x.size))
```

В листинге 9.5 приведен пример быстрой сборки расширенной матрицы масс (размер $M \times M$). При создании CSR-матрицы все элементы с одинаковыми индексами складываются. Аналогичным образом можно заполнить расширенную матрицу жесткости. Однако, там получится большее количество различающихся коэффициентов. С помощью срезов из расширенных матриц можно выделить только необходимые части.

Этап 6. На предыдущем этапе одним из способов были построены разреженные матрицы L и B, векторы правых частей f и g. Пример хорошей организации кода:

```
mesh, n_inner = get_mesh(h=0.02)

L, B, f, g = get_problem(mesh, n_inner)

L, B = get_matrices(mesh, n_inner)
```

Функция `get_mesh` строит сетку по характерному размеру `h`, функция возвращает триангуляцию `mesh` и число внутренних узлов сетки `n_inner`. Функция `get_problem` выполняет аппроксимацию уравнения Пуассона и возвращает матрицы и векторы, необходимые для решения неоднородного уравнения $Lu = Bf + g$. Функция `get_matrices` возвращает пару квадратных матриц L и B размера $N \times N$, которые подходят для решения однородных задач. Для метода конечных элементов функция `get_matrices` должна возвращать квадратные симметричные матрицы. Для неоднородных задач (МКЭ) вектор `f` имеет длину M, а матрица B имеет размер $N \times M$.

Уравнение Пуассона

Рассмотрим задачу Дирихле для уравнения Пуассона:

$$\operatorname{div}(\varkappa(x,\, y)\boldsymbol{\nabla} u) = -f(x,\, y), \qquad \varkappa(x,\, y) = \cos x \cos y,$$

$$f(x,\, y) = 2xy \sin y \cos x - 2x(1 + 3x)\cos x \cos y + (2x^3 + y^2)\sin x \cos y.$$

Точное решение и граничные условия заданы функцией:

$$u_0(x, y) = \frac{1}{2} + xy^2 + \frac{1}{2}x^4.$$

Листинг 9.6. Решение задачи Дирихле и визуализация.

```
L, B, f, g = get_problem(mesh, n_inner)

u = np.zeros_like(mesh.x)

# set boundary conditions
u[n_inner:] = bound(mesh.x[n_inner:], mesh.y[n_inner:])

# solve SLE for inner nodes
u[:n_inner] = sp.linalg.spsolve(L, B @ f + g)

fig = plt.figure(dpi=150, figsize=(5.8, 3.1))
if surface:
    # plot as surface
    ax = fig.add_subplot(projection='3d')
    ax.set_box_aspect([2.0, 1.0, 1.0])
    tcf = ax.plot_trisurf(mesh, u, cmap='RdBu')
else:
    # plot as colormap
    ax = plt.gca()
    ax.set_aspect('equal')
    tcf = ax.tricontourf(mesh, u, levels=30, cmap='RdBu')
    ax.colorbar(tcf, ax=ax)
plt.show()
```

Точное решения позволяет протестировать схему и проверить точность метода. Схема имеет второй порядок аппроксимации от характерного размера сетки $O(h^2)$. На Рис. 9.6 показано точное решение задачи $u_0(x, y)$, численное решение имеет погрешность порядка 10^{-5} и визуально не отличается от точного. Цветовая карта справа построена с помощью функции `tricontourf`, график поверхности построен функцией `plot_trisurf`.

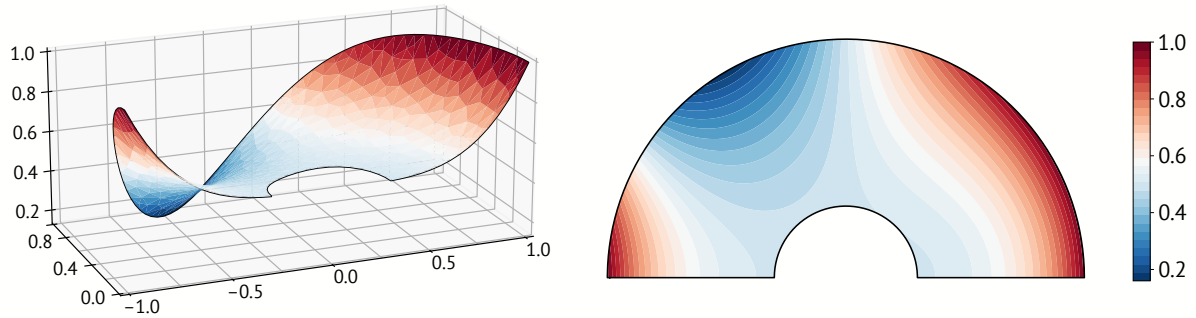

Рис. 9.6. Точное решение $u_0(x, y)$.

Собственные функции

Найдем собственные функции оператора $\mathcal{P}u = -\Delta u$ в рассматриваемой области. В Листинге 9.7 выполняется поиск 13 собственных значений и собственных функций с минимальными по модулю собственными значениями. Используется специальный метод `eigsh` для эрмитовых матриц. Затем строится пятая (начиная с нуля) собственная функция.

Листинг 9.7. Поиск и визуализация собственных функций.

```
L, B = get_matrices(mesh, n_inner)

eig_val, eig_vec = sp.linalg.eigsh(L, M=B, k=13, which='SM')

z = np.zeros_like(mesh.x)
z[:n_inner] = eig_vec[:, 5]

ax = plt.gca()
ax.set_aspect('equal')
ax.tricontourf(mesh, z, levels=30, cmap='RdBu')
plt.show()
```

Для поиска приближенных собственных значений оператора $\mathcal{P}$ решается обобщенная задача на собственные значения: $Lu = \lambda Bu$. Часто библиотеки линейной алгебры имеют специальные функции для решения такой задачи. В Python задача решается функцией `scipy.sparse.linalg.eigsh`, которая вызывает библиотеку ARPACK, написанную на языке Fortran.

Преимущество решения обобщенной задачи на собственные значения заключается в том, что могут быть использованы специализированные (более точные, быстрые и устойчивые) методы решения, предназначенные для эрмитовых матриц. Кроме того, для пары эрмитовых матриц L и B нам точно

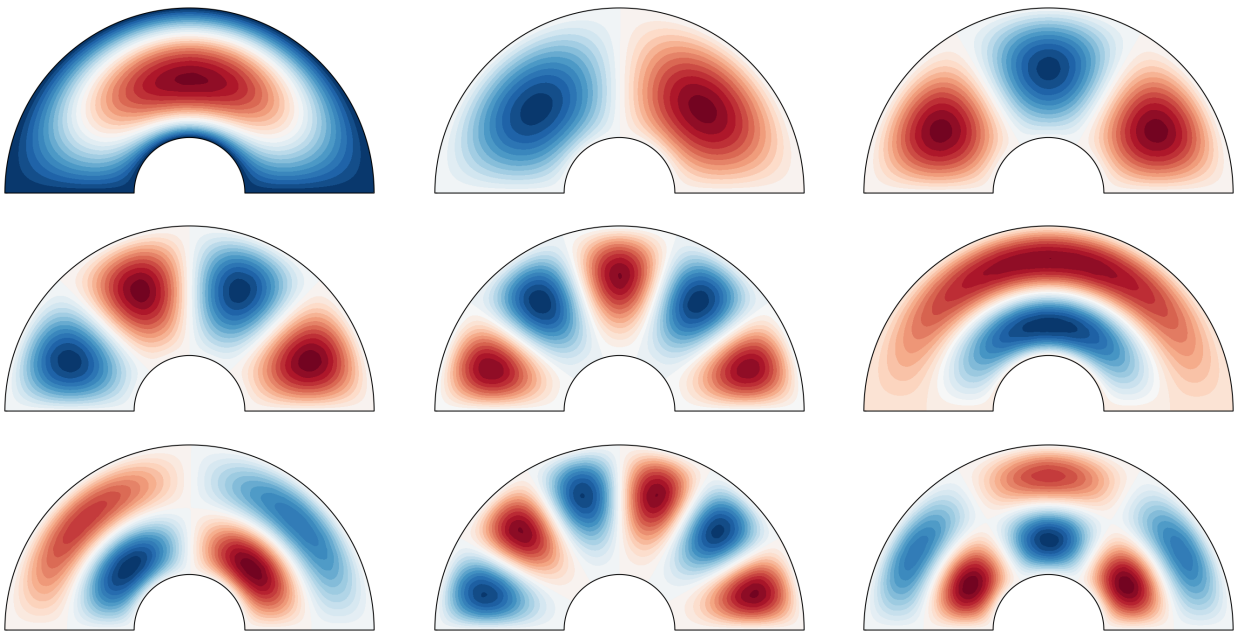

Рис. 9.7. Собственные функции оператора $\mathcal{P}$.

известно, что спектр является действительным, поэтому решения можно искать в поле действительных чисел.

На Рис. 9.7 приведены первые 9 собственных функций для оператора Лапласа в рассматриваемой области. В данной задаче собственные функции можно интерпретировать как моды колебаний двумерной мембраны, или как стационарные состояния квантовой частицы в двумерной бесконечно глубокой потенциальной яме с такой необычной формой.

Волновое уравнение

Рассмотрим однородное волновое уравнение с постоянным коэффициентом. Начальное возмущение задано внутри круга радиуса r_0, начальные скорости равны нулю:

$$\begin{cases} u_{tt} = a^2 \Delta u, & \boldsymbol{x} \in D, \quad 0 < t \\ u(\boldsymbol{x},\, t) = 0, & \boldsymbol{x} \in \partial D, \\ u(\boldsymbol{x},\, 0) = \eta(r_0 - |\boldsymbol{x} - \boldsymbol{x}_0|), \\ u_t(\boldsymbol{x},\, 0) = 0, \end{cases}$$

здесь $\eta(x)$ — функция Хевисайда. Динамика распространения волн приведена на Рис. 9.8. Для моделирования используется явная схема крест, которая после небольших преобразований принимает форму:

$$u^{k+1} = 2u^k - u^{k-1} - \tau^2 B^{-1} L u^k,$$

Данная формула без изменений воспроизведена в программе (Листинг 9.8).

В четвертой строке выполняется оценка спектрального радиуса оператора $B^{-1}L$. Для поиска спектрального радиуса используется функция `eigsh`. Для метода конечных объемов можно также использовать оценку (9.5). Спектральный радиус используется в 9 строке, чтобы определить шаг интегрирования по времени, который удовлетворяет условию устойчивости Куранта.

Для метода конечных объемов матрица B является диагональной, поэтому схему также можно записать в явном виде, если предварительно вычислить матрицу $B^{-1}L$. Для метода конечных элементов матрица B не является диагональной, поэтому вычислять матрицу $B^{-1}L$ не целесообразно, получившееся произведение не будет разреженной матрицей и, вероятнее всего, переполнит оперативную память. Вместо этого в коде используется функция `spsolve`, которая позволяет получить точное решение СЛАУ. Для диагональной матрицы B вызов `spsolve` не приведет к дополнительным вычислительным расходам, поскольку вычисление будет выполняться в один проход по диагональным элементам матрицы B.

Листинг 9.8. Моделирование волнового уравнения.

```
mesh, n_inner = get_mesh(h=0.01)

L, B = get_matrices(mesh, n_inner)

# Spectral radius of the operator
lambda_max = sp.linalg.eigsh(L, M=B, k=1, which='LM',
                             return_eigenvectors=False)[0]

CFL = 0.9 # Courant number

tau = CFL * 2.0 / np.sqrt(lambda_max)

x0, y0, r0 = -0.5, 0.3, 0.12
r = np.hypot(mesh.x - x0, mesh.y - y0)

u_prev = np.heaviside(r0 - r, 0.0)
u_curr = np.copy(u_prev)
u_next = np.zeros_like(xs)

def solution_step():
    global u_prev, u_curr, u_next

    u_next = 2 * u_curr - u_prev
    u_next[:n_inner] -= tau**2 * spsolve(B, L @ u_curr[:n_inner])
    u_prev, u_curr, u_next = u_curr, u_next, u_prev
```

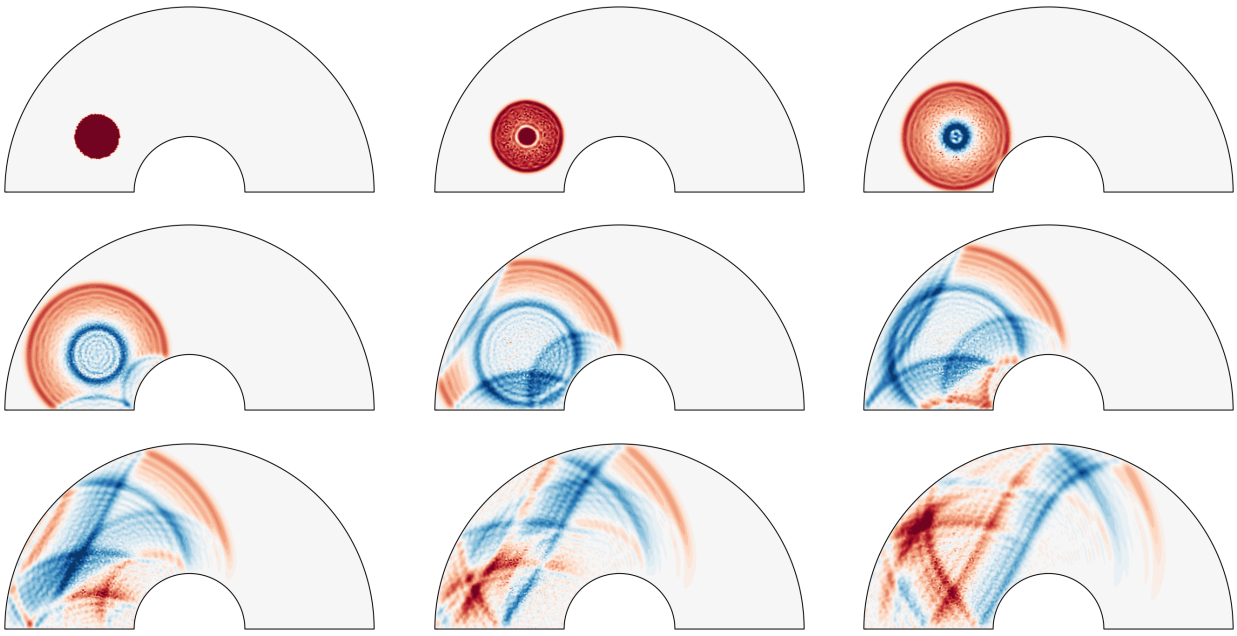

Рис. 9.8. Численное решение волнового уравнения. Начальное возмущение распространяется и отражается от границ области.

Уравнение Шрёдингера

Рассмотрим модельное уравнение Шрёдингера:

$$\begin{cases} i\psi_t = -\Delta\psi, & \boldsymbol{x} \in D, \quad 0 < t, \\ \psi(\boldsymbol{x}, t) = 0, & \boldsymbol{x} \in \partial D, \\ \psi(\boldsymbol{x}, 0) = \psi_0(\boldsymbol{x}). \end{cases}$$

Начальные условия заданы в форме волнового пакета:

$$\psi_0(\boldsymbol{x}) = \eta(r_0 - r)\cos^3\left(\frac{\pi r}{2r_0}\right) \cdot e^{-ik_x x}, \quad r = |\boldsymbol{x} - \boldsymbol{x}_0|,$$

где $\eta(x)$ — функция Хевисайда. В начальный момент плотность вероятности сосредоточена в круге $|\boldsymbol{x} - \boldsymbol{x}| < r_0$ и имеет куполообразную форму. Волновой пакет характеризуется волновым числом k_x и движется вправо. Динамика процесса приведена на Рис. 9.9. Графики демонстрируют «обтекание» границы волновым пакетом и отражение от границ.

Для численного моделирования используем схему Кранка – Николсон. Запишем схему в следующем виде:

$$S\psi^{k+1} = R\psi^k, \qquad S = iB + \nu\tau L, \quad R = iB - (1 - \nu)\tau L.$$

В таком виде она реализована в программе (Листинг 9.9). Используется параметр $\nu = \frac{1}{2}$, при таком значении сохраняется суммарная плотность вероятности, то есть величина $(\psi^{k+1}, \psi^{k+1})_B = (\psi^k, \psi^k)_B$.

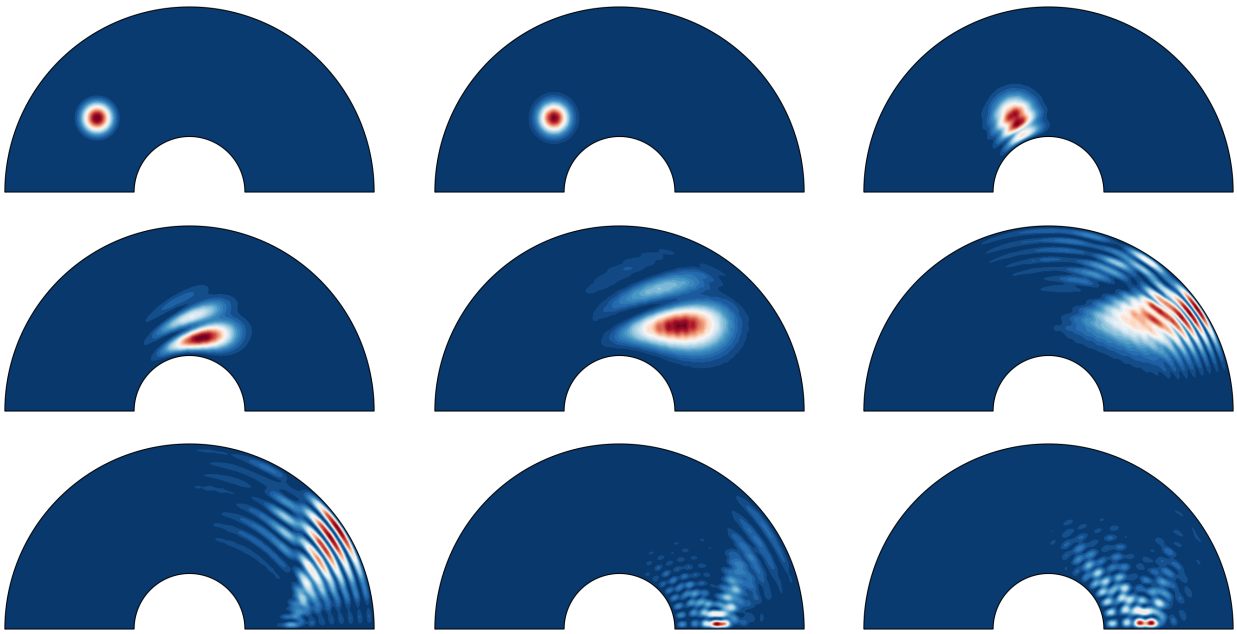

Рис. 9.9. Численное решение нестационарного уравнения Шрёдингера. Показан график плотности вероятности $|\psi|^2$.

Схема позволяет выбрать произвольный временной шаг τ. Для моделирования данной задачи целесообразно выбирать временной шаг $\tau \sim 1/k_x^2$, что позволит разрешить частоты, соответствующие k_x. В условии используется параметр $k_x = 60$, выбирается соответствующий шаг $\tau = 5 \cdot 10^{-5}$.

Листинг 9.9. Моделирование уравнения Шрёдингера.

```
mesh, n_inner = get_mesh(h=0.01)

L, B = get_matrices(mesh, n_inner)

nu = 0.5
tau = 5.0e-5

S = 1.0j * B + nu * tau * L
R = 1.0j * B - (1.0 - nu) * tau * L

# initial wave packet
x0, y0, r0 = -0.5, 0.4, 0.2
r = np.hypot(mesh.x - x0, mesh.y - y0)

amp = np.where(r < r0, np.cos(0.5 * np. pi * r / r0)**3, 0.0)
psi = amp * np.exp(-60.0j * mesh.x)

def solution_step(psi):
    psi[:n_inner] = sp.linalg.spsolve(S, R @ psi[:n_inner])
```

Список литературы

1. Самарский А. А., Гулин А. В. Численные методы. — М: Наука, 1989. — P. 432.

2. Ахиезер Н. И., Глазман И. М. Теория линейных операторов в гильбертовом пространстве. — Издательство Харьковского Университета, 1977.

3. Колмогоров А. Н., Юшкевич А. П. и др. Математика XIX века. Книга 1. Математическая логика, алгебра, теория чисел, теория вероятностей. — М: Наука, 1978. — С. 256.

4. Hart John F. Computer approximation. — Krieger Publishing Co., Inc., 1978.

5. Русский орфографический словарь / О. Е. Иванова, В. В. Лопатин, И. В. Нечаева и др. — Российская академия наук. Институт русского языка им. В. В. Виноградова, 2005. — С. 960.

6. Агеенко Флоренция. Словарь собственных имён русского языка. — Litress, 2022.

7. Абилов В. А., Керимов М. К. Об оценках остаточных членов кратных рядов Фурье–Чебышёва и кубатурных формул Чебышёвского типа // Журнал вычислительной математики и математической физики. — 2003. — Vol. 43, no. 5. — P. 643–663.

8. Remez E. Y. Sur le calcul effectif des polynomes d'approximation de Tschebyscheff // CR Acad. Sci. Paris. — 1934. — Vol. 199, no. 2. — P. 337–340.

9. Лебедев В. И., Финогенов С. А. О порядке выбора итерационных параметров в чебышёвском циклическом итерационном методе // Журнал вычислительной математики и математической физики. — 1971. — Vol. 11, no. 2. — P. 425–438.

10. Самарский А. А. Теория разносных схем. — М: Наука, 1977. — P. 656.

11. Вабищевич П. Н., Самарский А. А. Разностные схемы для задач конвекции-диффузии на нерегулярных сетках // Журнал вычислительной математики и математической физики. — 2000. — Vol. 40, no. 5. — P. 726–739.

12. Shivakumar P.N., Kim Ho Chew. A sufficient condition for nonvanishing of determinants // Proceedins of the American mathematical society. — 1974. — Vol. 43, no. 1. — P. 63–66.

13. Roberts Martin. The unreasonable effectiveness of quasirandom sequences. — 2018. — http://extremelearning.com.au/

unreasonable-effectiveness-ofquasirandom-sequences/ [Accessed: 01.04.2025].

14. Shimada Kenji. Physically-based mesh generation: automated triangulation of surfaces and volumes via bubble packing : Ph. D. thesis ; Massachusetts Institute of Technology. — 1993.

15. Lloyd Stuart. Least squares quantization in PCM // IEEE transactions on information theory. — 1982. — Vol. 28, no. 2. — P. 129–137.

16. Горюнов Анатолий Федорович. Методы математической физики в примерах и задачах. Том I. — М: Физматлит, 2015. — P. 872.

Приложение. Задачи к лабораторным работам

П.1. Собственные векторы и собственные значения

В тексте задач используются следующие обозначения: $\delta(x)$ — дельта-функция Дирака, $\eta(x)$ — функция Хевисайда, $J_n(x)$ — функция Бесселя первого рода порядка n.

Задача 1.1. Левый конец струны закреплен, а правый конец может перемещаться свободно в перпендикулярном направлении. Линейная плотность струны равна ρ, на свободном конце подвешена бусина массой m. Масса бусины равна четверти веса всей струны: $m = \frac{1}{4}\rho l$. Определить форму стоячих волн струны (*Горюнов А.Ф. [16], задача 2.11*).

$$\begin{cases} X'' = -\lambda\tilde{\rho}(x)X, \quad 0 < x < l, \\ X(0) = 0, \quad X'(l) = 0, \\ \tilde{\rho}(x) = 1 + \dfrac{m}{\rho}\delta(x-l). \end{cases}$$

Решение:

$$X_k(x) = \sin\sqrt{\lambda_k}x, \quad \lambda_k = \left(\frac{\mu_k}{l}\right)^2, \quad \mu_k - \text{корни} \quad \mu\operatorname{tg}\mu = \frac{\rho l}{m}.$$

Задача 1.2. В центре струны длиной l, закрепленной за два конца, подвешена бусина массой m. Линейная плотность струны равна ρ, а масса бусины равна половине веса всей струны: $m = \frac{1}{2}\rho l$. Определить форму стоячих волн струны (*Горюнов А.Ф. [16], задача 2.8*).

$$\begin{cases} X'' = -\lambda\tilde{\rho}(x)X, \quad -l < x < l, \\ X(-l) = X(l) = 0, \\ \tilde{\rho}(x) = 1 + \dfrac{m}{\rho}\delta(x). \end{cases}$$

Собственные функции распадаются на две серии:

$$X_k(x) = \sin\sqrt{\lambda_k}(l - |x|), \;\; \lambda_k = \left(\frac{\mu_k}{l}\right)^2, \;\; \mu_k - \text{корни } \mu\operatorname{tg}\mu = \frac{2\rho l}{m},$$

$$X_n(x) = \sin\sqrt{\lambda_n}x, \qquad \lambda_n = \left(\frac{\pi n}{l}\right)^2, \;\; n \in \mathbb{N}.$$

Собственные значения из двух серий чередуются, начиная с $\lambda_{k=1}$.

Задача 1.3. Концы неоднородной струны ($0 < x < l$) закреплены. Линейная плотность левой половины струны в 4 раза меньше плотности правой половины. Определить форму стоячих волн струны (*Горюнов А.Ф. [16], задача 2.7.1*).

$$\begin{cases} X'' = -\lambda\tilde{\rho}(x)X, \quad 0 < x < l, \\ X(0) = X(l) = 0, \\ \tilde{\rho}(x) = 1 + 3\eta(2x - l). \end{cases}$$

Общий вид собственных функций задачи:

$$X(x) = \begin{cases} \sin\sqrt{\lambda}x, & x < \dfrac{l}{2}, \\ B(\lambda)\sin\left(2\sqrt{\lambda}(l - x)\right), & x \geq \dfrac{l}{2}. \end{cases}$$

Собственные значения λ разделяются на три серии:

$$\sqrt{\lambda_k} = \frac{(2k-1)\pi \mp \arccos\frac{1}{3}}{l}, \quad k \in \mathbb{N}, \quad B(\lambda_k) = \frac{1}{2\cos\frac{\sqrt{\lambda_k}l}{2}},$$

$$\sqrt{\lambda_n} = \frac{2\pi n}{l}, \qquad n \in \mathbb{N}, \quad B(\lambda_n) = \frac{(-1)^{n+1}}{2}.$$

Собственные значения возрастают при чередовании значений из трех серий. Сначала идут значения из серии λ_k с минусом, потом λ_k с плюсом, затем λ_n.

Задача 1.4. Разложение в классические ряды Фурье осуществляется по собственным функциям, которые являются решением краевой задачи с периодическими граничными условиями.

$$\begin{cases} X'' = -\lambda X, \quad -l < x < l, \\ X(x) = X(x + 2l). \end{cases}$$

Положительные собственные значения λ_k вырождены и имеют кратность равную двум. Таким образом, каждому собственному значению $\lambda_k > 0$ соответствует пара линейно независимых собственных функций $X_k^{(1)}$ и $X_k^{(2)}$. В качестве пары таких функций удобно выбрать одну четную (косинус) функцию и одну нечетную (синус):

$$X_k^{(1)}(x) = \sin\sqrt{\lambda_k}\,x, \qquad \lambda_k = \left(\frac{\pi k}{l}\right)^2, \quad k \in \mathbb{N},$$

$$X_k^{(2)}(x) = \cos\sqrt{\lambda_k}\,x, \qquad \lambda_k = \left(\frac{\pi k}{l}\right)^2, \quad k \in \mathbb{N}_0.$$

Пусть для произвольного начального приближения метод Рэлея сходится к собственному значению λ_k. Метод не гарантирует сходимости к конкретной функции $X_k^{(1)}$ или $X_k^{(2)}$. В результате итераций, вероятнее всего, будет получена линейная комбинация этих функций $X_k(x)$. Восстановить $X_k^{(1)}$ и $X_k^{(2)}$ можно путем симметризации:

$$X_k^{(1)}(x) \sim \frac{X_k(x) - X_k(-x)}{2}, \quad X_k^{(2)}(x) \sim \frac{X_k(x) + X_k(-x)}{2}.$$

Задача 1.5. В точке $x = 0$ бесконечно глубокой потенциальной ямы ($|x| < l$) находится полупроницаемый барьер с коэффициентом проницаемости $\alpha = 50/l$. Определить уровни энергии и волновые функции стационарных состояний частицы в таком поле (*Горюнов А.Ф. [16], задача 2.63*).

$$\begin{cases} \left[-\dfrac{d^2}{dx^2} + \alpha\delta(x)\right]\psi = E\psi(x), \quad -l < x < l, \\ \psi(-l) = \psi(l) = 0. \end{cases}$$

Решение задачи распадается на две серии:

$$\psi_k(x) = \sin\sqrt{\lambda_k}(l - |x|), \;\; E_k = \left(\frac{\mu_k}{l}\right)^2, \;\; \mu_k - \text{корни } \mu\operatorname{ctg}\mu = -\frac{\alpha l}{2},$$

$$\psi_n(x) = \sin\sqrt{\lambda_n}x, \qquad E_n = \left(\frac{\pi n}{l}\right)^2, \;\; n \in \mathbb{N}.$$

Собственные значения из двух серий чередуются, начиная с $E_{k=1}$.

Задача 1.6. Частица находится в бесконечно глубокой потенциальной яме $0 < x < l$. В части ямы $x > l/3$ потенциал равен $U_0 = 900/l^2$. Определить первые уровни энергии частицы, которые превышают значения U_0. Найти соответствующие волновые функции стационарных состояний частицы.

$$\begin{cases} \left[-\dfrac{d^2}{dx^2} + U_0\,\eta(3x - l)\right]\psi = E\psi(x), \quad 0 < x < l, \\ \psi(0) = \psi(l) = 0. \end{cases}$$

Решение задачи для $E > U_0$:

$$E_k = \left(\frac{3\mu_k}{l}\right)^2, \quad \psi_k(x) = \begin{cases} \sin\dfrac{2\sqrt{E_k - U_0}\,l}{3}\sin\sqrt{E_k}x, & x < \dfrac{l}{3}, \\ \sin\dfrac{\sqrt{E_k}\,l}{3}\,\sin\left(\sqrt{E_k - U_0}\,(l - x)\right), & x \geq \dfrac{l}{3}, \end{cases}$$

где μ_k – корни уравнения $\sqrt{\mu^2 - p^2}\ \mathrm{tg}\,\mu + \mu\,\mathrm{tg}\left(2\sqrt{\mu^2 - p^2}\right) = 0$, параметр $p = \frac{l\sqrt{U_0}}{3} = 10$. Последнее уравнение имеет бесконечное число положительных корней $\mu_k > p$, что эквивалентно условию $E_k > U_0$.

Задача 1.7. Частица находится в потенциальной яме с потенциалом $U = +\infty$ при $x < 0$ и постоянным потенциалом $U_0 = 400/l^2$ при $x > l$. Найти уровни энергии связанных состояний частицы и соответствующие волновые функции (*Горюнов А.Ф. [16], задача 2.64*).

$$\begin{cases} \left[-\dfrac{d^2}{dx^2} + U_0\,\eta(x-l)\right]\psi = E\psi(x), \quad 0 < x, \\ \psi(0) = 0, \quad \lim\limits_{x\to\infty}\psi(x) = 0. \end{cases}$$

В связанном состоянии находятся частицы, которые нельзя обнаружить на бесконечности, в данной задаче это частицы с энергиями $E < U_0$. Вообще говоря, для произвольного потенциала U_0 не очевидно, что такие состояния есть. Численное моделирование провести в области $x \in [0,\ 2l]$ с заданием нулевого условия на правой границе.

Точное решение задачи:

$$E_k = \left(\frac{\mu_k}{l}\right)^2, \quad \psi_k(x) = \begin{cases} e^{-\sqrt{U_0 - E_k}\,l}\,\sin\sqrt{E_k}x, & x < l, \\ \sin\left(\sqrt{E_k}\,l\right)e^{-\sqrt{U_0 - E_k}\,x}, & x \geq l, \end{cases}$$

где μ_k – корни $\sin\mu = \pm\dfrac{\mu}{\mu_0}$ такие, что $\mathrm{tg}\,\mu < 0$, $\mu_0 = l\sqrt{U_0} = 20$. Уравнение на константы μ имеет конечное число корней до $\mu < \mu_0$, что эквивалентно условию связанного состояния $E < U_0$.

Задача 1.8. В неглубокой потенциальной яме ($-l < x < l$) находится частица. Потенциал вне ямы равен $U_0 = 400/l^2$. Найти уровни энергии связанных состояний частицы и соответствующие волновые функции. Задачу решать в ограниченной области $[-2l,\ 2l]$ с заданием нулевых граничных условий (*Горюнов А.Ф. [16], задача 2.65*).

$$\begin{cases} \left[-\dfrac{d^2}{dx^2} + U_0\,\eta(|x| - l)\right]\psi = E\psi(x), \quad x \in \mathbb{R}, \\ \lim\limits_{x\to\pm\infty}\psi(x) = 0. \end{cases}$$

В решении чередуются квантовые состояния с четными −

$$E_k = \left(\frac{\mu_k}{l}\right)^2, \quad \psi_k(x) = \begin{cases} e^{-\sqrt{U_0 - E_k}\, l} \cos\sqrt{E_k}x, & |x| < l, \\ \cos\left(\sqrt{E_k}\, l\right) e^{-\sqrt{U_0 - E_k}\,|x|}, & |x| \geq l, \end{cases}$$

где μ_k − корни $\cos\mu = \pm\frac{\mu}{\mu_0}$ такие, что $\operatorname{tg}\mu > 0$, $\mu_0 = l\sqrt{U_0} = 20$, и нечетными волновыми функциями:

$$E_k = \left(\frac{\mu_k}{l}\right)^2, \quad \psi_k(x) = \begin{cases} e^{-\sqrt{U_0 - E_k}\, l} \sin\sqrt{E_k}x, & |x| < l, \\ \sin\left(\sqrt{E_k}\, l\right) e^{-\sqrt{U_0 - E_k}\,|x|} \operatorname{sgn}(x), & |x| \geq l, \end{cases}$$

где μ_k − корни $\sin\mu = \pm\frac{\mu}{\mu_0}$ такие, что $\operatorname{tg}\mu < 0$, $\mu_0 = l\sqrt{U_0} = 20$.

Задача 1.9. Найти волновые функции и уровни энергии стационарных состояний частицы в треугольной потенциальной яме $U(x > 0) = \alpha^3 x$, где $\alpha > 0$ − характеристика наклона.

$$\begin{cases} \left[-\frac{d^2}{dx^2} + \alpha^3 x\right]\psi = E\psi(x), \quad 0 < x, \\ \psi(0) = 0, \quad \lim\limits_{x\to\infty} \psi(x) = 0. \end{cases}$$

Задачу следует моделировать в ограниченной области $x \in [0,\, l]$ с заданием нулевого условия на правой границе. Использовать $l \gg 1/\alpha$, к примеру $l = 15/\alpha$. Точное решение:

$$\psi_k(x) = \mathrm{Ai}(\alpha x - \mu_k), \quad E_k = \alpha^2\mu_k, \quad \mu_k - \text{корни } \mathrm{Ai}(-\mu) = 0,$$

где Ai − функция Эйри.

Задача 1.10. Найти волновые функции и уровни энергии стационарных состояний частицы внутри параболической потенциальной ямы (гармонический осциллятор). Потенциал $U(x) = \omega^2 x^2$.

$$\begin{cases} \left[-\frac{d^2}{dx^2} + U(x)\right]\psi = \lambda\psi(x), \quad x \in \mathbb{R}, \\ \lim\limits_{x\to\infty} \psi(x) = 0. \end{cases}$$

Задачу следует моделировать в ограниченной области $x \in [-l,\, l]$ с заданием нулевых условий на границах. Использовать $l \gg 1/\omega$, к примеру $l = 8/\omega$.

Точное решение:

$$\psi_k(x) = \exp\left(-\frac{\omega x^2}{2}\right) H_k\left(\sqrt{\omega}x\right), \qquad E_k = (2k+1)\,\omega,$$

где H_k — полиномы Эрмита.

Задача 1.11. В центре круговой потенциальной ямы ($r < r_0$) задан потенциал $U(r) = 4/r^2$. Определить уровни энергии и волновые функции стационарных состояний частицы в таком поле.

$$\begin{cases} \left(-\frac{1}{r}\frac{d}{dr}r\frac{d}{dr} + U(r)\right)\psi(r) = \lambda\psi(r), \quad 0 < r < r_0, \\ \psi(r_0) = 0, \qquad \lim\limits_{r\to 0} r\,\psi'(r) = 0. \end{cases}$$

Решения выражаются через функции Бесселя:

$$\psi_k(r) = J_2\left(\frac{\mu_k r}{r_0}\right), \quad E_k = \left(\frac{\mu}{r_0}\right)^2, \quad \mu_k - \text{корни} \quad J_2(\mu) = 0.$$

Задача 1.12. Найти волновые функции и уровни энергии стационарных состояний частицы в бесконечно глубокой сферической потенциальной яме ($r < r_0$).

$$\begin{cases} \frac{1}{r^2}\frac{d}{dr}r^2\frac{d\psi}{dr} = -E\psi(r), \quad 0 < r < r_0, \\ \psi(r_0) = 0, \qquad \lim\limits_{r\to 0} r\psi'(r) = 0. \end{cases}$$

Решение:

$$\psi_k(r) = \frac{\sin\sqrt{\lambda_k}r}{r}, \qquad E_k = \left(\frac{\pi k}{r_0}\right)^2, \quad k \in \mathbb{N}.$$

Задача 1.13. Найти волновые функции и уровни энергий электрона в атоме водорода без учета углового момента. Рассмотреть модель атома с кулоновским потенциалом $U(r) = -2/(a_0 r)$, где a_0 — боровский радиус:

$$\begin{cases} \left(-\frac{1}{r^2}\frac{d}{dr}r^2\frac{d}{dr} - \frac{2}{a_0 r}\right)\psi = E\psi(r), \quad 0 < r, \\ \psi(r_0) = 0, \qquad \lim\limits_{r\to 0} r\psi'(r) = 0. \end{cases}$$

Моделирование проводить в ограниченной области $r < r_0$, с заданием нулевых условий на границе. Использовать $r_0 \gg a_0$, для получения первых энергетических уровней достаточно выбрать $r_0 = 100a_0$.

Точное решение:

$$\psi_k(r) = \exp\left(-\frac{r}{a_0 k}\right) L_k^1\left(\frac{2r}{a_0 k}\right), \qquad E_k = -\frac{1}{(ak)^2}, \qquad k \in \mathbb{N},$$

где L_k^1 — обобщенные полиномы Лагерра. Построить графики плотности вероятности $p(r) = |\psi(r)|^2 r^2$ для самых низких уровней энергии. Какое положение является наиболее вероятным для электрона в основном состоянии?

Задача 1.14. Найти волновые функции и уровни энергии связанных состояний частицы в потенциале Пёшль – Теллера:

$$\begin{cases} \left[-\dfrac{d^2}{dx^2} + U(x)\right]\psi = E\psi(x), \quad x \in \mathbb{R}, \\ \lim\limits_{x\to\infty} \psi(x) = 0. \end{cases}$$

Использовать потенциал $U(x) = U_0 \tanh^2(ax)$, который на константу отличается от оригинального. Задачу следует моделировать в ограниченной области $|x| < l$ с заданием нулевого условия на правой границе. Использовать $l \gg 1/a$, к примеру $l = 6/a$. Если потенциал $U_0 = m(m+1)a^2$, то существует m связанных состояний ($E < U_0$) в потенциальной яме:

$$\psi_k(x) = P_m^k(\tanh(ax)), \quad E_k = U_0 - (ak)^2, \qquad k = \overline{1,\, m}.$$

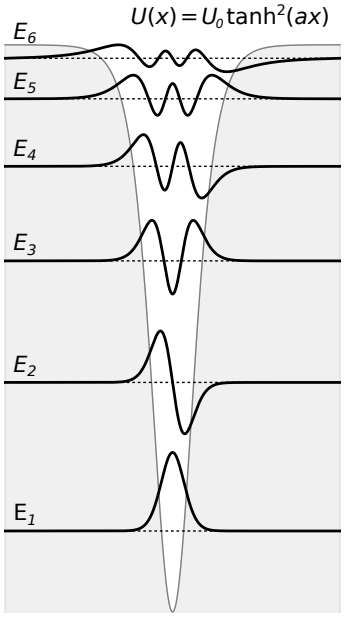

Рис. П.1.1. Решение задачи 1.14

Здесь P_m^k — присоединенные функции Лежандра. Собственные значения нумеруются, начиная с высоких энергий.

На рис. П.1.1 показан потенциал Пешль – Теллера и волновые функции связанных состояний частицы в таком потенциале. Волновые функции ψ_k на графике смещены на соответствующие уровни энергии E_k.

Задача 1.15. Полиномы Лежандра встречаются при решении задач математической физики в сферических координатах. Найти численно первые полиномы Лежандра как решение задачи на собственные функции:

$$\begin{cases} \dfrac{d}{dx}(1-x^2)\dfrac{du}{dx} = -\lambda u(x), \quad -1 < x < 1, \\ \lim\limits_{x\to\pm1} (1-x^2)u'(x) = 0. \end{cases}$$

Решение задачи через полиномы Лежандра $P_k(x)$:

$$u_k(x) = P_k(x), \qquad \lambda_k = k(k+1), \quad k \in \mathbb{N}_0.$$

Задача 1.16. Присоединенные функции Лежандра встречаются при решении задач математической физики в сферических координатах. В частности, через них вводятся сферические функции. Присоединенные функции Лежандра являются ограниченными решениями задачи на собственные функции:

$$\begin{cases} \dfrac{d}{dx}(1-x^2)\dfrac{du}{dx} - \dfrac{m^2}{1-x^2}u = -\lambda u(x), \quad -1 < x < 1, \\ \lim\limits_{x \to \pm 1} (1-x^2)u'(x) = 0. \end{cases}$$

Приближенно найти присоединенные функции Лежандра P_n^m для $m = 1$ как решение задачи на собственные функции и собственные значения.

$$u_n(x) = P_n^m(x), \quad \lambda_n = n(n+1), \quad n \in \mathbb{N}, \quad n \geq m.$$

Задача 1.17. Струна длиной l зафиксирована с двух концов. Определить моды колебаний струны, которые соответствуют самым низким собственным частотам.

$$\begin{cases} X'' = -\lambda X, \quad 0 < x < l, \\ X(0) = X(l) = 0. \end{cases}$$

Решение:

$$X_k(x) = \sin\sqrt{\lambda_k}\, x, \qquad \lambda_k = \left(\frac{\pi k}{l}\right)^2, \quad k \in \mathbb{N}.$$

Задача 1.18. Концы струны длиной l могут свободно смещаться в перпендикулярном направлении. Определить моды колебаний струны, которые соответствуют самым низким собственным частотам.

$$\begin{cases} X'' = -\lambda X, \quad 0 < x < l, \\ X'(0) = X'(l) = 0. \end{cases}$$

Решение:

$$X_k(x) = \cos\sqrt{\lambda_k}\, x, \qquad \lambda_k = \left(\frac{\pi k}{l}\right)^2, \quad k \in \mathbb{N}_0.$$

Задача 1.19. Левый конец струны зафиксирован, а правый может свободно смещаться в перпендикулярном направлении. Определить моды колебаний струны, которые соответствуют самым низким собственным частотам.

$$\begin{cases} X'' = -\lambda X, \quad 0 < x < l, \\ X(0) = 0, \quad X'(l) = 0. \end{cases}$$

Решение:

$$X_k(x) = \sin\sqrt{\lambda_k}\,x, \qquad \lambda_k = \left(\frac{\pi(2k+1)}{2l}\right)^2, \quad k \in \mathbb{N}_0.$$

Задача 1.20. Левый конец струны зафиксирован, а правый — закреплен через пружину. Определить моды колебаний струны, соответствующие самым низким собственным частотам.

$$\begin{cases} X'' = -\lambda X, & 0 < x < l, \\ X(0) = 0, & 3X(l) + lX'(l) = 0. \end{cases}$$

Решение:

$$X_k(x) = \sin\sqrt{\lambda_k}\,x, \quad \lambda_k = \left(\frac{\mu_k}{l}\right)^2, \quad \mu_k - \text{корни } \mu + 3\operatorname{tg}\mu = 0.$$

Задача 1.21. Круглая мембрана радиусом r_0 закреплена по краю. Определить моды колебаний мембраны, которые соответствуют самым низким собственным частотам.

$$\begin{cases} \dfrac{1}{r}\dfrac{d}{dr}r\dfrac{dR}{dr} = -\lambda R(r), & 0 < r < r_0, \\ R(r_0) = 0, \qquad \lim\limits_{r\to 0} rR'(r) = 0. \end{cases}$$

Решение (J_k — функции Бесселя первого рода):

$$R_k(r) = J_0\left(\frac{\mu_k r}{r_0}\right), \quad \lambda_k = \left(\frac{\mu_k}{r_0}\right)^2, \quad \mu_k - \text{корни } J_0(\mu) = 0.$$

Задача 1.22. Края круглой мембраны радиусом r_0 могут свободно смещаться в перпендикулярном направлении. Определить моды колебаний мембраны, которые соответствуют самым низким собственным частотам.

$$\begin{cases} \dfrac{1}{r}\dfrac{d}{dr}r\dfrac{dR}{dr} = -\lambda R(r), & 0 < r < r_0, \\ R'(r_0) = 0, \qquad \lim\limits_{r\to 0} rR'(r) = 0. \end{cases}$$

Решение (J_k — функции Бесселя первого рода):

$$R_k(r) = J_0\left(\frac{\mu_k r}{r_0}\right), \quad \lambda_k = \left(\frac{\mu_k}{r_0}\right)^2, \quad \mu_k - \text{корни } J_1(\mu) = 0.$$

Задача 1.23. Края круглой мембраны радиусом r_0 закреплены через пружину. Определить моды колебаний мембраны, которые соответствуют самым низким собственным частотам.

$$\begin{cases} \dfrac{1}{r}\dfrac{d}{dr}r\dfrac{dR}{dr} = -\lambda R(r), \quad 0 < r < r_0, \\ 20R(r_0) + r_0R'(r_0) = 0, \qquad \lim\limits_{r\to 0} rR'(r) = 0. \end{cases}$$

Решение (J_k — функции Бесселя первого рода):

$$R_k(r) = J_0\left(\frac{\mu_k r}{r_0}\right), \quad \lambda_k = \left(\frac{\mu_k}{r_0}\right)^2, \quad \mu_k - \text{корни } 20J_0(\mu) = \mu J_1(\mu).$$

Задача 1.24. Найти волновые функции и уровни энергии стационарных состояний частицы в бесконечно глубокой потенциальной яме:

$$\begin{cases} \dfrac{d^2\psi}{dx^2} = -\lambda\psi(x), \quad 0 < x < l, \\ \psi(0) = \psi(l) = 0. \end{cases}$$

Волновые функции:

$$\psi_k(x) = \sin\sqrt{\lambda_k}\, x, \qquad \lambda_k = \left(\frac{\pi k}{l}\right)^2, \quad k \in \mathbb{N}.$$

Задача 1.25. Найти волновые функции и уровни энергии стационарных состояний частицы в бесконечно глубокой круглой потенциальной яме:

$$\begin{cases} \dfrac{1}{r}\dfrac{d}{dr}r\dfrac{d\psi}{dr} = -\lambda\psi(r), \quad 0 < r < r_0, \\ \psi(r_0) = 0, \qquad \lim\limits_{r\to 0} r\psi'(r) = 0. \end{cases}$$

Решение:

$$\psi_k(r) = J_0\left(\frac{\mu_k r}{r_0}\right), \quad \lambda_k = \left(\frac{\mu_k}{r_0}\right)^2, \quad \mu_k - \text{корни } J_0(\mu) = 0.$$

Задача 1.26. Функции Бесселя 1-го порядка встречаются при решении задач математической физики в полярных координатах. К примеру, задача о колебаниях закрепленной мембраны. Найти собственные функции,

соответствующие наименьшим собственным значениям.

$$\begin{cases} \dfrac{1}{r}\dfrac{d}{dr}r\dfrac{d}{dr}R(r) - \dfrac{1}{r^2}R(r) = -\lambda R(r), \quad 0 < r < r_0, \\ R(r_0) = 0, \qquad \lim\limits_{r\to 0} rR'(r) = 0. \end{cases}$$

Решения выражаются через функции Бесселя:

$$R_k(r) = J_1\left(\frac{\mu_k r}{r_0}\right), \quad \lambda_k = \left(\frac{\mu_k}{r_0}\right)^2, \quad \mu_k - \text{корни} \quad J_1(\mu) = 0.$$

П.2. Эллиптические уравнения

Каждая задача в разделе включает два пункта. Задание в пункте (а) приведено в полной постановке и с точным решением. Пункт (а) подходит для верификации численного метода. Наличие точного решения позволяет оценить порядок аппроксимации, скорость сходимости методов и т.д.

Для решения пункта (б) сначала следует выполнить математическую постановку задачи. Тип уравнения и граничных условий в пункте (б) такой же, как и для задачи в пункте (а), это должно упростить постановку.

Задача 2.1. Найти решение уравнения Пуассона с граничными условиями Дирихле в прямоугольнике:

а). $$\begin{cases} \Delta u = -f(x,\,y), & 0 < x < 1, \quad 0 < y < \sqrt{2}, \\ u(0,\,y) = 0, & u(1,\,y) = \sin(5)\cos\frac{\pi y^2}{2}, \\ u(x,\,0) = \sin 5x, & u(x,\,\sqrt{2}) = -\sin 5x; \end{cases}$$

$$f(x,\,y) = \sin 5x\Big(\big(25 + \pi^2 y^2\big)\cos\frac{\pi y^2}{2} + \pi\sin\frac{\pi y^2}{2}\Big).$$

Точное решение: $u_0(x,\,y) = \sin 5x\cos\dfrac{\pi y^2}{2}$.

б). Квадратную металлическую пластинку достаточно долго нагревают точечным лазерным излучением с постоянной мощностью. Края пластинки поддерживаются при нулевой температуре. Построить распределение температуры в пластинке.

Задача 2.2. Найти решение уравнения Пуассона с граничными условиями Неймана по оси y и периодическими условиями вдоль оси x:

а). $$\begin{cases} \Delta u = (9y - 10)\sin 4x\, e^{-5y}, & 0 < x < \pi, \quad 0 < y < 1, \\ u(x + \pi,\,y) = u(x,\,y), \\ u_y(x,\,0) = \sin 4x, & u_y(x,\,1) = -\dfrac{4}{e^5}\cdot\sin 4x. \end{cases}$$

Точное решение: $u_0(x,\,y) = \sin 4x\cdot y\, e^{-5y}$.

б). В доме сделан водяной теплый пол. Трубки с горячей водой имеют радиус r_0, они расположены параллельно друг другу на расстоянии L и залиты бетонной стяжкой толщиной H. Трубки создают постоянный источник тепла Q в толще бетона. Полагаем, что нижняя часть стяжки хорошо теплоизолирована, а полученное от труб тепло равномерно рассеивается с поверхности пола. Найти распределение температуры в толще бетона.

Задача 2.3. Решить задачу Дирихле в прямоугольнике для экранированного уравнения Пуассона:

а). $$\begin{cases} \Delta u - 4u = 0, \qquad -1 < x < 1, \quad -1 < y < 1, \\ u(-1,\, y) = u(1,\, y) = \sinh(1 - y)\sinh(1 + y), \\ u(x,\, -1) = u(x,\, 1) = \sinh(x + 1)\sinh(x - 1). \end{cases}$$

Точное решение: $u_0(x,\, y) = \sinh(x + y)\sinh(x - y)$.

б). На прямоугольный пружинный матрас кладут тонкое тяжелое кольцо массой M. Пружины матраса распределены равномерно и имеют жесткость k. Построить поверхность матраса, считать, что края матраса закреплены.

Задача 2.4. Решить задачу Неймана для уравнения Пуассона:

а). $$\begin{cases} \Delta u = -6, \qquad -1 < x < 1, \quad -1 < y < 1, \\ u_x(-1,\, y) = -12, \qquad u_x(1,\, y) = 4, \\ u_y(x,\, -1) = 14, \qquad u_y(x,\, 1) = -14. \end{cases}$$

Точное решение: $u_0(x,\, y) = (1 - 2x)^2 - 7y^2$.

б). По трубе с сечением S течет несжимаемая жидкость. Жидкость поступает в трубу слева через отверстие с сечением $S_1 = 0,5S$ со скоростью v_1 и вытекает справа через два отверстия с сечениями $S_2 = 0,05S$ со скоростью v_2. Построить векторное поле скорости жидкости в трубе.

Задача 2.5. Найти решение уравнения Пуассона с граничными условиями Дирихле по оси x и периодическими условиями по оси y:

а). $$\begin{cases} \Delta u = 4 - 29\sin(5x + 2y), \qquad -1 < x < 1, \quad 0 < y < \pi, \\ u(-1,\, y) = 2 + \sin(2y - 5), \qquad u(1,\, y) = 2 + \sin(2y + 5), \\ u(x,\, y + \pi) = u(x,\, y). \end{cases}$$

Точное решение: $u_0(x,\, y) = 2x^2 + \sin(5x + 2y)$.

б). Между обкладками бесконечного конденсатора (моделируется периодическими граничными условиями) помещены два заряда $-\ q_1$ и q_2. Построить линии напряженности электростатического поля.

Задача 2.6. Решить задачу Неймана для экранированного уравнения Пуассона в прямоугольнике:

а). $$\begin{cases} \Delta u - 4u = 12\big(1 - 2x^2\big), \qquad -1 < x < 1, \quad -1 < y < 1, \\ u_x(-1,\, y) = 2\cosh 2 - 12, \qquad u_x(1,\, y) = 2\cosh 2 + 12, \\ u_y(x,\, -1) = 2\sinh 2, \qquad u_y(x,\, 1) = -2\sinh 2. \end{cases}$$

Точное решение: $u_0(x,\, y) = 6x^2 + \sinh 2x - \cosh 2y$.

б). В прямоугольную камеру через точечное отверстие с постоянной скоростью поступает газ, который содержит радиоактивный изотоп с коэффициентом распада β. Через некоторое время в камере устанавливается стационарное распределение изотопа. Построить его концентрацию.

Задача 2.7. Найти решение уравнения Пуассона в прямоугольной периодически замкнутой области:

а). $$\begin{cases} \Delta u = -25\sin(3x+4y) - 32\cos 4y, \quad 0 < x < 2\pi,\ 0 < y < \pi, \\ u(x+2\pi,\ y) = u(x,\ y), \qquad u(x,\ y+\pi) = u(x,\ y). \end{cases}$$

Точное решение: $u_0(x,\ y) = \sin(3x+4y) + 2\cos 4y.$

б). В узлах квадратной решетки расположены в шахматном порядке положительно и отрицательно заряженные ионы. Ионы имеют одинаковый по модулю заряд, т.е. решетка является электронейтральной. Построить эквипотенциальные линии электростатического поля, которое создает решетка.

Задача 2.8. Найти решение уравнения Пуассона с условиями Дирихле и Неймана на границах:

а). $$\begin{cases} \Delta u = \left[6 - 72\left(4(1-2x)^2 + (1-3y)^2\right)\theta(x,\ y)\right]\left(1 - \theta^2(x,\ y)\right), \\ \theta(x,\ y) = \tanh\left(3(1-2x)^2 - (1-3y)^2\right),\ 0 < x < 1,\ 0 < y < 1, \\ u(0,\ y) = u(1,\ y) = \tanh\left(3 - (1-3y)^2\right), \\ u_y(x,\ 0) = 6\left(1 - \tanh^2\left(3(1-2x)^2 - 1\right)\right), \\ u_y(x,\ 1) = -12\left(1 - \tanh^2\left(3(1-2x)^2 - 4\right)\right). \end{cases}$$

Точное решение: $u_0(x,\ y) = \theta(x,\ y).$

б). На подвешенный гамак кладут тяжелое тонкое кольцо массой M. Построить поверхность гамака. Предполагается, что два противоположных края гамака зафиксированы, а другие два свободны.

Задача 2.9. Найти решение экранированного уравнения Пуассона в квадратной периодически замкнутой области:

а). $$\begin{cases} \Delta u - u = -f(x,\ y), \qquad 0 < x < \pi, \quad 0 < y < \pi, \\ u(x+\pi,\ y) = u(x,\ y), \qquad u(x,\ y+\pi) = u(x,\ y); \end{cases}$$

$$f(x,\ y) = \sin 2x\, e^{-\cos 2y}\,(2\cos 2y - 1)^2.$$

Точное решение: $u_0(x,\ y) = \sin 2x\, e^{-\cos 2y}.$

б). В узлах квадратной решетки расположены в шахматном порядке положительно и отрицательно заряженные ионы. Положительные ионы имеют

заряд q_1, отрицательные $-q_2$. Предполагаем, что решетка создает электростатическое поле, которое удовлетворяет экранированному уравнению Пуассона. Построить эквипотенциальные линии поля.

Задача 2.10. Найти решение уравнения Пуассона с граничными условиями Дирихле в прямоугольнике:

а). $$\begin{cases} \Delta u = -f(x,\,y), & 0 < x < \sqrt{2}, \quad 0 < y < 1, \\ u(0,\,y) = \sin 5y, & u(\sqrt{2},\,y) = -\sin 5y, \\ u(x,\,0) = 0, & u(x,\,1) = \sin(5)\cos\dfrac{\pi x^2}{2}; \end{cases}$$

$$f(x,\,y) = \sin 5y\left(\left(25 + \pi^2 x^2\right)\cos\frac{\pi x^2}{2} + \pi\sin\frac{\pi x^2}{2}\right).$$

Точное решение: $u_0(x,\,y) = \sin 5y \cdot \cos\dfrac{\pi x^2}{2}$.

б). Эластичный тент натянут на четыре параболических опоры, $f(x) = x\,(3 - x)$, расположенных по сторонам квадрата 3×3 м. Построить поверхность получившегося шатра.

Задача 2.11. Найти решение уравнения Пуассона с граничными условиями Неймана по оси x и периодическими условиями вдоль оси y:

а). $$\begin{cases} \Delta u = (9x - 10)\sin 4y\, e^{-5x}, & 0 < x < 1, \quad 0 < y < \pi, \\ u_x(0,\,y) = \sin 4y, & u_x(1,\,y) = -\dfrac{4}{e^5}\sin 4y, \\ u(x,\,y + \pi) = u(x,\,y). \end{cases}$$

Точное решение: $u_0(x,\,y) = x\,e^{-5x}\,\sin 4y$.

б). Бесконечная металлическая пластина имеет множество прямых параллельных прорезей, расположенных с периодом L, ширина прорезей равна αL. Рядом на расстоянии H расположена точно такая же пластина. Смоделировать движение несжимаемой жидкости через такую систему из двух решеток. Построить векторное поле скорости.

Задача 2.12. Решить задачу Дирихле для экранированного уравнения Пуассона в прямоугольнике:

а). $$\begin{cases} \Delta u - 4u = 0, & -1 < x < 1, \quad -1 < y < 1, \\ u(-1,\,y) = u(1,\,y) = \sinh(y + 1)\sinh(y - 1), \\ u(x,\,-1) = u(y,\,1) = \sinh(1 - x)\sinh(1 + x). \end{cases}$$

Точное решение: $u_0(x,\,y) = \sinh(x + y)\sinh(y - x)$.

б). Прямоугольную камеру заполняют газом, который содержит радиоактивный изотоп с коэффициентом распада β. Газ поступает в камеру через 12 форсунок, расположенных по окружности. Мощность всех форсунок одинакова и постоянна. Концентрация изотопа на границе камеры поддерживается равной нулю. Построить распределение изотопа в камере, которое установится спустя некоторое время.

Задача 2.13. Решить задачу Неймана для уравнения Пуассона:

а). $$\begin{cases} \Delta u = -6, & -1 < x < 1, \quad -1 < y < 1, \\ u_x(-1,\, y) = 14, & u_x(1,\, y) = -14, \\ u_y(x,\, -1) = -12, & u_y(x,\, 1) = 4. \end{cases}$$

Точное решение: $u_0(x,\, y) = -7x^2 + (1 - 2y)^2$.

б). Прямоугольная камера размером $[0,\, l_x] \times [0,\, l_y]$ заполнена несжимаемой жидкостью. Жидкость втекает с постоянной скоростью через левую границу камеры $x \in [x_1,\, l_x]$ и вытекает с постоянной скоростью через нижнюю границу $y \in [y_1,\, l_y]$. Построить поле скорости жидкости.

Задача 2.14. Найти решение уравнения Пуассона с граничными условиями Дирихле по оси y и периодическими условиями по оси x:

а). $$\begin{cases} \Delta u = 4 - 29\sin(2x + 5y), & 0 < x < \pi, \quad -1 < y < 1, \\ u(x + \pi,\, y) = u(x,\, y), & \\ u(x,\, -1) = 2 + \sin(2x - 5), & u(x,\, 1) = 2 + \sin(2x + 5). \end{cases}$$

Точное решение: $u_0(x,\, y) = \sin(2x + 5y) + 2y^2$.

б). Имеется бесконечная пластина толщиной H, т.е. $y \in [0,\, H]$. В пластине случайным образом расположено большое количество точечных источников тепла с постоянной мощностью. Для охлаждения пластину помещают в теплоемкую среду, которая охлаждает поверхность пластины до температуры T_0. Построить стационарное распределение тепла в пластине. Задача моделируется с периодическими граничными условиями по оси x.

Задача 2.15. Решить задачу Неймана для экранированного уравнения Пуассона в прямоугольнике:

а). $$\begin{cases} \Delta u - 4u = 12\big(1 - 2y^2\big), & -1 < x < 1, \quad -1 < y < 1, \\ u_x(-1,\, y) = 2\sinh 2, & u_x(1,\, y) = -2\sinh 2, \\ u_y(x,\, -1) = 2\cosh 2 - 12, & u_y(x,\, 1) = 2\cosh 2 + 12. \end{cases}$$

Точное решение: $u_0(x,\, y) = -\cosh 2x + \sinh 2y + 6y^2$.

б). На прямоугольный пружинный матрас кладут тонкое тяжелое кольцо массой M. Пружины матраса распределены равномерно и имеют жесткость k. Построить поверхность матраса.

Задача 2.16. Найти решение уравнения Пуассона в периодически замкнутой прямоугольной области:

а). $$\begin{cases} \Delta u = -32\cos 4x - 25\sin(4x+3y), \quad 0 < x < \pi,\ 0 < y < 2\pi, \\ u(x+\pi,\, y) = u(x,\, y), \qquad u(x,\, y+2\pi) = u(x,\, y). \end{cases}$$

Точное решение: $u_0(x,\, y) = 2\cos 4x + \sin(4x+3y)$.

б). В узлах квадратной решетки расположены в шахматном порядке положительно и отрицательно заряженные ионы. Ионы имеют одинаковый по модулю заряд, т.е. решетка является электронейтральной. Построить линии напряженности поля, которое создает решетка.

Задача 2.17. Найти решение уравнения Пуассона с условиями Дирихле и Неймана на границах:

а). $$\begin{cases} \Delta u = \left[6 - 72\left((1-3x)^2 + 4(1-2y)^2\right)\theta(x,\, y)\right]\left(1 - \theta^2(x,\, y)\right), \\ \theta(x,\, y) = \tanh\left(3(1-2y)^2 - (1-3x)^2\right), \quad 0 < x < 1,\ 0 < y < 1, \\ u_x(0,\, y) = 6\left(1 - \tanh^2\left(3(1-2y)^2 - 1\right)\right), \\ u_x(1,\, y) = -12\left(1 - \tanh^2\left(3(1-2y)^2 - 4\right)\right), \\ u(x,\, 0) = u(x,\, 1) = \tanh\left(3 - (1-3x)^2\right). \end{cases}$$

Точное решение: $u_0(x,\, y) = \theta(x,\, y)$.

б). Имеется прямоугольный блок. Левую и правую поверхности блока нагревают постоянным тепловым потоком q. Верхняя и нижняя поверхности блока поддерживаются при температурах T_1 и T_2. Построить стационарное распределение температуры в блоке.

Задача 2.18. Найти решение экранированного уравнения Пуассона в квадратной периодически замкнутой области:

а). $$\begin{cases} \Delta u - u = -f(x,\, y), \qquad 0 < x < \pi, \quad 0 < y < \pi, \\ u(x+\pi,\, y) = u(x,\, y), \qquad u(x,\, y+\pi) = u(x,\, y); \end{cases}$$

$$f(x,\, y) = e^{-\cos 2x}\,\sin 2y\,(2\cos 2x - 1)^2.$$

Точное решение: $u_0(x,\, y) = e^{-\cos 2x}\,\sin 2y$.

б). В узлах квадратной решетки расположены в шахматном порядке положительно и отрицательно заряженные ионы с зарядами q_1 и q_2. Предпо-

лагаем, что решетка создает электростатическое поле, которое удовлетворяет экранированному уравнению Пуассона. Построить линии напряженности поля.

П.3. Параболические уравнения

В постановках для уравнения Шрёдингера встречается понятие волнового пакета. *Волновой пакет* — совокупность волн с разными частотами, которые описывают обладающую волновыми свойствами формацию, в общем случае ограниченную во времени и пространстве.

Пусть α — угол, под которым распространяется волновой пакет, единичный вектор $\boldsymbol{n} = (\cos\alpha, \sin\alpha)$ — направление распространения. Обозначим волновое число как $\boldsymbol{k} = k \cdot \boldsymbol{n}$, где k — модуль волнового числа. Обозначим как $\boldsymbol{r}$ радиус-вектор, а как $\boldsymbol{r}_0$ — точку, на которую приходится максимум плотности вероятности в начальный момент времени. Амплитуду волнового пакета в начальный момент времени будем описывать функцией $a(r;\ \varepsilon)$.

В качестве амплитуды $a(r;\ \varepsilon)$ можно выбрать произвольную гладкую куполообразную функцию, которая имеет максимум в нуле и обращается в нуль при значениях $|r| \geq \varepsilon$, к примеру:

$$a(r;\ \varepsilon) = \eta(\varepsilon - |r|) \cdot \cos^3\left(\frac{\pi r}{2\varepsilon}\right), \quad r \in \mathbb{R}, \tag{П.3.1}$$

где $\eta(x)$ — функция Хевисайда.

Локализованный волновой пакет:

$$\psi(\boldsymbol{r}) = a(|\boldsymbol{r} - \boldsymbol{r}_0|;\ \varepsilon) \cdot e^{i(\boldsymbol{k}, \boldsymbol{r})}. \tag{П.3.2}$$

Плоский волновой пакет:

$$\psi(\boldsymbol{r}) = a((\boldsymbol{r} - \boldsymbol{r}_0, \boldsymbol{n});\ \varepsilon) \cdot e^{i(\boldsymbol{k}, \boldsymbol{r})}. \tag{П.3.3}$$

Волновой пакет, вообще говоря, состоит из суммы множества плоских волн с различными волновыми числами. При движении волновой пакет неизбежно будет распадаться, поскольку слагаемые с разными волновыми числами имеют различные энергии и скорости распространения. Волновое число свободной частицы связано с ее энергией по формуле $E = \frac{\hbar^2 k^2}{2m}$. Эта формула позволяет сделать оценки высоты потенциального барьера, который сможет преодолеть волновой пакет.

Задача 3.1. Прямоугольная область разделена посередине потенциальным барьером с потенциалом U_0, смоделировать прохождение локализованного волнового пакета (П.3.2) через потенциальный барьер.

Задача 3.2. Имеется бесконечная (периодическое замыкание вдоль x) металлическая пластина $0 < y < H$. Круглая область в центре пластины $r_0 = 0,2H$ имеет теплоемкость c в несколько раз выше окружающей. Верхняя граница пластины теплоизолирована, а нижнюю границу нагревают точечным тепловым потоком q в точке x_0. Смоделировать распространение тепла.

Задача 3.3. Прямоугольную камеру заполняют газом, который содержит радиоактивный изотоп с коэффициентом распада β. Газ поступает в камеру через 12 форсунок, расположенных по окружности. Мощность всех форсунок одинакова и постоянна. Концентрация изотопа на границе камеры поддерживается равной нулю. Построить распределение концентрации изотопа в камере в различные моменты времени, если в начальный момент времени концентрация равна нулю.

Задача 3.4. Прямоугольная комната обогревается системой «теплый пол». Предполагаем, что стены комнаты хорошо теплоизолированы, а теплый пол равномерно прогревает комнату по всей площади. Когда средняя температура в комнате достигает отметки $T_{\max}$, хозяин открывает форточку для проветривания. С освежающим зимним воздухом тепло из комнаты уходит с постоянным потоком тепла q. Когда средняя температура в комнате опускается до $T_{\min}$, хозяин закрывает форточку. Рассчитать распределение тепла в комнате при таком режиме проветривания, если начальная температура равна $T_{\min}$.

Задача 3.5. В прямоугольной периодически замкнутой (по оси y) области расположен круглый потенциальный барьер $U(r) = U_0\eta(r_0 - r)$. Смоделировать прохождение плоского волнового пакета (П.3.3) через барьер, волновой пакет движется вдоль оси x.

Задача 3.6. Прямоугольную камеру заполняют газом, который содержит радиоактивный изотоп с коэффициентом распада β. Газ поступает в камеру через 12 форсунок, расположенных по три штуки на каждой границе камеры. Мощность всех форсунок одинакова и постоянна. Построить распределение концентрации изотопа в камере в различные моменты времени, если в начальный момент времени концентрация изотопа равна нулю.

Задача 3.7. Локализованный волновой пакет (П.3.2) распространяется под углом в периодически замкнутой квадратной области.

Задача 3.8. Имеется прямоугольный блок $|x| < l_x$, $0 < y < l_y$. Основной материал блока имеет теплоемкость c_0, в то время как небольшой внутренний слой $|x| < H$ обладает высокой теплоемкостью $c_1 \gg c_0$. Плотность материала блока ρ и теплопроводность κ постоянны. Слева и справа от блока отводят тепло, поэтому температура на боковых поверхностях равна нулю. Нижнюю поверхность блока подогревают постоянным тепловым потоком, а верхняя поверхность блока теплоизолирована. Смоделировать распространение тепла в блоке.

Задача 3.9. Имеется бесконечная (периодическое замыкание) композитная пластина (рис. П.3.1). Основной материал пластины (матрица) имеет теплоемкость c_0, дополнительный материал (наполнитель) в форме небольших гранул имеет очень высокую теплоемкость $c_1 \gg c_0$. Плотность ρ и теплопроводность κ двух материалов совпадают. Начальная температура

пластины однородна и равна T_0. Затем пластина охлаждается при контакте со средой с нулевой температурой, теплообмен со средой происходит по закону Ньютона. Смоделировать процесс охлаждения пластины.

Задача 3.10. Численно смоделировать рассеяние локализованного волнового пакета (П.3.2) на клиновидном потенциальном барьере. Пример потенциального барьера, клин с вершиной в точке (x_0, y_0):

$$U(x, y) = U_0\,\eta(x - x_0)\eta(3(x_0 - x) + (y_0 - y)).$$

Задача 3.11. Имеется бесконечная (периодическое замыкание вдоль y) композитная пластина (см. рис. П.3.1). Основной материал пластины (матрица) имеет теплоемкость c_0, дополнительный материал (наполнитель) в форме небольших гранул имеет очень высокую теплоемкость $c_1 \gg c_0$. Плотность ρ и теплопроводность κ двух материалов совпадают. Левая граница пластины теплоизолирована, а правую границу пластины нагревают постоянным тепловым потоком, равномерно распределенным по границе. Смоделировать распространение тепла в пластине.

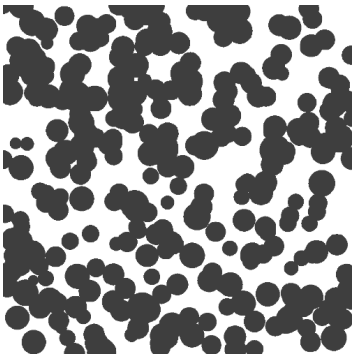

Рис. П.3.1. Композит

Задача 3.12. Прямоугольную камеру заполняют газом, который содержит радиоактивный изотоп с коэффициентом распада β. Газ поступает в камеру с постоянной скоростью через форсунку, которая перемещается по камере по траектории:

$$x(t) = 16a\sin^3 t, \quad y(t) = a(13\cos t - 5\cos 2t - 2\cos 3t - \cos 4t).$$

На границах камеры поддерживается нулевая концентрация изотопа. Построить распределение концентрации изотопа в камере в различные моменты времени. В начальный момент времени концентрация изотопа равна нулю.

Задача 3.13. Прямоугольная комната обогревается печкой с постоянной мощностью Q, которая расположена в центре комнаты. Система «умный дом» связывает работу обогревателя с показаниями термометра на стене комнаты. Обогреватель включается, когда температура на термометре опускается до значения $T_{\min}$ и снова выключается при достижении температуры $T_{\max}$. Теплопотери в комнате вызваны двумя «холодными» стенами, которые являются внешними стенами дома, они и обеспечивают постоянный отток тепла q из комнаты. Рассчитать распределение тепла в комнате при таком режиме работы обогревателя, если начальная температура в комнате равна $T_{\min}$.

Задача 3.14. В прямоугольной периодически замкнутой (по оси x) области расположен потенциальный барьер с щелью $U(x) = U_0\eta(|x| - \varepsilon)$. Смоделировать прохождение плоского волнового пакета (П.3.3) через щель, волновой пакет движется вдоль оси y.

Задача 3.15. Прямоугольную камеру заполняют газом, который содержит радиоактивный изотоп с коэффициентом распада β. Газ поступает в камеру с постоянной скоростью через форсунку, которая перемещается по камере по траектории:

$$x(t) = \frac{a\cos\omega t}{1+\sin^2\omega t}, \qquad y(t) = \frac{a\sin\omega t\cos\omega t}{1+\sin^2\omega t}.$$

Границы камеры изолированы. Построить распределение концентрации изотопа в камере в различные моменты времени, если в начальный момент времени концентрация изотопа равна нулю.

Задача 3.16. В центре квадратной периодически замкнутой области расположен гладкий куполообразный потенциальный барьер. Смоделировать рассеяние плоского волнового пакета (П.3.3) на данном потенциале. В качестве потенциала можно выбрать знакомую функцию (П.3.1):

$$U(x,\, y) = U_0 \cdot a(\sqrt{x^2+y^2};\, \varepsilon).$$

Задача 3.17. Имеется квадратный блок $|x| < H$, $|y| < H$. Основной материал блока имеет теплоемкость c_0, в то время как круглая область в центре $r < r_0$ обладает высокой теплоемкостью $c_1 \gg c_0$. Плотность материала блока ρ и теплопроводность κ постоянны. Левая и правая границы блока теплоизолированы, верхняя поверхность поддерживается при нулевой температуре. Нижняя поверхность блока нагревается по линейному закону до некоторого предельного значения T_0, т.е. $T_B(t) = \min(\alpha t,\, T_0)$. Смоделировать распространение тепла в блоке, если начальная температура равна нулю.

Задача 3.18. Имеется бесконечная (периодическое замыкание) композитная пластина (см. рис. П.3.1). Основной материал пластины (матрица) имеет теплоемкость c_0, дополнительный материал (наполнитель) в форме небольших гранул имеет очень высокую теплоемкость $c_1 \gg c_0$. Плотность ρ и теплопроводность κ двух материалов совпадают. Пластина по всей поверхности контактирует со средой, теплообмен с которой происходит по закону Ньютона. В начальный момент времени пластина и среда находятся в равновесии при нулевой температуре, затем среда начинает нагреваться по линейному закону $T(t) = \alpha t$. Смоделировать процесс нагревания пластины.