СОДЕРЖАНИЕ

1. ЭЛЕМЕНТЫ ТЕОРИИ ПОГРЕШНОСТЕЙ

Одно из основных понятий вычислительной математики – понятие погрешности. ***Погрешностью*** называют отклонение приближенного решения от истинного. Обычно погрешность оценивают при решении каких-либо задач приближенными методами. Целью изучения погрешностей решения задачи, естественно, является их уменьшение. Поэтому все погрешности процесса решения задачи сначала разделяют на два больших класса:

1) неустранимая погрешность;
2) устранимая погрешность.

Неустранимая погрешность обусловлена неточностью ***исходных данных*** (например, неточностью знания физических констант или других параметров метода решения) и никак не может быть уменьшена в процессе решения. Это означает, что борьба с неустранимой погрешностью с точки зрения программиста невозможна, для её уменьшения требуется разработка принципиально новых методов подготовки исходных данных (например, использование более точных измерительных приборов уменьшает так называемую ***инструментальную ошибку***). Однако разработчикам методов решения задач все же следует выяснять неустранимую погрешность решения задачи, поскольку эта погрешность тоже влияет на конечный результат, и требовать от метода решения точности выше, чем дает возможность получить неустранимая погрешность, бессмысленно. Кроме того, наличие неустранимой погрешности требует изучения распространения погрешности в процессе вычислений для разрабатываемых методов.

Устранимая погрешность – это погрешность, связанная с самим процессом решения задачи, т.е. как раз та погрешность, которая может быть уменьшена выбором более точных методов решения. Как правило, выделяют несколько типов устранимой погрешности.

1. Погрешность, связанная с математической постановкой задачи. Математическая формулировка (чаще называемая ***математической моделью***) редко точно отображает реальные явления: обычно она дает лишь более или менее идеализированную модель. Как правило, при по-

строении такой модели используются некоторые упрощения, что вызывает появление погрешности. Иногда бывает и так, что решить задачу в точной постановке трудно или даже невозможно. Тогда ее заменяют близкой по результатам приближенной задачей. При этом также возникает погрешность. Погрешность этого типа обычно называют ***погрешностью модели***.

2. Погрешность, связанная с использованием в математической модели бесконечных процессов математического анализа. Функции, фигурирующие в математических формулах, часто задаются в виде бесконечных последовательностей или рядов. Так как бесконечный процесс не может быть завершен в конечное число шагов, то мы вынуждены остановиться на некотором члене последовательности, считая его приближением к искомому решению. Понятно, что такой обрыв процесса вызывает погрешность, называемую обычно ***остаточной погрешностью*** или ***погрешностью аппроксимации***.

3. Погрешность, связанная с использованием конечной арифметики. При переводе даже рациональных чисел в любую позиционную систему счисления (при компьютерных вычислениях чаще всего в двоичную) результат может содержать бесконечное число цифр (например, может получиться периодическая дробь). В вычислениях же используется, как правило, конечное число знаков. Так возникает ***погрешность округления***.

4. Погрешность, связанная с действиями над приближенными числами (***накапливаемая погрешность***). В процессе вычислений с приближенными числами погрешности исходных данных в какой-то мере переносятся в результат вычислений. В зависимости, например, от порядка действий накопление погрешности может идти по-разному. В этом смысле накапливаемая погрешность является устранимой, хотя обычная её причина – неустранимая погрешность в исходных данных. Часто накапливаемую погрешность и погрешность округления отдельно не рассматривают, а объединяют и называют ***вычислительная погрешность.***

Заметим, что при решении конкретной задачи некоторые виды погрешности могут отсутствовать или их влияние может быть несущественным. Ниже мы рассмотрим основные определения теории приближенных вычислений и некоторые наиболее часто применимые методы оценки накопления погрешности при вычислениях с конечной арифметикой.

1.1. Абсолютная и относительная погрешности

Обозначим через X^* ***точное значение*** некоторой величины. Число X, незначительно отличающееся от X^* и заменяющее последнее в вычислениях, называется ***приближенным значением числа*** X^***.** Если при этом $X < X^*$ ($X > X^*$), то X называется ***приближенным значением числа*** X^* ***по недостатку*** **(*по избытку*)**. Величина $X - X^*$ называется ***погрешностью***, а ее абсолютное значение $\Delta X = \left|X^* - X\right|$ – ***абсолютной погрешностью***.

Очевидно, что сама абсолютная погрешность $\left|X^* - X\right|$ представляет чисто теоретический интерес, так как без знания X^* её нельзя вычислить. Но практически всегда можно указать границы, в которых изменяется абсолютная погрешность. Эти границы определяются тем способом, которым было получено приближенное число. Например, если для измерений длины используется обычная ученическая линейка, цена деления которой 1 мм, то модуль абсолютной погрешности измерения не будет превышать 0.5 мм. Поэтому чаще всего используется не сама абсолютная погрешность, а *некоторая* ее оценка сверху $\Delta_X \geq \left|X^* - X\right|$ – ***предельная абсолютная погрешность***.

Таким образом, если Δ_X – предельная абсолютная погрешность приближенного числа X, заменяющего точное X^*, то точное значение X^* заключено в границах

$$X - \Delta_X \leq X^* \leq X + \Delta_X \tag{1}$$

Для краткости выражение (1) часто записывают в виде $X^* = X \pm \Delta_X$.

Очевидно, что если Δ_X – предельная абсолютная погрешность, то $\forall c > 0$ величина $\Delta_X + c$ также является предельной абсолютной погрешностью, поэтому целесообразно в качестве Δ_X выбирать как можно меньшее значение.

Абсолютная погрешность и предельная абсолютная погрешность еще не характеризуют точность результата, как ее обычно интуитивно понимают, если не указан сам результат. Например, пусть предельная

абсолютная погрешность результата измерения равна 1 см. Если при этом измерялась длина комнаты, то точность удовлетворительная. Если же измерялось расстояние между двумя пунктами различных городов, то точность очень велика. Поэтому используют еще одну характеристику точности – ***относительную погрешность*** $\delta X = \dfrac{X^* - X}{|X|}$.

Иногда относительную погрешность определяют как отношение абсолютной ошибки к точному значению, но поскольку значение обычно неизвестно, такое определение непрактично. Более того, если ошибка мала, то разница в определениях не скажется существенно на численной величине относительной ошибки.

Так же как и для абсолютной погрешности, вводится понятие ***предельной относительной погрешности***

$$\delta_x \geq \frac{X - X^*}{|X|},$$

откуда, учитывая определение предельной абсолютной погрешности, получим

$$\delta_x = \frac{\Delta_X}{|X|}.$$

Обратим внимание, что для величин, близких по значению к единице, абсолютная и относительная ошибки практически одинаковы. Для очень больших или очень малых величин относительная и абсолютная ошибки представляются совершенно разными числами. В том случае, когда точное значение величины – нуль, понятие относительной погрешности обычно не определяют.

Заметим, что в отличие от абсолютной погрешности, которая чаще всего бывает размерной величиной, относительная погрешность – величина безразмерная. При этом достаточно часто относительную погрешность определяют в процентах.

В дальнейшем ***предельные абсолютные*** и ***относительные погрешности***, если не будет опасности смешения терминов, будем называть просто ***абсолютными*** и ***относительными погрешностями***.

1.2. Представление вещественных чисел

Любое число A может быть представлено в виде

$$A = m \cdot q^p,$$

где p – некоторое целое число, называемое ***порядком*** числа A, q – основание системы счисления, m – число, называемое ***мантиссой*** числа A. Представление числа A называется:

– ***нормальным***, если $q^{-1} \le |m| \le 1$;

– ***нормализованным***, если $1 \le |m| \le q$.

Значащими цифрами числа называются все цифры в его изображении, отличные от нуля, и нули, если они содержатся между значащими цифрами или расположены в конце числа и указывают на сохранение разряда точности. Нули, стоящие левее первой отличной от нуля цифры, *не являются* значащими цифрами.

Например, в числе 0,002 080 первые три нуля не являются значащими цифрами, так как они служат только для установления десятичных разрядов других цифр. Остальные два нуля являются значащими цифрами, так как первый из них находится между значащими цифрами 2 и 8, а второй, как это отражено в записи, указывает, что в приближенном числе сохранен десятичный разряд 10^{-6}. В случае, если в данном числе 0,002 080 последняя цифра не является значащей, то это число должно быть записано в виде 0,002 08. С этой точки зрения числа 0,002 080 и 0,002 08 неравноценны, так как первое из них содержит четыре значащие цифры, а второе – лишь три значащие цифры.

При написании больших чисел нули справа могут служить как для обозначения значащих цифр, так и для определения разрядов остальных цифр. Поэтому при обычной записи чисел могут возникнуть неясности. Например, рассматривая число 689 000, мы не имеем возможности по его виду судить о том, сколько в нем значащих цифр, хотя можно утверждать, что их не меньше трех. Этой неопределенности можно избежать, записав число в виде $0,689 \cdot 10^6$, если оно имеет три значащие цифры; или $0,68900 \cdot 10^6$, если число имеет пять значащих цифр, и т.п.

Вообще, нормализованная запись удобна также и для чисел, содержащих большое количество незначащих первых нулей, например $0,000000120 = 0,120 \cdot 10^{-6}$ и т.п.

Говорят, что n первых значащих цифр приближенного числа являются ***верными***, если абсолютная погрешность этого числа не превышает половины единицы разряда, выражаемого n-й значащей цифрой, считая слева направо.

Например, для точного числа $X^* = 35,97$ число $X = 36,00$ является приближением с тремя верными знаками, так как

$$\left|X^* - X\right| = 0,03 < \frac{1}{2} \cdot 10^{-1}.$$

Термин «n верных знаков» не следует понимать буквально, т.е. так, что в данном приближенном числе X, имеющем n верных знаков, n первых значащих цифр его совпадают с соответствующими цифрами точного числа. Например, приближенное число $X = 9,996$, заменяющее точное $X^* = 10$, имеет три верных знака, причем все цифры этих чисел различны.

Обратим внимание, что если в последнем примере поменять местами значения, т.е. $X = 10$, $X^* = 9,996$, то число верных знаков станет 4. Из-за такой неоднозначности данное выше определение числа верных знаков в настоящее время используется редко. Часто под числом с n верными знаками понимают число, относительная погрешность которого не превышает 10^{-n+1}. Заметим, что лучше вообще не использовать для оценки погрешности понятие «число верных цифр», а вместо него рассматривать относительную погрешность (возможно, в процентах).

1.3. Округление чисел

Рассмотрим некоторое число X^*, записанное в десятичной нумерации. Часто бывает необходимо ***округлить*** это число, т.е. заменить его числом X с меньшим количеством значащих цифр. Число X выбирают так, чтобы погрешность округления $\left|X^* - X\right|$ была минимальной.

Правило округления (по дополнению). Чтобы округлить число до n значащих цифр, отбрасывают все его цифры, стоящие справа от n-й значащей цифры, или, если это нужно для сохранения разрядов, заменяют их нулями. При этом:

1) если первая из отброшенных цифр меньше 5, то оставшиеся десятичные знаки сохраняются без изменения;

2) если первая из отброшенных цифр больше 5, то к последней оставшейся цифре прибавляется единица;

3) если первая из отброшенных цифр равна 5 и среди остальных отброшенных цифр имеются ненулевые, то последняя оставшаяся цифра увеличивается на единицу;

4) если же первая из отброшенных цифр равна 5 и все остальные отброшенные цифры являются нулями, то последняя оставшаяся цифра сохраняется неизменной, если она четная, и увеличивается на единицу, если она нечетная (правило четной цифры).

1.4. Распространение ошибок

Один из наиболее важных вопросов в численном анализе – вопрос о том, как ошибка, возникшая в определенном месте в ходе вычислений (или в исходных данных), распространяется дальше, т.е. становится ли ее влияние больше или меньше по мере того, как производятся последующие операции [3].

Крайним случаем является вычитание двух почти равных чисел: даже при очень маленьких ошибках обоих этих чисел относительная ошибка разности может оказаться очень большой. Эта большая относительная ошибка будет распространяться дальше при выполнении всех последующих арифметических операций.

Влияние накопления погрешности на точность результата может быть достаточно заметным даже при не очень сложных вычислениях. Покажем это на примере.

Пусть требуется возвести в куб выражение $\frac{\sqrt{2}-1}{\sqrt{2}+1}$ (обратим внимание, что выражение положительное). Это возможно сделать многими способами, рассмотрим некоторые из них:

$$\left(\frac{\sqrt{2}-1}{\sqrt{2}+1}\right)^3 = \left(\sqrt{2}-1\right)^6 = \left(3-2\sqrt{2}\right)^3 = 99-70\sqrt{2}. \qquad (2)$$

Вычислим все 4 последние выражения, взяв за приближенное значение величины $\sqrt{2}$ числа $\frac{7}{5}=1.4$, $\frac{17}{12}=1.41666...$ и $\frac{707}{500}=1.414$. Так как $\sqrt{2}=1.4142135624...$, то все эти значения приближают точное. Результаты представлены в табл. 1.

Таблица 1

$\sqrt{2}$	$\left(\frac{\sqrt{2}-1}{\sqrt{2}+1}\right)^3$	$\left(\sqrt{2}-1\right)^6$	$\left(3-2\sqrt{2}\right)^3$	$99-70\sqrt{2}$
$\frac{7}{5}$	$\frac{1}{216}\approx$ $\approx 0.00\underline{4}6$	$\frac{64}{15625}\approx$ $\approx 0.00\underline{5}1$	$\frac{1}{125}=$ $=0.008$	1
$\frac{17}{12}$	$\frac{125}{24389}\approx$ $\approx 0.00\underline{51}3$	$\frac{15625}{2985354}\approx$ $\approx 0.00\underline{5}2$	$\frac{1}{216}\approx$ $\approx 0.00\underline{4}6$	$-\frac{1}{6}=$ $=-0.6(6)$
$\frac{707}{500}$	$\frac{8869743}{1758416743}\approx$ $\approx 0.00\underline{50}44$	$\frac{78672340886049}{15625\cdot 10^{12}}\approx$ $\approx 0.00\underline{50}4$	$\frac{636056}{125000000}\approx$ $\approx 0.00\underline{50}9$	0.02

Мы получили значительно отличающиеся друг от друга ответы, причем все очень неточные (верные знаки подчеркнуты), правильный ответ с точностью 5 значащих цифр 0.0050506. Данный пример показывает, что с приближенными вычислениями нужно обращаться очень аккуратно.

Вообще говоря, любое действие с приближенными числами можно рассматривать как функцию нескольких аргументов.

Пусть задана дифференцируемая функция $u = f\left(x_1, x_2, \ldots, x_n\right)$ и пусть Δx_i, $i = \overline{1,n}$ – погрешности аргументов функции. Тогда абсолютная погрешность функции определяется выражением

$$\left|\Delta u\right| = \left|f\left(x_1 + \Delta x_1, \ldots, x_n + \Delta x_n\right) - f\left(x_1, \ldots, x_n\right)\right|.$$

Обычно на практике Δx_i – это малые величины, произведениями, квадратами и высшими степенями которых можно пренебречь. Поэтому можно оценить эту погрешность через дифференциал:

$$\left|\Delta u\right| \approx \left|df\left(x_1, x_2, \ldots, x_n\right)\right| = \left|\sum_{i=1}^{n} \frac{\partial u}{\partial x_i} \Delta x_i\right| \le \sum_{i=1}^{n} \left|\frac{\partial u}{\partial x_i}\right| \left|\Delta x_i\right|. \qquad (3)$$

Обозначив через Δ_{x_i}, $i = \overline{1,n}$ предельные абсолютные погрешности аргументов x_i, а через Δ_u – предельную погрешность функции u, для малых Δ_{x_i} получим[1]:

$$\Delta_u = \sum_{i=1}^{n} \left|\frac{\partial u}{\partial x_i}\right| \Delta_{x_i}. \qquad (4)$$

Разделив обе части неравенства (3) на $\left|u\right|$, получим оценку для относительной погрешности u:

$$\left|\delta u\right| \le \sum_{i=1}^{n} \left|\frac{\partial u}{\partial x_i} \cdot \frac{1}{u}\right| \left|\Delta x_i\right| = \sum_{i=1}^{n} \left|\frac{\partial \ln u}{\partial x_i}\right| \left|\Delta x_i\right|,$$

следовательно,

$$\delta_u = \sum_{i=1}^{n} \left|\frac{\partial \ln u}{\partial x_i}\right| \left|\Delta x_i\right|. \qquad (5)$$

[1] Знак равенства в этой формуле можно поставить в соответствии с определением предельной погрешности, поскольку полученная величина действительно оценивает погрешность сверху, предполагая, что все погрешности аргументов одного знака.

Заметим, что основные арифметические действия можно также рассматривать как функции двух переменных. Поэтому если в формулах (4) и (5) в качестве функции u взять одну из функций двух аргументов: сумму, разность, произведение или частное, то можно легко получить формулы для ***предельных погрешностей основных арифметических действий***:

$$\Delta_{(X+Y)} = \Delta_X + \Delta_Y, \quad \delta_{(X+Y)} = \left|\frac{X}{X+Y}\right|\delta_X + \left|\frac{Y}{X+Y}\right|\delta_Y, \tag{6}$$

$$\Delta_{(X-Y)} = \Delta_X + \Delta_Y, \quad \delta_{(X-Y)} = \left|\frac{X}{X-Y}\right|\delta_X + \left|\frac{Y}{X-Y}\right|\delta_Y, \tag{7}$$

$$\Delta_{(X\cdot Y)} \approx |Y|\Delta_X + |X|\Delta_Y, \quad \delta_{(X\cdot Y)} = \delta_X + \delta_Y, \tag{8}$$

$$\Delta_{(X/Y)} \approx \left|\frac{1}{Y}\right|\Delta_X + \left|\frac{X}{Y^2}\right|\Delta_Y, \quad \delta_{(X/Y)} = \delta_X + \delta_Y. \tag{9}$$

Обратим внимание, что согласно формулам (6), (7), вычитание (сложение) близких по абсолютной величине чисел одного знака (разного знака) может приводить к очень существенному росту относительной погрешности.

Вернемся теперь к вычислениям, приведенным в табл. 1. Именно из-за вычитания близких чисел и возникает большая относительная погрешность. Как видно из таблицы, различные по последовательности действий (математически одинаковые) формулы расположились по возрастанию погрешности в следующем порядке:

$$\left(\frac{\sqrt{2}-1}{\sqrt{2}+1}\right)^3,\ \left(\sqrt{2}-1\right)^6,\ \left(3-2\sqrt{2}\right)^3,\ 99-70\sqrt{2}.$$

Покажем, что с помощью дифференциала (формул (4) и (5)) о таком расположении можно было бы узнать и до вычислений. Для этого вычислим относительные погрешности каждой из формул, выбрав в них в качестве параметра $x = \sqrt{2}$. Пусть

$$f_1 = \left(\frac{x-1}{x+1}\right)^3,\ f_2 = (x-1)^6,\ f_3 = (3-2x)^3,\ f_4 = 99-70x.$$

Тогда

$$\delta_{f_1} = \left|\frac{d}{dx}\ln\left(\frac{x-1}{x+1}\right)^3\right||\Delta x| = 3\left|\frac{1}{x-1}-\frac{1}{x+1}\right||\Delta x|,$$

$$\delta_{f_2} = \left|\frac{d}{dx}\ln(x-1)^6\right||\Delta x| = 6\left|\frac{1}{x-1}\right||\Delta x|,$$

$$\delta_{f_3} = \left|\frac{d}{dx}\ln(3-2x)^3\right||\Delta x| = 3\left|\frac{2}{3-2x}\right||\Delta x| = 6\left|\frac{1}{3-2x}\right||\Delta x|,$$

$$\delta_{f_4} = \left|\frac{d}{dx}\ln(99-70x)\right||\Delta x| = \left|\frac{70}{99-70x}\right||\Delta x|.$$

Подставляя теперь одно из значений x, например, $x=7/5$, получим:

$$\delta_{f_1} = 3\left|\frac{1}{x-1}-\frac{1}{x+1}\right||\Delta x| = 6.25|\Delta x|,$$

$$\delta_{f_2} = 6\left|\frac{1}{x-1}\right||\Delta x| = 15|\Delta x|,$$

$$\delta_{f_3} = 6\left|\frac{1}{3-2x}\right||\Delta x| = 30|\Delta x|,$$

$$\delta_{f_4} = \left|\frac{70}{99-70x}\right||\Delta x| = 70|\Delta x|.$$

Таким образом, оценки погрешностей также указывают на меньшую погрешность при вычислениях по первой формуле. Однако необходимо понимать, что полученные оценки являются оценками сверху и не всегда точными (например, можно сравнить полученные оценки для $x=7/5$ с реальными погрешностями, которые легко получить из табл. 1). Поэтому выбирать по этим оценкам наилучший способ вычислений разумно, но гарантировать, что при реальных вычислениях погрешности будут именно такими, нельзя (правда, они могут быть только меньше, чем полученные оценки).

Заметим, что ошибки округления в формулах (6)–(9) не учитываются, поэтому, если необходимо подсчитать, как распространяется погрешность в последующих арифметических операциях, то следует к вычисленной по одной из формул ошибке прибавить ошибку округления.

1.5. Общие рекомендации по повышению точности вычислений

Рассмотрим пример, показывающий, что повышение точности иногда может быть достигнуто за счет несложного алгебраического преобразования.

Пусть необходимо вычислить наименьший корень уравнения $y^2 - 140y + 1 = 0$. Для определенности будем считать, что вычисления производятся в десятичной системе счисления, причем в мантиссе числа после округлений удерживается 4 разряда, т.е. до округления

$$y = 70 - \sqrt{4899}\,,\ \sqrt{4899} = 69{,}992\ldots,$$

после округления

$$\sqrt{4899} \approx 69{,}99\,,\ y \approx 70 - 69{,}99 = 0{,}0\underline{1}\,.$$

То же самое значение y можно, «избавившись от иррациональности в числителе», представить в виде $y = \dfrac{1}{70 + \sqrt{4899}}$. Последовательно производя вычисления, получаем

$$\sqrt{4899} = 69{,}99\,,\ 70 + 69{,}99 = 139{,}99$$

и после округления $70 + 69{,}99 = 140{,}0$.

Наконец, $\dfrac{1}{140} = 0{,}00714285...$, и после округления $y \approx 0{,}00\underline{7143}$. Производя вычисления с дополнительными разрядами, можно проверить, что в обоих случаях все подчеркнутые цифры результатов верные, однако во втором случае точность результата существенно выше. Дело в том, что в первом случае пришлось вычитать близкие большие числа; так как эти числа были большие, то они были округлены с большой абсолютной погрешностью, в результате и ответ получился с большой абсолютной погрешностью. Это явление называется ***потеря значащих цифр***, оно имеет место при вычитании близких величин.

Рассмотрим другой типичный пример, где порядок выполнения операций влияет на погрешность результата. На компьютере с плавающей точкой вычисляется значение суммы

$$S_{1000000} = \sum_{j=1}^{1000000} \frac{1}{j^2}.$$

Можно вычислить $S_{1000000}$ либо в цикле по возрастанию, которому соответствует рекуррентная формула

$$S_n = S_{n-1} + \frac{1}{n^2},\ n = 1,...,1000000,\ S_0 = 0.$$

либо в цикле по убыванию, т.е. по рекуррентной формуле

$$\sum_{n-1} = \sum_{n} + \frac{1}{n^2},\ n = 1000000,...,1,\ \sum_{0} = S_{1000000},\ \sum_{1000000} = 0.$$

Оказывается, во втором случае суммарная вычислительная погрешность существенно меньше.

Дело заключается в том, что в современных компьютерах сложение чисел осуществляется по следующей схеме. Два числа a и b складываются с большим числом значащих цифр, а затем происходит отбрасывание последних знаков и округление результата с тем, чтобы осталось t или $t-1$ значащих цифр. В результате получается приближенное значение суммы $a+b$ с погрешностью, не превосходящей $2^{-t}|a+b|$, но в неблагоприятном случае большей, чем $2^{-1-t}|a+b|$. В первом случае при каждом сложении значение суммы больше 1 и в принципе возможно получение погрешности около $10^6 \cdot 2^{-t}$. Во втором случае

$$\sum_{n} = O\left(\frac{1}{n}\right),$$

поэтому погрешность накапливается существенно медленнее. Можно показать, что погрешность окончательного результата $100 \cdot 2^{-t}$.

На практике было проведено вычисление по обоим алгоритмам и оказалось, что для первого алгоритма погрешность $\approx 2 \cdot 10^{-4}$, а для второго – $\approx 6 \cdot 10^{-8}$.

Чтобы избежать подобных проблем, можно воспользоваться следующими несложными рекомендациями:

1) если необходимо произвести сложение-вычитание длинной последовательности чисел, то начинать надо с наименьших чисел;

2) желательно избавиться от вычитания двух почти равных чисел, по возможности преобразуя формулы;

3) необходимо сводить к минимуму число используемых арифметических операций (эта рекомендация вообще-то нужна и для ускорения работы программ);

4) если язык программирования и компьютер позволяют использовать вещественные числа разных типов (с разным количеством хранимых разрядов числа), то увеличение числа разрядов всегда повышает точность вычислений (правда, в ущерб памяти).

1.6. Правила подсчета верных цифр (по В.М. Брадису)

В предыдущих пунктах мы рассмотрели способы оценки предельной абсолютной погрешности арифметических действий. При этом предполагалось, что абсолютные погрешности аргументов усиливают друг друга, а это практически бывает сравнительно редко. Для оценки же погрешности вычислений при небольшом числе действий (например, при ручных вычислениях) можно воспользоваться следующими эмпирическими правилами. Эти правила даются в предположении, что аргументы действий содержат только верные цифры и число действий невелико.

1) При сложении и вычитании приближенных чисел в результате следует сохранить столько десятичных знаков, сколько их в приближенном данном с наименьшим числом десятичных знаков после запятой.

2) При умножении и делении в результате следует сохранить столько значащих цифр, сколько их в приближенном данном с наименьшим числом верных значащих цифр.

3) При возведении приближенного числа в квадрат или куб в результате следует сохранить столько значащих цифр, сколько их в основании степени.

4) При извлечении квадратного и кубического корней из приближенного числа в результате следует сохранить столько значащих цифр, сколько их в подкоренном числе.

5) При вычислении промежуточных результатов следует сохранить на одну цифру больше, чем рекомендуют правила 1–4. В окончательном результате эта «запасная цифра» отбрасывается.

6) Если данные можно брать с произвольной точностью, то для получения результата с m верными цифрами исходные данные следует брать с таким числом цифр, которые согласно предыдущим правилам обеспечивают m+1 цифру в результате.

Еще раз обратим внимание, что сформулированные выше правила подсчета цифр именно эмпирические: они описывают наиболее вероятные ситуации, хотя существуют примеры, не удовлетворяющие этим правилам. Следовательно, вычисления способом подсчета цифр – самый грубый способ оценки погрешности результатов действий. Однако он очень прост и удобен, а точность таких вычислений вполне достаточна для большинства технических расчетов. Поэтому данный способ широко распространен, например, в практике измерений.

2. ПРЕДСТАВЛЕНИЕ ВЕЩЕСТВЕННЫХ ЧИСЕЛ В ЭВМ

При программировании вычислительных задач часто необходимо знать особенности представления вещественных чисел в компьютере. Это связано с тем, что в отличие от целых чисел вещественные числа (например, иррациональные) не всегда могут быть представлены точно, и поэтому при их использовании требуется учесть погрешность округления и правильно организовать вычисления в исключительных ситуациях. Представление вещественных чисел в большинстве современных компьютеров соответствует принятому в 2008 году стандарту IEEE 754. Стандарт IEEE 754-2008 регламентирует:

- определения двоичных форматов хранения мантиссы, показателя и знака, форматы положительного и отрицательного нуля, плюс и минус бесконечностей, а также определение «не числа» (NaN);
- методы, которые используются для преобразования числа в процессе математических операций;
- обработку исключительных ситуаций таких, как деление на нуль, переполнение, потеря значимости, работу с денормализованными числами и т.д.;
- арифметические и другие операции с вещественными числами в различных форматах.

2.1. Общие принципы побитного представления вещественного числа

Как уже было сказано выше, любое число A может быть представлено в виде: $A = m \cdot q^p$, где p – порядок числа A, q – основание системы счисления, m – мантисса числа A. Стандарт определяет, что в двоичном представлении вещественного числа выделяется три группы бит. Первая группа состоит из одного бита s и определяет знак числа, 0 – положительный, 1 – отрицательный. Вторая группа содержит N_1 бит и предназначена для хранения *информации о порядке* числа для основания $q = 2$. Третья группа содержит N_2 бит и предназначена для хранения *информации о мантиссе* числа.

В качестве информации о порядке хранится так называемая ***характеристика*** числа – $G = p + g$, где g – сдвиг порядка. Сдвиг нужен для того, чтобы можно было хранить как положительные, так и отрицательные значения порядка. Значение сдвига порядка определяется числом бит N_1: $g = 2^{N_1 - 1} - 1$.

Информация о мантиссе хранится следующим образом. Число представляется в нормализованном виде так, что $1 \le |m| < 2$. В этом случае, очевидно, первый бит мантиссы всегда равен 1 и поэтому его можно не хранить (этот бит обычно называют ***скрытым***). Поэтому в N_2 битах фактически хранится натуральное число M, содержащее информацию о $N_2 + 1$ значащих цифрах мантиссы m (в двоичной системе счисления).

Таким образом, для восстановления числа A по его бинарному представлению можно использовать формулу

$$A = (-1)^s \left(1 + M \cdot 2^{-N_2}\right) \cdot 2^{G-g}. \qquad (10)$$

Заметим, что стандарт определяет несколько исключений из описанного способа хранения и формулы (10).

Первое исключение касается представления нуля, поскольку по формуле (10) значение нуль получить невозможно. Для представления нуля выделяется сразу два кода (так называемые положительный и отрицательный нули): все биты в представлении числа равны нулю (***положительный нуль***), и все биты, кроме первого, равны нулю (***отрицательный нуль***). Обратим внимание, что при сравнении чисел положительный нуль равен отрицательному.

Второе исключение касается нормализации. Поскольку в вычислительных задачах работа с числами малых порядков обычно встречается чаще, стандарт расширяет диапазон представления малых (по модулю) чисел за счет ***денормализации.*** Признаком денормализации является равенство нулю всех бит характеристики числа. Для денормализованного числа не используется скрытый бит и соответственно мантисса полностью определяется значением M, т.е. вместо формулы (10) используется формула

$$A = (-1)^s \left(M \cdot 2^{-N_2}\right) \cdot 2^{1 - 2^{N_1 - 1}}. \qquad (11)$$

Ситуация, когда результат операции настолько мал, что непредставим даже в виде денормализованного числа, называется ***исчезновением порядка***. Результатом исчезновения порядка обычно является нуль (знак которого определяется знаком операндов и смыслом операции, вызвавшей исчезновение порядка).

Третье исключение определяет специальные коды, которые могут быть использованы в исключительных ситуациях. Признаком специального кода является равенство единице всех N_1 бит характеристики числа. Стандарт определяет два специальных кода для ***положительной*** и ***отрицательной бесконечности***. Знак бесконечности определяется знаковым битом, а признаком бесконечности является равенство нулю всех N_2 бит мантиссы. Бесконечность может получаться в результате арифметических действий или функций, если результат действия (или функции) вызывает так называемое ***переполнение*** – когда результат превышает (по модулю) максимально представимое число. Например, бесконечностью будет результат деления единицы на денормализованное число. Действия с бесконечностью совершаются по правилам, аналогичным предельным соотношениям математического анализа. Например, умножение бесконечности на число – бесконечность, сложение бесконечности с числом – бесконечность, деление на нуль – бесконечность, деление числа на бесконечность – нуль и т.д.

Все остальные специальные коды определяются как ***не число*** (***NaN – Not Arithmetic Number***). Эти коды могут использоваться процессором как результаты запрещенных операций.

К операциям, приводящим к появлению *NaN* в качестве ответа, относятся:

- все математические операции, содержащие *NaN* в качестве одного из операндов;
- деление нуля на нуль;
- деление бесконечности на бесконечность;
- умножение нуля на бесконечность;
- сложение бесконечностей разных знаков;
- вычисление квадратного корня отрицательного числа;
- логарифмирование отрицательного числа.

Поясним теперь, зачем в описанном представлении чисел с плавающей запятой существует два нуля, которые отличаются только знаком. Это сделано для того, чтобы выражения, которые в результате переполнения или потери значимости превращаются в бесконечность или в нуль, при умножении и делении все же могли представить максимально корректный результат. Например, если бы у нуля не было знака, выражение $\frac{1}{1/x}$ было бы не равно x при $x=\pm\infty$, так как нельзя было бы различить $\frac{1}{+\infty}$ и $\frac{1}{-\infty}$.

2.2. Число с плавающей точкой одинарной точности

Число одинарной точности по стандарту IEEE 754-2008 занимает в памяти 32 бита (4 байта). Для характеристики отводится $N_1=8$ бит, для мантиссы $N_2=23$. Сдвиг порядка $g=2^{8-1}-1=127$. Формула (10) принимает вид

$$A=(-1)^s\left(1+M\cdot 2^{-23}\right)\cdot 2^{G-127}, \tag{12}$$

а формула (11) –

$$A=(-1)^s M\cdot 2^{-149}. \tag{13}$$

Самое большое представимое в этом формате число имеет, очевидно, код

```
0 11111110 11111111111111111111111
```

(максимально возможную характеристику и мантиссу). В десятичной системе счисления согласно формуле (12) это число

$$\left(2-2^{-23}\right)\cdot 2^{127}\approx 2^{128}\approx 3.4\cdot 10^{38}.$$

Самое маленькое положительное число в нормализованном виде имеет код

```
0 00000001 00000000000000000000000.
```

В десятичной системе счисления согласно формуле (12) это число

$$(1)\cdot 2^{-126}\approx 1.175\cdot 10^{-38}.$$

Заметим, что реально представимое минимальное положительное число является денормализованным и его код равен

```
0 00000000 00000000000000000000001,
```

т.е. в десятичной системе счисления согласно формуле (13) его значение равно $2^{-149}\approx 1.4\cdot 10^{-45}$.

Приведем еще несколько примеров в виде табл.2:

Таблица 2

Число	Разложение по степеням двойки	*G*	Код представления числа с одинарной точностью (знаковый бит и мантисса отделены пробелом)
1	2^0	127	0 01111111 00000000000000000000000
2	2^1	128	0 10000000 00000000000000000000000
8	2^3	130	0 10000010 00000000000000000000000
0.125	2^{-3}	124	0 01111100 00000000000000000000000
7	$2^2+2^1+2^0$	129	0 10000001 11000000000000000000000
7.5	$2^2+2^1+2^0+2^{-1}$	129	0 10000001 11100000000000000000000
-1	-2^0	127	1 01111111 00000000000000000000000
-7	$-2^2-2^1-2^0$	129	1 10000001 11000000000000000000000

Представление даже целых чисел в вещественном и целом форматах не совпадает (за исключением положительного нуля). Например, для представленных в табл. 2 значений 1, 2, 8, 7, -1, -7 их целые представления в виде 32-разрядного целого числа будут, соответственно, иметь коды

```
0 00000000 00000000000000000000001,
0 00000000 00000000000000000000010,
0 00000000 00000000000000000001000,
0 00000000 00000000000000000000111,
1 11111111 11111111111111111111111,
1 11111111 11111111111111111111001.
```

Обратим внимание, что некоторые десятичные дроби не имеют точного двоичного представления, например, преобразуя число 0.1 в двоичную систему счисления, получим периодическую дробь, и её бинарное представление с одинарной точностью будет

```
0 01111011 10011001100110011001101.
```

Легко проверить прямым вычислением, что представленное вещественное число чуть больше 0.1, а предыдущее

```
0 01111011 10011001100110011001100
```

чуть меньше.

Это одна из основных причин того, что преобразование числа из текстовой формы в формат с плавающей точкой и обратно может привести к изменению значения числа. Поэтому обычно рекомендуют для исключения возможной потери точности промежуточные результаты вычислений сохранять в файлах (если, конечно, это необходимо) в двоичном, а не текстовом формате.

2.3. Число с плавающей точкой двойной, расширенной и четверной точности

Стандарт IEEE 754-2008, кроме числа одинарной точности, описывает ещё число двойной и четверной точности.

Число двойной точности занимает 64 бита (8 байт), число бит под характеристику $N_1 = 11$, число бит под мантиссу $N_2 = 52$. Сдвиг порядка равен 1023, формула (10) принимает вид

$$A = (-1)^s \left(1 + M \cdot 2^{-52}\right) \cdot 2^{G-1023}, \tag{14}$$

а формула (11) –

$$A = (-1)^s M \cdot 2^{-1074}. \tag{15}$$

Число четверной точности занимает 128 бит, из них под характеристику отводится $N_1 = 15$ бит, под мантиссу $N_2 = 112$ бит.

Заметим, что в большинстве современных языков программирования тип данных «вещественное число четверной точности» не вводится (но, возможно, будет добавлен в следующих версиях). Однако для многих языков программирования существуют компиляторы, которые поддерживают тип вещественного числа расширенной точности (которого в стандарте IEEE 754-2008 нет), поскольку он был реализован аппаратно в архитектуре процессоров x86.

Число расширенной точности занимает 80 бит (10 байт), число бит под характеристику $N_1 = 15$, под мантиссу – 64. Заметим, что при хранении числа в расширенной точности число хранится без скрытого бита, и один из бит мантиссы (первый) выделен под целую часть числа. Поэтому фактически для этого формата $N_2 = 63$.

Например, в языке FORTRAN число расширенной точности в компиляторах, которые поддерживают такой тип, описывается как REAL*10, в C++ – как *long double*. Обратим внимание, что один из самых распространенных компиляторов C++, компилятор Microsoft, хотя и поддерживает тип *long double*, но фактически отводит под переменную этого типа 8 байт.

Представим все сведения о форматах числа с плавающей точкой в виде итоговой таблицы:

Точность	Одинарная	Двойная	Расширенная	Четверная
Размер (байты)	4	8	10	16
Число бит характеристики (N_1)	8	11	15	15
Число бит мантиссы (N_2)	23	52	63	112
Число десятичных знаков, которые сохраняются в мантиссе в нормализованном виде $\left(\left[\log_{10}\left(2^{N_2+1}\right)\right]\right)$	7	15	19	34
Наименьшее положительное значение	$1{,}4\cdot10^{-45}$	$5{,}0\cdot10^{-324}$	$1{,}9\cdot10^{-4951}$	10^{-4965}
Наименьшее нормализованное положительное значение	$1{,}2\cdot10^{-38}$	$2{,}3\cdot10^{-308}$	$3{,}4\cdot10^{-4932}$	$3{,}4\cdot10^{-4932}$
Наибольшее значение	$3{,}4\cdot10^{+38}$	$1{,}7\cdot10^{+308}$	$1{,}1\cdot10^{+4932}$	$1{,}2\cdot10^{+4932}$

Обратим внимание, что в связи с существованием различных форматов представления чисел и ошибок округления при вычислениях сравнение вещественных чисел на равенство чаще всего бессмысленно. Например, если присвоить значение 0.1 вещественной переменной двойной точности и затем сравнить эту переменную с константой 0.1, то во многих языках программирования получится, что значения не равны. Дело в том, что 0.1, как уже было сказано, в двоичной системе счисления – периодическая дробь, и поэтому её округление зависит от размера мантиссы, а представление констант во многих языках программирования по умолчанию не производится с двойной точностью. Поэтому сравнивать два вещественных числа имеет смысл только с заданной точностью, причем лучше с относительной.

3. РАЗРЕЖЕННЫЕ МАТРИЦЫ

Одним из наиболее затратных по ресурсам этапов большинства современных алгоритмов численного моделирования является этап решения системы линейных алгебраических уравнений. Часто матрица этой системы имеет очень большую размерность (например, несколько миллионов), но при этом матрица содержит относительно небольшое число ненулевых элементов. Очевидно, что хранить такую матрицу требуется каким-то специальным образом, поскольку хранить все ячейки матрицы размерности миллион несколько затруднительно (для этого потребуется сохранить $1\,000\,000^2 = 10^{12}$ вещественных чисел, что даже при одинарной точности означает $4 \cdot 10^{12}$ байт, а это несколько больше объема оперативной памяти современного компьютера). Матрица, которая содержит достаточно большое (относительно размерности) число нулевых элементов, обычно называется ***разреженной*** и хранится специальным образом. Если же матрица хранится обычным образом, т.е. без учета структуры расположения её нулевых элементов, то такой формат хранения матрицы иногда называют ***плотным***.

Существует множество форматов хранения разреженных матриц. Обычно формат матрицы выбирают в зависимости от алгоритма, который будет использовать эту матрицу, и структуры матрицы.

Наиболее распространенными являются четыре формата: ***диагональный, ленточный, профильный и разреженный***. *Далее эти форматы будут рассмотрены только для квадратных матриц.*

Каждый из форматов хранения имеет несколько модификаций в зависимости от следующих характеристик использующего их алгоритма:

– учитывается ли симметрия матрицы;

– отдельно ли используются верхний и нижний треугольники матрицы;

– требуется ли ускоренный доступ к строкам матрицы или к столбцам.

Рассмотрим эти форматы подробнее. При описании форматов будем называть «ненулевыми» те элементы, которые фактически предполагается хранить (на самом деле среди них реально могут быть и нули).

3.1. Диагональный формат

Диагональный формат хранения матриц используется, когда все ненулевые элементы матрицы расположены на относительно небольшом числе диагоналей. Пример такой матрицы приведен на рис. 1. Индексы i_1, i_2, i_3, i_4, i_5 обозначают сдвиг диагоналей нижнего и верхнего треугольника от главной диагонали.

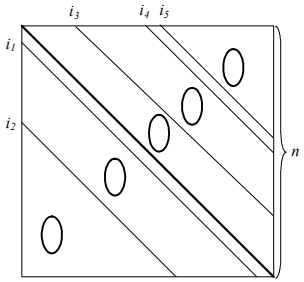


Рис. 1. Диагональный формат

Хранение таких матриц обычно организуется достаточно просто, для работы с матрицами диагонального вида используются прямоугольные матрицы размером $n \times m$, где n – размерность исходной матрицы, m – количество ненулевых диагоналей. Прямоугольная матрица получается путем доопределения побочных диагоналей до общей размерности нулями (либо в начале диагоналей нижнего треугольника и в конце диагоналей верхнего треугольника – тогда хранение построчное, либо только в начале диагоналей – тогда хранение нижнего треугольника исходной матрицы осуществляется по строкам, верхнего – по столбцам). Дополнительно необходимо хранить одномерный целый массив размерностью $m-1$, в котором будет указан сдвиг каждой побочной диагонали от главной: положительные индексы для верхнего треугольника, отрицательные для нижнего треугольника. Например, для матрицы, изображенной на рис. 1, это будет массив $I = [-i_2,\ -i_1,\ i_3,\ i_4,\ i_5]$.

Заметим, что для диагональных матриц существует несколько частных случаев, для которых могут использоваться и другие форматы. Например, для хранения пятидиагональной матрицы, изображенной на рис. 2, достаточно одного параметра m (вместо пяти параметров).

Поэтому в рамках учебного пособия нет смысла описывать все возможные форматы диагональных матриц, тем более что алгоритмы работы с ними достаточно простые.

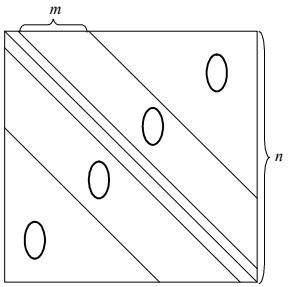


Рис. 2. Пятидиагональная матрица

3.2. Ленточный формат

Ленточный формат хранения матриц используется, когда все ненулевые элементы матрицы A расположены на диагоналях, прилегающих к главной диагонали (рис. 3), т.е. $a_{ij} = 0$, если $|i - j| > k$, где k – ***полуширина***, а $m = 2k + 1$ – ***ширина ленты***. Обычно ширина ленты значительно меньше размерности матрицы.

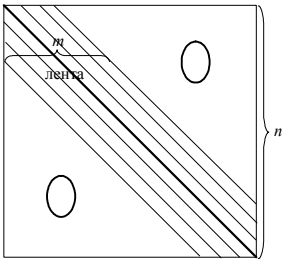


Рис. 3. Ленточная матрица

Для хранения исходной матрицы размерностью n и шириной ленты m обычно используется одна (или две в случае раздельного хранения верхнего и нижнего треугольников) прямоугольная матрица. Очевидно, что для такого способа хранения необходимо дополнить матрицу некоторыми лишними элементами, как это будет показано ниже. Заметим, что выгода от хранения лишних элементов матрицы заключается только в

более простом (и, следовательно, быстром) доступе к элементам (например, за счет использования двумерных массивов). Поэтому в тех случаях, когда затраты на хранение лишних элементов становятся велики (т.е. ширина ленты сравнима с размерностью матрицы), от хранения лишних элементов лучше отказаться.

В зависимости от требований использующего матрицу алгоритма ее можно хранить строками, как показано на рис. 4, столбцами, как это показано на рис. 5, или хранение может быть разным для нижнего и верхнего треугольников матрицы, как это показано на рис. 6 и 7. В последнем случае, при раздельном хранении верхнего и нижнего треугольников, главная диагональ обычно хранится отдельно.

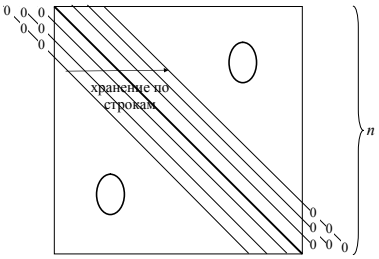


Рис. 4. Строчное хранение ленточной матрицы

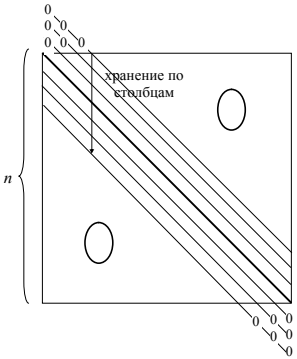


Рис. 5. Столбцовое хранение ленточной матрицы

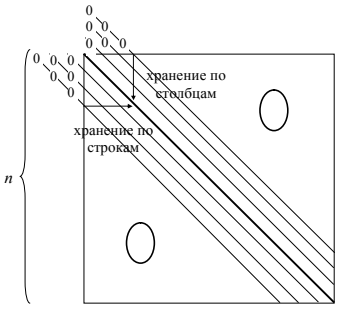


Рис. 6. Строчно-столбцовое хранение ленточной матрицы

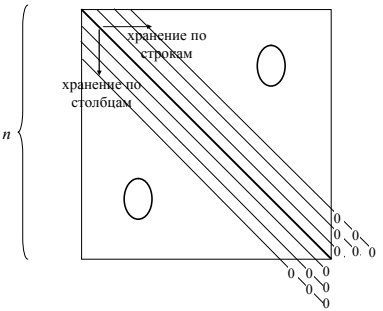


Рис. 7. Столбцово-строчное хранение ленточной матрицы

Заметим, что если хранимая матрица симметричная, то можно хранить только один (например, нижний) треугольник матрицы (строками или столбцами).

Рассмотрим на примере два возможных способа хранения ленточной матрицы размерностью 10×10 с шириной ленты $m=7$, изображенной на рис. 8.

$$\begin{bmatrix}
\mathbf{1} & 12 & 13 & 14 & & & & & & \\
2 & \mathbf{2} & 13 & 14 & 15 & & & & & \\
3 & 3 & \mathbf{3} & 14 & 15 & 16 & & & & \\
4 & 4 & 4 & \mathbf{4} & 15 & 16 & 17 & & & \\
 & 5 & 5 & 5 & \mathbf{5} & 16 & 17 & 18 & & \\
 & & 6 & 6 & 6 & \mathbf{6} & 17 & 18 & 19 & \\
 & & & 7 & 7 & 7 & \mathbf{7} & 18 & 19 & 20 \\
 & & & & 8 & 8 & 8 & \mathbf{8} & 19 & 20 \\
 & & & & & 9 & 9 & 9 & \mathbf{9} & 20 \\
 & & & & & & 10 & 10 & 10 & \mathbf{10}
\end{bmatrix}$$

Рис. 8. Ленточная матрица

Сначала рассмотрим построчное хранение. Доопределим матрицу нулями в начале диагоналей для нижнего треугольника, в конце диагоналей – для верхнего треугольника, как это показано на рис. 5.

Тогда фактически для хранения матрицы будет использоваться массив:

$$\begin{bmatrix}
0 & 0 & 0 & \mathbf{1} & 12 & 13 & 14 \\
0 & 0 & 2 & \mathbf{2} & 13 & 14 & 15 \\
0 & 3 & 3 & \mathbf{3} & 14 & 15 & 16 \\
4 & 4 & 4 & \mathbf{4} & 15 & 16 & 17 \\
5 & 5 & 5 & \mathbf{5} & 16 & 17 & 18 \\
6 & 6 & 6 & \mathbf{6} & 17 & 18 & 19 \\
7 & 7 & 7 & \mathbf{7} & 18 & 19 & 20 \\
8 & 8 & 8 & \mathbf{8} & 19 & 20 & 0 \\
9 & 9 & 9 & \mathbf{9} & 20 & 0 & 0 \\
10 & 10 & 10 & \mathbf{10} & 0 & 0 & 0
\end{bmatrix}$$

Теперь рассмотрим раздельное строчно-столбцовое хранение нижнего и верхнего треугольников для той же самой матрицы, изображенной на рис. 8. Дополним матрицу нулями так, как это показано на рис. 7. Тогда фактически для хранения матрицы будут использоваться массивы:

$$al = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 2 \\ 0 & 3 & 3 \\ 4 & 4 & 4 \\ 5 & 5 & 5 \\ 6 & 6 & 6 \\ 7 & 7 & 7 \\ 8 & 8 & 8 \\ 9 & 9 & 9 \\ 10 & 10 & 10 \end{bmatrix} \quad au = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 12 \\ 0 & 13 & 13 \\ 14 & 14 & 14 \\ 15 & 15 & 15 \\ 16 & 16 & 16 \\ 17 & 17 & 17 \\ 18 & 18 & 18 \\ 19 & 19 & 19 \\ 20 & 20 & 20 \end{bmatrix} \quad di = \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \\ 9 \\ 10 \end{bmatrix},$$

где *di* – вектор диагональных элементов, *al* и *au* – прямоугольные матрицы размерностью 10×3, в которых хранятся нижний треугольник по строкам и верхний треугольник по столбцам соответственно.

3.3. Профильный формат

Изучим идею профильного формата на примере раздельного хранения нижнего и верхнего треугольников, при этом нижний треугольник будем хранить строками, а верхний – столбцами.

Профильный формат хранения матриц в некотором смысле является обобщением ленточного формата на случай, когда матрица не обладает строго диагональной структурой и ненулевые элементы расположены в произвольном порядке, но при этом они сосредоточены у главной диагонали (т.е. в каждой строке матрицы ширина ленты будто бы разная).

В такой ситуации в каждой строке (столбце) можно выделить так называемый ***профиль строки*** (***столбца***) – это количество элементов строки (столбца) от первого ненулевого элемента в строке (столбца) до диагонального элемента (не включая его). Заметим, что часто всю структуру матрицы, в которой выделены профили строк (столбцов), называют ***профилем матрицы*** (т.е. профиль матрицы определяет положение ненулевых элементов в ней, но не их значения)**.**

Обычно в рассматриваемом формате хранят несимметричные матрицы, профили строк нижнего треугольника которых обязательно совпадают с профилями столбцов верхнего треугольника, т.е. структурно матрица симметрична относительно главной диагонали (хотя сами значения в верхнем и нижнем треугольнике на симметричных местах могут быть разные). Пример такой матрицы приведен на рис. 9.

Для хранения квадратной матрицы размерности n в профильном формате используется следующая структура данных:

1) Вещественный массив ***di***. Этот массив имеет размерность n и содержит последовательно диагональные элементы матрицы.

2) Вещественные массивы ***al*** и ***au*** для хранения внедиагональных элементов матрицы нижнего (по строкам) и верхнего (по столбцам) треугольников матрицы соответственно. Если матрица симметричная не только по структуре, но и по значениям, то массивы al и au совпадают и хранится только один из них.

3) Целочисленный массив ***ia*** для хранения информации о профиле. Элемент $ia(k)$ равен индексу (в нумерации 1), с которого начинаются элементы k-й строки (столбца) в массивах al и au. Размерность массива ia равна $n+1$, причем $ia(n+1)$ равен индексу первого незанятого элемента в массивах al и au (т.е. размерность этих массивов равна $ia(n+1)-1$). Разность $ia(i+1)-ia(i)$ равна значению профиля i-й строки (столбца) нижнего (верхнего) треугольника. Очевидно, что в любом случае $ia(1)=ia(2)=1$.

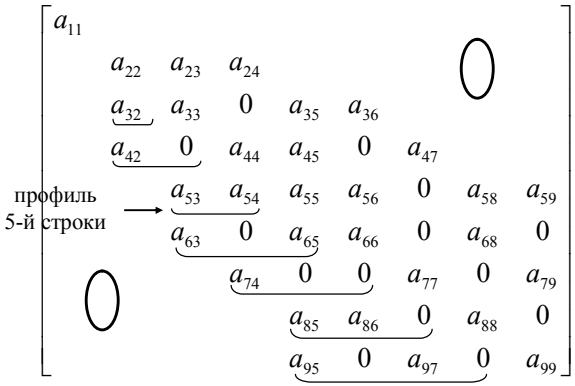


Рис. 9. Профильная матрица

В качестве примера приведем значения всех массивов для матрицы размерностью 9×9, изображенной на рис. 9.

$$ia = \{\,1 \quad 1 \quad 1 \quad 2 \quad 4 \quad 6 \quad 9 \quad 12 \quad 15 \quad 19\}$$

$$di = \{a_{11} \quad a_{22} \quad a_{33} \quad a_{44} \quad a_{55} \quad a_{66} \quad a_{77} \quad a_{88} \quad a_{99}\}$$

$$\begin{array}{rcl} & 1\ \ 2\ \ 3\ \ 4\ \ 5\ \ 6\ \ 7\ \ 8\ \ 9\ \ 10\ 11\ 12\ \ 13\ 14\ 15\ 16\ 17\ 18 & \\ al = \{ & a_{32}\ \ a_{42}\ \ 0\ \ a_{53}\ \ a_{54}\ \ a_{63}\ \ 0\ \ a_{65}\ \ a_{74}\ \ 0\ \ 0\ \ a_{85}\ \ a_{86}\ \ 0\ \ a_{95}\ \ 0\ \ a_{97}\ \ 0 & \} \\ au = \{ & a_{23}\ \ a_{24}\ \ 0\ \ a_{35}\ \ a_{45}\ \ a_{36}\ \ 0\ \ a_{56}\ \ a_{47}\ \ 0\ \ 0\ \ a_{58}\ \ a_{68}\ \ 0\ \ a_{59}\ \ 0\ \ a_{79}\ \ 0 & \} \end{array}$$

Рассмотренная структура данных позволяет легко находить элементы любой строки нижнего треугольника (или столбца верхнего), например, для 6-й строки: первый элемент 6-й строки: $al\left(ia\left(6\right)\right) = al\left(6\right) = a_{63}$. Количество хранимых элементов (профиль) 6-й строки равно $ia\left(7\right) - ia\left(6\right) = 9 - 6 = 3$, т.е. это три элемента в массиве al, начиная с элемента $al\left(6\right) = a_{63}$: a_{63}, 0, a_{65}. Поиск же элементов столбца нижнего (строки верхнего) треугольника в этой структуре данных не столь прост, что необходимо учитывать в использующих эту структуру данных алгоритмах.

Строго говоря, рассмотренный формат хранения правильнее называть ***профильным строчно-столбцовым.*** Поскольку, как и для лен-

точной матрицы, для профильной возможно аналогично строчно-столбцовому определить ***столбцово-строчный профильный*** формат, а также ***строчный, столбцовый*** и т.п. (естественно, для, например, строчного хранения профильной матрицы потребуется соответственно изменить использованное выше определение *профиля строки*).

3.4. Разреженный формат

Разреженный формат хранения матриц используется, когда в матрице ненулевые элементы расположены в произвольном порядке. Этот формат похож на профильный формат, но хранятся не все элементы внутри профиля, а только ненулевые элементы. Поэтому дополнительно используется еще один массив, содержащий информацию о втором индексе внедиагонального элемента. Обычно при рассмотрении разреженных матриц используют понятие ***портрет матрицы*** – местоположение её ненулевых элементов. Чаще всего на практике встречаются матрицы с симметричным относительно главной диагонали портретом (хотя значения на симметричных местах матрицы могут быть разными). Поэтому далее мы будем понимать под несимметричной матрицей портретно-симметричную матрицу с несовпадающими значениями на симметричных местах.

Как и остальные форматы, ***разреженный формат*** может быть ***строчным*** (когда хранятся строки и дополнительно номера столбцов), ***столбцовым*** (когда хранятся столбцы и дополнительно номера строк) или смешанным, когда нижний и верхний треугольник хранятся симметрично (***строчно-столбцовый*** и ***столбцово-строчный*** форматы).

Рассмотрим структуру данных для ***разреженного строчно-столбцового формата***.

1) Вещественный массив ***di***. Этот массив имеет размерность n и содержит последовательно диагональные элементы матрицы.

2) Вещественные массивы ***al*** и ***au*** для хранения внедиагональных элементов матрицы нижнего (по строкам) и верхнего (по столбцам) треугольников матрицы соответственно. Если матрица симметричная, то массивы *al* и *au* совпадают и хранится только один из них.

3) Целочисленный массив ***ja*** содержит номера столбцов (строк) хранимых внедиагональных элементов нижнего (верхнего) треугольника матрицы. Для произвольного $j \le m$, где m – размерность массивов ja, al и au, $ja(j)$ – это номер столбца для элемента $al(j)$ и номер строки для элемента $au(j)$.

4) Целочисленный массив ***ia***. Элемент $ia(k)$ равен индексу (в нумерации 1), с которого начинаются элементы k-й строки (столбца) в массивах ja, al и au. Размерность массива ia равна $n+1$, причем $ia(n+1)$ равен индексу первого незанятого элемента в массивах ja, al и au (т.е. размерность этих массивов равна $ia(n+1)-1$). Разность $ia(i+1)-ia(i)$ равна количеству хранимых внедиагональных элементов i-й строки (столбца) нижнего (верхнего) треугольника. Очевидно, что в любом случае $ia(1)=ia(2)=1$.

Заметим, что массивы ia и ja полностью определяют места хранимых элементов и часто именно они называются ***портретом матрицы***.

В качестве примера заполним описанную структуру данных для матрицы, приведенной на рис. 9:

$$ia = \{\,1 \quad 1 \quad 1 \quad 2 \quad 3 \quad 5 \quad 7 \quad 8 \quad 10 \quad 12\}$$

$$di = \{a_{11} \quad a_{22} \quad a_{33} \quad a_{44} \quad a_{55} \quad a_{66} \quad a_{77} \quad a_{88} \quad a_{99}\}$$

$$\begin{array}{lccccccccccc} & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 \\ ja = \{ & 2 & 2 & 3 & 4 & 3 & 5 & 4 & 5 & 6 & 5 & 7 \ \} \end{array}$$

$$al = \{\,a_{32} \quad a_{42} \quad a_{53} \quad a_{54} \quad a_{63} \quad a_{65} \quad a_{74} \quad a_{85} \quad a_{86} \quad a_{95} \quad a_{97}\,\}$$

$$au = \{\,a_{23} \quad a_{24} \quad a_{35} \quad a_{45} \quad a_{36} \quad a_{56} \quad a_{47} \quad a_{58} \quad a_{68} \quad a_{59} \quad a_{79}\,\}$$

Элементы в этой структуре данных можно искать следующим образом, например, для шестой строки: начало шестой строки в массивах ja и al: $ia(6)=5$. Количество элементов в шестой строке $ia(7)-ia(6)=7-5=2$. Начиная с пятого элемента в массиве al, последовательно находятся два элемента шестой строки a_{63} и a_{65}, которые располагаются в столбцах $ja(ia(6))=3$ и $ja(ia(6)+1)=5$.

3.5. Общие рекомендации

Мы рассмотрели некоторые из возможных форматов хранения разреженных матриц. Обратим внимание, что алгоритмы работы с разреженными матрицами должны учитывать используемую для хранения матрицы структуру данных. Например, алгоритм умножения матрицы на вектор не должен пытаться умножать строку матрицы на вектор, если матрица хранится в каком-то из столбцовых форматов, а должен умножать каждый столбец на соответствующий элемент вектора и накапливать в векторе результата сумму. Это позволит существенно сократить время работы алгоритма.

При необходимости использования специальных форматов, например, для вызова программ из каких-либо специализированных библиотек, следует внимательно ознакомиться с их описанием в документации библиотеки. Большинство библиотек программ для работы с разреженными матрицами позволяет выбрать наиболее удобный из несколько форматов, многие библиотеки допускают использование индексных массивов с нумерацией как с единицы, так и с нуля.

4. ИНТЕРПОЛЯЦИОННЫЙ ПОЛИНОМ

Рассмотрим классическую задачу ***интерполяции***. Пусть известны значения некоторой функции f в $n+1$ различных точках $x_0, x_1, ..., x_n$. Требуется восстановить (естественно, приближенно) значение функции f в произвольной точке x. Одним из способов решения этой задачи является построение ***интерполяционного полинома***. Интерполяционным называется такой полином $L_n(x)$ степени n, который в точках x_i (называемых в этом случае ***узлами интерполяции***) принимает заданные значения:

$$L_n(x_i) = f(x_i), \ \ i = \overline{0,n}. \tag{16}$$

Приближенное восстановление функции f по формуле $f(x) \approx L_n(x)$ в случае, когда x расположен внутри минимального отрезка, содержащего все узлы интерполяции $x_0, x_1, ..., x_n$, называется ***интерполяцией*** функции f. Если же x расположен вне этого отрезка, то процедуру приближенного восстановления значения функции обычно называют ***экстраполяцией***.

Теорема (о существовании и единственности интерполяционного полинома). Существует единственный интерполяционный полином *n*-й степени, удовлетворяющий условию (16).

▸ Докажем сначала единственность такого полинома. Предположим, что условию (16) удовлетворяют два различных полинома степени n. Вычтем их друг из друга. Полученная разность, очевидно, является полиномом степени не выше n. При этом в узлах интерполяции эта разность по условию (16) равна нулю, т.е. мы получили полином, у которого $n+1$ различных корней, но степень не выше n, что возможно только в случае, когда разность тождественно равна нулю.

Существование докажем конструктивно. Рассмотрим функции

$$p_i(x) = \frac{(x-x_0)...(x-x_{i-1})(x-x_{i+1})...(x-x_n)}{(x_i-x_0)...(x_i-x_{i-1})(x_i-x_{i+1})...(x_i-x_n)},\ \ i=\overline{0,n}. \qquad (17)$$

Очевидно, что эти функции по построению являются полиномами степени n, в точке x_i равны единице и в остальных узлах интерполяции равны нулю. Поэтому их линейная комбинация с весами $f(x_i)$

$$L_n(x) = \sum_{i=0}^{n} p_i(x) f(x_i) \qquad (18)$$

является искомой функцией. ◂

Интерполяционный полином, представленный в виде (18), называется ***интерполяционным полиномом Лагранжа***, а функции (полиномы) вида (17) – ***базисными полиномами Лагранжа***.

5. ЧИСЛЕННОЕ ИНТЕГРИРОВАНИЕ

Классическая задача численного интегрирования заключается в следующем. Пусть необходимо вычислить определенный интеграл

$$I = \int_a^b f(x)dx \tag{19}$$

от непрерывной на $[a,b]$ функции f. При этом сама функция f, возможно, неизвестна, или непредставима аналитически, но её вычисление возможно в любой точке отрезка $[a,b]$. Приближенное равенство

$$\int_a^b f(x)dx \approx \sum_{i=1}^{n} q_i f(x_i) \tag{20}$$

где $q_i \in \mathbb{R}$, $x_i \in [a,b]$, называется ***квадратурной формулой*,** определяемой ***весами*** q_i и ***узлами*** x_i.

Очевидно, что при построении квадратурных формул следует оценивать их точность и время вычислений. Для оценки времени вычислений обычно предполагается, что самая «дорогая» операция – это вычисление самой функции, т.е. оценку времени вычисления квадратурной формулы заменяют числом вычислений подынтегральной функции. Для оценки же точности можно использовать два способа.

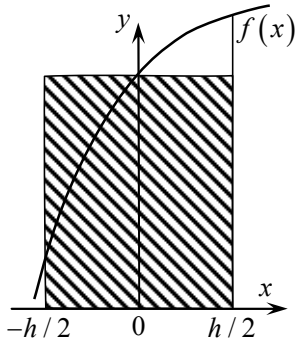
Первый способ заключается в оценке максимальной степени полинома m, для которого приближенное равенство (20) является точным. Будем называть эту степень ***порядком точности*** квадратурной формулы.

Второй способ определяется несколько сложнее. Будем считать, что отрезок $[a,b]$ разбивается на n одинаковых отрезков размера h (этот размер называют ***шагом*** разбиения), и вычисления по квадратурной формуле на каждом шаге производятся как-то однотипно (по одинаковой формуле относительно границ отрезка). Обозначим через I^* точное значение интеграла (19) (которое, конечно, не всегда известно), а через I^h значение, вычисленное по изучаемой квадратурной формуле с шагом h.

Естественно, квадратурная формула обычно строится таким образом, чтобы разность $I^* - I^h$ стремилась к нулю при стремлении к нулю шага h. Порядок малости этой разности относительно шага обычно называют ***порядком метода*** (иногда его называют ***порядком аппроксимации***) и именно его чаще всего используют для оценки точности метода.

Рассмотрим теперь принципы построения квадратурных формул.

5.1. Метод прямоугольников



Простейшим методом численного интегрирования является метод прямоугольников. Его идея основана фактически на определении интегрирования. На каждом отрезке значение интеграла вычисляется как площадь прямоугольника, основание которого – шаг интегрирования, а высота – значение функции в центре отрезка. В соответствии с обозначениями на рисунке

$$\int_{-\frac{h}{2}}^{\frac{h}{2}} f(x)dx = hf(0) + \varepsilon(h), \qquad (21)$$

где $\varepsilon(h)$ – остаточный член (погрешность формулы).

Получим остаточный член для формулы (21) в явном виде для функции[2] $f \in C^2\left[-\frac{h}{2}, \frac{h}{2}\right]$. Для этого запишем первообразную $F(x)$ функции $f(x)$ в виде

$$F(x) = \int_0^x f(t)dt. \qquad (22)$$

[2] $C^k(\Omega)$ – класс k раз непрерывно дифференцируемых на множестве Ω функций

Так как по определению

$$F(0)=0,\ F'(x)=f(x),\ F''(x)=f'(x),\ F'''(x)=f''(x)\ \ldots,$$

то согласно формуле Тейлора с остаточным членом в форме Лагранжа имеем:

$$F\left(\pm\frac{h}{2}\right)=\pm\frac{h}{2}f(0)+\frac{h^2}{8}f'(0)\pm\frac{h^3}{48}f''(\xi_{\pm}), \tag{23}$$

где $\xi_{-}\in\left[-\frac{h}{2},0\right]$, $\xi_{+}\in\left[0,\frac{h}{2}\right]$.

Учитывая, что $F(x)$ является первообразной для функции $f(x)$ (см. (22)), из (21) и (23) получим:

$$\int\limits_{-\frac{h}{2}}^{\frac{h}{2}} f(x)dx=F\left(\frac{h}{2}\right)-F\left(-\frac{h}{2}\right)=hf(0)+\frac{h^3}{24}\frac{f''(\xi_{-})+f''(\xi_{+})}{2}, \tag{24}$$

откуда

$$\int\limits_{-\frac{h}{2}}^{\frac{h}{2}} f(x)dx=hf(0)+\frac{h^3}{24}f''(\xi),\quad |\xi|\le\frac{h}{2}. \tag{25}$$

Формула (25) называется ***канонической квадратурной формулой метода прямоугольников*** с остаточным членом.

Обратим внимание, что каноническая формула (25) содержит вторую производную интегрируемой функции. Поэтому, во-первых, формула (25) имеет первый порядок точности (поскольку полиномы до первого порядка имеют нулевую вторую производную). Во-вторых, этот остаточный член показывает, что точность численного интегрирования существенно зависит от интегрируемой функции (поскольку содержит её вторую производную). А в-третьих, этот остаточный член напоминает, что все наши выкладки делались для функций, имеющих на интервале непрерывную вторую производную. То есть близость значения формулы метода прямоугольников к точному значению интеграла обоснована (см. (25)) только для функций, у которых эта производная есть. Вообще говоря, численное интегрирование функций, не обладающих достаточной гладкостью, является отдельной проблемой, выходящей за рамки данного пособия.

Перейдем теперь от канонической формулы к вычислению интеграла на всем отрезке $[a,b]$. Разобьём область интегрирования $[a,b]$ на N отрезков точками $a = x_0 < x_1 < ... < x_N = b$ (обычно такое разбиение называют ***сеткой***, а его точки разбиения называют ***узлами сетки***). Обозначим через h_i длину частичного сегмента $[x_{i-1}, x_i]$ (эти длины h_i часто называют ***шагами сетки***). На каждом частичном отрезке применим каноническую квадратурную формулу и просуммируем полученные результаты. Построенная таким образом квадратурная формула называется ***усложненной***:

$$\int_a^b f(x)dx = \sum_{i=1}^{N} h_i f\left(\frac{x_{i-1}+x_i}{2}\right) + \sum_{i=1}^{N} \frac{h_i^3}{24} f''(\xi_i), \quad \xi_i \in [x_{i-1}, x_i]. \qquad (26)$$

Заметим, что вид остаточного члена в формуле (26) не вполне удачен, поскольку по нему сложно судить о порядке метода (хотя легко видеть, что порядок точности остался первым). Проблема в том, что понятие «порядок метода» мы определяли для постоянного шага, а не для переменного. Для непостоянного шага порядок метода обычно определяют следующим образом. Выбирается некоторый ***параметр сетки*** h, называемый тоже ***шагом***, который является общим множителем всех остальных шагов (например, максимальный или минимальный из всех), т.е. $h_i = h \cdot c_i$. Смысл в том, чтобы определить понятие «изменение шага сетки», т.е. при уменьшении параметра h вдвое все шаги сетки одновременно уменьшатся вдвое. Тогда появляется возможность определить порядок малости остаточного члена относительно этого параметра h, что и будем называть далее ***порядком метода на неравномерной сетке***.

Определим порядок метода прямоугольников. Пусть

$$M = \max_{x\in[a,b]} f''(x), \quad C = \left(\max_{i=\overline{1,n}} c_i\right)^2.$$

Тогда остаточный член в формуле (26) можно оценить следующим образом:

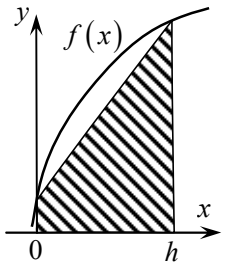
$$\varepsilon(h) = \frac{1}{24}\sum_{i=1}^{N} h_i^3 f''(\xi_i) \le \frac{M}{24}\sum_{i=1}^{N} h_i^2 h_i \le \frac{M}{24} C h^2 \sum_{i=1}^{N} h_i = \frac{C\cdot M(b-a)}{24} h^2.$$

Так как $\dfrac{C \cdot M\left(b-a\right)}{24}$ – константа, не зависящая от параметра h, то можно утверждать, что остаточный член усложненной квадратурной формулы прямоугольников имеет второй порядок относительно параметра h, т.е. $\varepsilon\left(h\right)=O\left(h^2\right)$.

5.2. Семейство формул Ньютона–Котеса

Описанный выше метод прямоугольников фактически построен на том, что интегрируемая функция заменяется кусочно-постоянной. Идея улучшить точность квадратурных формул путем более точной интерполяции подынтегрального выражения приводит к ***семейству формул Ньютона–Котеса***. Основной идеей метода Ньютона–Котеса является замена на каждом шаге интегрирования подынтегральной функции каким-либо интерполяционным полиномом (чаще всего полиномом Лагранжа с равномерной расстановкой точек, включая концы отрезка).

5.2.1 Метод трапеций



Простейшим вариантом формул Ньютона–Котеса является ***метод трапеций***, получаемый заменой подынтегральной функции на каждом шаге линейной функцией, определяемой двумя точками – значениями интегрируемой функции на концах отрезка.

Формула получила такое название, поскольку для положительной функции, как это видно из рисунка, площадь криволинейной трапеции, ограниченной сверху графиком функции, аппроксимируется площадью трапеции с вершинами $\left(0,0\right)$, $\left(0,f\left(0\right)\right)$, $\left(h,f\left(h\right)\right)$, $\left(h,0\right)$:

$$\int_0^h f\left(x\right)dx = h\frac{f\left(0\right)+f\left(h\right)}{2}+\varepsilon\left(h\right), \qquad (27)$$

где $\varepsilon\left(h\right)$ – остаточный член (погрешность формулы).

Получим остаточный член для формулы (27) в явном виде для функции $f \in C^2[0,h]$. Как и ранее, определим первообразную $F(x)$ функции $f(x)$ выражением (22). По формуле Тейлора с остаточным членом в интегральной форме

$$f(h) = f(0) + hf'(0) + \int_0^h (h-t) f''(t)\,dt, \tag{28}$$

и аналогично для первообразной $F(x)$:

$$F(h) = F(0) + hF'(0) + \frac{h^2}{2}F''(0) + \frac{1}{2}\int_0^h (h-t)^2 F'''(t)\,dt =$$

$$= 0 + hf(0) + \frac{h^2}{2}f'(0) + \frac{1}{2}\int_0^h (h-t)^2 f''(t)\,dt. \tag{29}$$

Из выражений (27)-(29), учитывая, что $F(x)$ – первообразная $f(x)$, получаем:

$$\varepsilon(h) = \int_0^h f(x)\,dx - h\frac{f(0)+f(h)}{2} = F(h) - F(0) - h\frac{f(0)+f(h)}{2} =$$

$$= hf(0) + \frac{h^2}{2}f'(0) + \frac{1}{2}\int_0^h (h-t)^2 f''(t)\,dt - h\frac{f(0)+f(h)}{2} =$$

$$= h\frac{f(0)-f(h)}{2} + \frac{h}{2}\left(f(h) - f(0) - \int_0^h (h-t) f''(t)\,dt\right) + \frac{1}{2}\int_0^h (h-t)^2 f''(t)\,dt =$$

$$= \frac{1}{2}\int_0^h (t-h)\,t\,f''(t)\,dt.$$

Отсюда, используя теорему о среднем значении интеграла, получаем остаточный член в явном виде:

$$\varepsilon(h) = \frac{1}{2}\int_0^h (t-h)\,t\,f''(t)\,dt = \frac{1}{2}f''(\xi)\int_0^h (t-h)\,t\,dt = -\frac{h^3}{12}f''(\xi).$$

Таким образом, ***каноническая формула трапеций*** с остаточным членом имеет вид

$$\int_0^h f(x)\,dx = h\frac{f(0)+f(h)}{2} - \frac{h^3}{12}f''(\xi), \quad \xi \in [0,\, h]. \tag{30}$$

Для получения усложненной квадратурной формулы метода трапеций разобьем, как и для метода прямоугольников (см. с. 42), промежуток интегрирования на N отрезков. Просуммируем все значения, найденные на каждом из отрезков в соответствии с (30), получим ***усложненную квадратурную формулу трапеций***:

$$\int_a^b f(x)dx = \frac{1}{2}\left[f(a)\cdot h_1 + \sum_{i=1}^{N-1} f(x_i)\cdot(h_i + h_{i+1}) + f(b)\cdot h_N \right] + \varepsilon(h), \quad (31)$$

где h, как и в методе прямоугольников, общий параметр – шаг. Несложно показать, что остаточный член в формуле (31) имеет второй порядок малости относительно этого параметра.

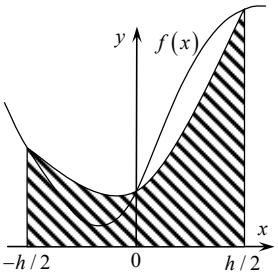
Если промежуток разбит на N равных частей, то усложненная квадратурная формула трапеций с остаточным членом примет более простой вид:

$$\int_a^b f(x)dx = h\left[\frac{f(a)}{2} + \sum_{i=1}^{N-1} f(x_i) + \frac{f(b)}{2} \right] - h^2 \frac{b-a}{12} f''(\xi),\ \xi \in [a, b].$$

Заметим, что по сравнению с методом прямоугольников точность метода трапеций в смысле порядка остаточного члена и порядка точности не увеличилась. В принципе, это и не слишком удивительно, поскольку число вычислений функции тоже практически не изменилось (при больших N, очевидно, не важно, делать N вычислений или $N+1$).

5.2.2 Метод Симпсона

Метод Симпсона также относится к семейству формул Ньютона–Котеса, в нем подынтегральная функция заменяется квадратичным полиномом.



Иногда этот метод называют методом парабол, поскольку для положительной функции, как видно из рисунка, интеграл $\int_{-h/2}^{h/2} f(x)dx$ приближенно заменяется площадью криволинейной трапеции, ограниченной сверху параболой, проходящей через точки:

$$\left(-\frac{h}{2}, f\left(-\frac{h}{2}\right)\right),\ (0, f(0)),\ \left(\frac{h}{2}, f\left(\frac{h}{2}\right)\right).$$

Указанная парабола задается уравнением

$$y(x)=f(0)+\frac{f\left(\frac{h}{2}\right)-f\left(-\frac{h}{2}\right)}{h}x+\frac{2\left[f\left(-\frac{h}{2}\right)-2f(0)+f\left(\frac{h}{2}\right)\right]}{h^2}x^2,$$

откуда получается ***каноническая формула Симпсона***

$$\int_{-h/2}^{h/2}f(x)dx=\int_{-h/2}^{h/2}y(x)dx+\varepsilon(h)=\frac{h}{6}\left(f\left(-\frac{h}{2}\right)+4f(0)+f\left(\frac{h}{2}\right)\right)+\varepsilon(h), \quad (32)$$

где $\varepsilon(h)$ – остаточный член (погрешность формулы).

Получим остаточный член формулы (32) в явном виде. Для этого потребуем от подынтегральной функции большей гладкости, чем для метода прямоугольников и трапеций, пусть теперь $f\in C^4\left[-\frac{h}{2},\frac{h}{2}\right]$. Пусть также F – функция, определяемая соотношением (22). Тогда согласно формуле Тейлора с остаточным членом в интегральной форме имеем:

$$f\left(\pm\frac{h}{2}\right)=f(0)\pm\frac{h}{2}f'(0)+\frac{h^2}{8}f''(0)\pm\frac{h^3}{48}f'''(0)+\frac{1}{6}\int_0^{h/2}\left(\frac{h}{2}-t\right)^3f^{(4)}(\pm t)dt, \quad (33)$$

и аналогично для первообразной $F(x)$:

$$F\left(\pm\frac{h}{2}\right)=\pm\frac{h}{2}f(0)+\frac{h^2}{8}f'(0)\pm\frac{h^3}{48}f''(0)+\frac{h^4}{384}f'''(0)\pm$$

$$\pm\frac{1}{24}\int_0^{h/2}\left(\frac{h}{2}-t\right)^4f^{(4)}(\pm t)dt, \quad (34)$$

Из выражений (33)–(34), учитывая, что $F(x)$ – первообразная $f(x)$, получаем

$$\varepsilon(h)=\int_{-h/2}^{h/2}f(x)dx-\frac{h}{6}\left[f\left(-\frac{h}{2}\right)+4f(0)+f\left(\frac{h}{2}\right)\right]=$$

$$=F\left(\frac{h}{2}\right)-F\left(-\frac{h}{2}\right)-\frac{h}{6}\left[f\left(-\frac{h}{2}\right)+4f(0)+f\left(\frac{h}{2}\right)\right]=$$

$$=hf(0)+\frac{h^3}{24}f''(0)+\frac{1}{24}\int_0^{h/2}\left(\frac{h}{2}-t\right)^4\left(f^{(4)}(t)+f^{(4)}(-t)\right)dt-$$

$$-\frac{h}{6}\left[2\left(f(0)+\frac{h^2}{8}f''(0)\right)+\frac{1}{6}\int_0^{h/2}\left(\frac{h}{2}-t\right)^3\left(f^{(4)}(t)-f^{(4)}(-t)\right)dt+4f(0)\right]=$$

$$=\int_0^{h/2}\left[\frac{1}{24}\left(\frac{h}{2}-t\right)^4-\frac{h}{36}\left(\frac{h}{2}-t\right)^3\right]\left(f^{(4)}(t)+f^{(4)}(-t)\right)dt \qquad (35)$$

Применив теорему о среднем и преобразовав полученное значение с учётом того, что

$$\int_0^{h/2}\left(\frac{h}{2}-t\right)^3\left(\frac{1}{24}\left(\frac{h}{2}-t\right)-\frac{h}{36}\right)dt=-\frac{h^5}{5760}, \qquad (36)$$

получим ***каноническую формулу Симпсона*** с остаточным членом:

$$\int_{-h/2}^{h/2} f(x)dx=\frac{h}{6}\left(f\left(-\frac{h}{2}\right)+4f(0)+f\left(\frac{h}{2}\right)\right)-\frac{h^5}{2880}f^{(4)}(\xi). \qquad (37)$$

Как и для метода трапеций и прямоугольников (см. с. 42), для получения усложненной квадратурной формулы метода Симпсона разобьём отрезок интегрирования на N сегментов. Запишем каноническую формулу Симпсона применительно к отрезку $[x_{i-1}, x_i]$:

$$\int_{x_{i-1}}^{x_i} f(x)dx=\frac{h_i}{6}\left[f(x_{i-1})+4f\left(\frac{x_{i-1}+x_i}{2}\right)+f(x_i)\right]+\varepsilon(h_i).$$

Суммируя левую и правую части этого соотношения по i от 1 до N, получаем ***усложненную квадратурную формулу Симпсона***

$$\int_a^b f(x)dx=\frac{1}{6}\left(f(a)h_1+f(b)h_N+4\sum_{i=1}^{N}f\left(\frac{x_{i-1}+x_i}{2}\right)h_i+\right.$$

$$\left.+\sum_{i=1}^{N-1}f(x_i)(h_i+h_{i+1})\right)+O(h^4). \qquad (38)$$

Таким образом, метод Симпсона имеет четвертый порядок. В принципе, можно строить по той же идее, путем повышения степени интерполяционного полинома, и другие методы семейства Ньютона–Котеса (заметим, однако, что фактически порядок метода повышается только при четных степенях полинома [2], т.е. следующий после метода Симпсона порядок, шестой, будет у метода с интерполяцией по пяти точкам).

5.3. Метод Гаусса

В формулах семейства Ньютона–Котеса точки (узлы интерполяции) на каждом шаге расставлялись равномерно. Можно попробовать найти среди всех квадратурных формул с n узлами вида (20) квадратурную формулу с таким расположением узлов x_j на отрезке $[a, b]$ и с такими весами q_j, при которых она точна для алгебраических полиномов максимально возможной степени. Именно на такой идее основано семейство квадратурных формул Гаусса.

Заметим, что искомая максимальная степень полинома, для которой интегрирование будет точным, обязательно меньше, чем $2n$ (удвоенное число узлов интерполяции), так как при любом выборе узлов интерполяции x_j и весов q_j, $j=\overline{1,n}$, можно построить полином степени $2n$ вида

$$P_{2n}=(x-x_1)^2(x-x_2)^2\ldots(x-x_n)^2,$$

который обладает тем свойством, что

$$\int_a^b P_{2n}(x)\,dx>0,\quad \sum_{j=1}^{n} q_j P_{2n}(x_j)=0,$$

т.е. приближенное равенство (20) не может быть выполнено точно.

5.3.1 Интегрирование полинома с использованием базисных полиномов Лагранжа

Сначала покажем, как можно просто и точно посчитать интеграл от любого полинома степени $n-1$, вычисляя его значения в n произвольных точках.

Построим формулы интегрирования для отрезка $[-1, 1]$. Пусть $x_j\in[-1,1]$, $j=\overline{1,n}$ – произвольные различные узлы. Тогда, если взять веса

$$q_j=\int_{-1}^{1} p_j(x)\,dx,\quad j=\overline{1,n}, \tag{39}$$

где $p_j(x)$ – базисные полиномы Лагранжа (17) для интерполяционного полинома степени $n-1$), то квадратурная формула будет точна для полиномов степени $n-1$ (поскольку любой полином $P_{n-1}(x)$ степени $n-1$

совпадает с построенным для него интерполяционным полиномом $L_{n-1}(x)$ той же степени в силу теоремы о существовании и единственности интерполяционного полинома).

Следовательно, учитывая определение полинома Лагранжа (18), получим

$$\int_{-1}^{1} P_{n-1}(x)dx = \int_{-1}^{1} L_{n-1}(x)dx = \int_{-1}^{1} \sum_{j=1}^{n} p_j(x) P_{n-1}(x_j)dx =$$

$$= \sum_{j=1}^{n} \left(\int_{-1}^{1} p_j(x)dx \right) P_{n-1}(x_j) = \sum_{j=1}^{n} q_j P_{n-1}(x_j).$$

А это и означает, что квадратурная формула с такими параметрами точна для любых полиномов $n-1$ степени.

Для построения метода Гаусса мы выберем n точек некоторым специальным образом. Для этого нам понадобится несколько свойств полиномов Лежандра.

5.3.2 Полиномы Лежандра

Полиномом Лежандра называется полином, определяемый выражением

$$P_n(x) = \frac{1}{(2n)!!} \frac{d^n}{dx^n} \left(x^2 - 1\right)^n. \tag{40}$$

Полином Лежандра $P_n(x)$ является алгебраическим полиномом *n*-й степени со старшим коэффициентом, не равным нулю, так как при *n*-кратном дифференцировании полинома

$$\left(x^2 - 1\right)^n = x^{2n} - nx^{2n-2} + \ldots$$

степень понижается в точности на *n*.

Для полиномов Лежандра справедлива рекуррентная формула [4]

$$P_{n+1}(x) = \frac{(2n+1)\,x\,P_n(x) - n\,P_{n-1}(x)}{n+1},$$

по которой с учетом того, что $P_0 = 1,\ P_1 = x$, может быть найден полином Лежандра любой степени. В частности:

$$P_2(x) = \frac{1}{2}\left(3x^2 - 1\right),\ \ P_3(x) = \frac{1}{2}\left(5x^3 - 3x\right),\ \ P_4(x) = \frac{1}{8}\left(35x^4 - 30x^2 + 3\right), \ldots.$$

Теорема об ортогональности полиномов Лежандра. Полином Лежандра $P_n(x)$ ортогонален любому полиному $Q_m(x)$ степени $m<n$.

▸ Заметив предварительно, что выражение $\dfrac{d^k(x^2-1)^n}{dx^k}$ при $k=0,1,2,...,n-1$, обращается в нуль в точках $x=-1$ и $x=1$, имеем, последовательно интегрируя по частям:

$$(2n)!!\int_{-1}^{1} Q_m(x)P_n(x)dx=\int_{-1}^{1} Q_m(x)\frac{d^n(x^2-1)^n}{dx^n}dx=$$

$$=Q_m(x)\frac{d^{n-1}(x^2-1)^n}{dx^{n-1}}\bigg|_{-1}^{1}-\int_{-1}^{1} Q_m'(x)\frac{d^{n-1}(x^2-1)^n}{dx^{n-1}}dx=$$

$$=(-1)^m Q_m^{(m)}(x)\int_{-1}^{1}\frac{d^{n-m}(x^2-1)^n}{dx^{n-m}}dx=(-1)^m Q_m^{(m)}(x)\frac{d^{n-m-1}(x^2-1)^n}{dx^{n-m-1}}\bigg|_{-1}^{1}+0=0.\ ◂$$

Следствие. На отрезке $[-1;1]$ полиномы Лежандра обладают свойством ортогональности:

$$(P_n,P_m)=\int_{-1}^{1} P_n(x)P_m(x)dx=\begin{cases}0, & n\neq m,\\ \dfrac{2}{2n+1}, & n=m.\end{cases} \tag{41}$$

▸ Ортогональность следует непосредственно из теоремы.

Для вычисления нормы полиномов Лежандра также воспользуемся интегрированием по частям:

$$\int_{-1}^{1} P_n^2(x)dx=\int_{-1}^{1} P_n(x)\left[\frac{(2n-1)!!}{n!}x^n+Q_{n-1}(x)\right]dx=$$

$$=\frac{(2n-1)!!}{n!}\int_{-1}^{1} P_n(x)x^n dx=\frac{(2n-1)!!}{n!(2n)!!}\int_{-1}^{1}\frac{d^n(x^2-1)^n}{dx^n}x^n dx=...=$$

$$=(-1)^{n-1}\frac{(2n-1)!!}{(2n)!!}\int_{-1}^{1} x\,d(x^2-1)^n=(-1)^{n-1}\frac{(2n-1)!!}{(2n-2)!!}\int_{-1}^{1}(x^2-1)^{n-1}x^2dx=$$

$$=(-1)^{n-1}\frac{(2n-1)!!}{(2n-2)!!}\cdot\frac{1}{3}\int_{-1}^{1}(x^2-1)^{n-1}dx^3=$$

$$=(-1)^{n-1}\frac{(2n-1)!!}{(2n-4)!!}\cdot\frac{1}{3}\int_{-1}^{1}(x^2-1)^{n-2}x^4dx=...=\int_{-1}^{1}x^{2n}dx=\frac{2}{2n+1}.\ ◂$$

Теорема о корнях полиномов Лежандра. Все корни полинома Лежандра действительные, простые (однократные) и расположены в интервале $(-1, 1)$ симметрично относительно начала координат.

▸ Достаточно показать, что полином $P_n(x)$ в интервале $(-1, 1)$ меняет знак ровно n раз. Допустим противное, что полином $P_n(x)$ меняет знак в точках $\xi_1, \xi_2, ..., \xi_m$, где $m < n$. Тогда полином

$$Q_{m(x)} = (x-\xi_1)(x-\xi_2)...(x-\xi_m)$$

меняет знак в тех же точках. Следовательно, произведение $P_n(x)Q_m(x)$ сохраняет знак на $(-1, 1)$. Если это произведение неотрицательно, то

$$\int_{-1}^{1} P_n(x)Q_m(x) > 0 \qquad (42)$$

(равенства быть не может, так как $\exists\tilde{x}\ P_n(\tilde{x})Q_m(\tilde{x}) > 0$). Если же $P_n(x)Q_m(x) \le 0$, то интеграл (42) будет строго меньше нуля. С другой стороны, в силу ортогональности полиномов при $m < n$ этот интеграл должен быть равен нулю.

Таким образом, полином Лежандра $P_n(x)$ имеет порядок n и в n точках меняет знак, значит, у него нет ни комплексных корней, ни кратных корней. ◂

5.3.3 Каноническая формула метода Гаусса

Возьмем в качестве узлов x_j, $j = \overline{1,n}$ корни полинома Лежандра $P_n(x)$, которые согласно доказанной выше теореме все различны и расположены в интервале $[-1, 1]$, а веса q_j найдем по формуле (39).

Покажем, что квадратурная формула (20) с выбранными узлами и весами точна для полиномов степени $2n-1$ и тем самым для полиномов максимально возможной степени.

Пусть $Q_{2n-1}(x)$ – произвольный алгебраический полином степени $2n-1$. Очевидно, его можно представить в виде

$$Q_{2n-1}(x) = U_{n-1}(x)P_n(x) + V_{n-1}(x),$$

где $U_{n-1}(x)$ и $V_{n-1}(x)$ – полиномы степени $n-1$ (частное и остаток от деления полинома $Q_{2n-1}(x)$ на полином Лежандра $P_n(x)$).

В силу ортогональности на отрезке $[-1;1]$ полинома Лежандра $P_n(x)$ любому полиному степени $n-1$ справедливо:

$$\int_{-1}^{1} Q_{2n-1}(x)\,dx = \int_{-1}^{1} U_{n-1}(x)P_n(x)\,dx + \int_{-1}^{1} V_{n-1}(x)\,dx = \int_{-1}^{1} V_{n-1}(x)\,dx.$$

Квадратурная формула (20) с заданными весами q_j (39) заведомо точна для полиномов степени $n-1$, поэтому

$$\int_{-1}^{1} V_{n-1}(x)\,dx = \sum_{j=1}^{n} q_j V_{n-1}(x_j).$$

Учитывая, что $P_n(x_j) = 0$ (так как x_j – корни полинома Лежандра), получим

$$\sum_{j=1}^{n} q_j Q_{2n-1}(x_j) = \sum_{j=1}^{n} q_j U_{n-1}(x_j)P_n(x_j) + \sum_{j=1}^{n} q_j V_{n-1}(x_j) = \sum_{j=1}^{n} q_j V_{n-1}(x_j).$$

Следовательно,

$$\int_{-1}^{1} Q_{2n-1}(x)\,dx = \sum_{j=1}^{n} q_j Q_{2n-1}(x_j). \tag{43}$$

Таким образом, ***каноническая квадратурная формула Гаусса*** имеет вид

$$\int_{-1}^{1} f(x)\,dx \approx \sum_{j=1}^{n} q_j f(x_j), \tag{44}$$

где веса q_j определяются соотношением (39), а x_j – корни полинома Лежандра $P_n(x)$.

Обратим внимание, что в полученной формуле (44) отсутствует остаточный член. Это объясняется тем, что формула получена для шаб-

лонного отрезка длины 2, а не длины h, что делает бессмысленным само понятие остаточного члена для неё.

Можно доказать [4], что узлы x_j расположены симметрично относительной точки $x=0$, а веса q_j положительны и в симметричных узлах совпадают. Таким образом, для практической реализации квадратурных формул семейства Гаусса необходимо знать выражения корней полиномов Лежандра x_j и весов q_j (интегралов от базисных полиномов Лагранжа). Приведем их для первых пяти формул семейства:

n	Корни полинома Лежандра (узлы интерполяции x_j)	Интегралы от базисных полиномов Лагранжа (веса q_j)
1	0	2
2	$\pm 1/\sqrt{3}$	1
3	0	$8/9$
	$\pm\sqrt{3/5}$	$5/9$
4	$\pm\sqrt{\left(3-2\sqrt{6/5}\right)/7}$	$\dfrac{18+\sqrt{30}}{36}$
	$\pm\sqrt{\left(3+2\sqrt{6/5}\right)/7}$	$\dfrac{18-\sqrt{30}}{36}$
5	0	$128/255$
	$\pm\dfrac{1}{3}\sqrt{5-2\sqrt{10/7}}$	$\dfrac{322+13\sqrt{70}}{900}$
	$\pm\dfrac{1}{3}\sqrt{5+2\sqrt{10/7}}$	$\dfrac{322-13\sqrt{70}}{900}$

Как видно, при $n=1$ формула Гаусса совпадает с канонической формулой прямоугольников.

5.3.4 Усложненная квадратурная формула Гаусса

Усложненная квадратурная формула для метода Гаусса строится также, как и для формул Ньютона–Котеса. Разобьем отрезок $[a, b]$ на N произвольных частичных отрезков $[x_{k-1}, x_k]$, $k = \overline{1, N}$, $x_0 = a$, $x_N = b$, $h_k = x_k - x_{k-1}$. На каждом частичном отрезке зададим по n узлов (пропорционально пересчитав корни полинома Лежандра):

$$x_{k,j} = \frac{x_{k-1} + x_k}{2} + \hat{x}_j \frac{h_k}{2}, \ j = \overline{1,n}, \tag{45}$$

где $\hat{x}_j$ – узлы канонической формулы Гаусса (44). В силу этого квадратурная формула

$$\int_{x_{k-1}}^{x_k} f(x)dx = \left[\xi = 1 + \frac{2(x - x_k)}{h_k} \right] =$$

$$= \frac{h_k}{2} \int_{-1}^{1} f\left(\frac{x_{k-1} + x_k}{2} + \xi \frac{h_k}{2} \right) d\xi =$$

$$= \frac{h_k}{2} \left(\sum_{j=1}^{n} q_j f\left(x_{k,j}\right) \right) + \varepsilon(h_k) \tag{46}$$

также точна для полиномов степени $2n-1$ на отрезке $[x_{k-1}, x_k]$. Можно доказать, что остаточный член данной формулы имеет вид [2]

$$\varepsilon(h_k) = \frac{(h_k)^{2n+1} (n!)^4}{((2n)!)^3 (2n+1)} f^{(2n)}(\xi). \tag{47}$$

Суммируя соотношение (46) по k, получаем ***усложненную квадратурную формулу Гаусса***:

$$\int_a^b f(x)dx = \frac{1}{2} \sum_{j=1}^{n} q_j \sum_{k=1}^{N} h_k \cdot f\left(x_{k,j}\right) + O\left(h^{2n}\right). \tag{48}$$

Обратим внимание, что квадратурные формулы Гаусса имеют очень высокий порядок, но их сходимость доказана только для случая, когда функция имеет $2n+1$ непрерывную производную. Этим существенно ограничивается область применимости методов типа Гаусса.

5.4. Правило Рунге и поправка Ричардсона

Мы рассмотрели несколько идей построения квадратурных формул, однако так и не стало ясно, как на практике (когда истинное значение интеграла неизвестно) оценить, насколько точно решена задача численного интегрирования. Рассмотрим один из универсальных способов такой оценки – правило Рунге.

Пусть I^* – неизвестное точное значение некоторого интеграла, I_h – его приближенное значение, полученное на сетке с параметром h (см. с. 42), который может принимать сколь угодно малые значения. Запишем квадратурную формулу в виде

$$I^* = I_h + Ch^k + O\left(h^{k+m}\right), \tag{49}$$

где C – некоторая не зависящая от h постоянная; k – порядок малости остаточного члена квадратурной формулы; $m \geq 1$ – некоторое натуральное число, определяемое остаточным членом квадратурной формулы.

Тогда для сетки с уменьшенным вдвое шагом справедливо

$$I^* = I_{\frac{h}{2}} + C\left(\frac{h}{2}\right)^k + O\left(h^{k+m}\right). \tag{50}$$

Вычитая (49) из (50), получим

$$I_{\frac{h}{2}} - I_h = C\left(\frac{h}{2}\right)^k \left(2^k - 1\right) + O\left(h^{k+m}\right),$$

откуда

$$C\left(\frac{h}{2}\right)^k = \frac{I_{\frac{h}{2}} - I_h}{2^k - 1} + O\left(h^{k+m}\right). \tag{51}$$

Подставив (51) в (50), получим ***правило Рунге*** оценки погрешности численного интегрирования:

$$I^* - I_{\frac{h}{2}} = \frac{I_{\frac{h}{2}} - I_h}{2^k - 1} + O\left(h^{k+m}\right). \tag{52}$$

Заметим, что остаточный член в (52) имеет более высокий порядок, чем сама квадратурная формула. Это позволяет использовать полученный результат оценки погрешности для уточнения решения. Квадратурная формула

$$I^* = I_{\frac{h}{2}} + \frac{I_{\frac{h}{2}} - I_h}{2^k - 1} + O(h^{k+m}) \tag{53}$$

имеет, очевидно, более высокий порядок малости остаточного члена, чем исходная формула (50). Такой способ повышения точности численного интегрирования называется ***поправкой Ричардсона***.

Обратим внимание, что правило Рунге – это лишь *приближенная* оценка погрешности (хотя порядок малости погрешности этой оценки выше, чем порядок малости остаточного члена самого метода интегрирования). Фактически это означает, что оценка погрешности по правилу Рунге, как и сам метод численного интегрирования, начинает работать только при достаточно малых шагах, при больших же h влияние остаточного члена (который мы записали в виде $O\left(h^{k+m}\right)$) может оказаться существенным и оценка (49) окажется практически неверной. Кроме того, оценка погрешности, как и сами квадратурные формулы, зависят от гладкости интегрируемой функции (поскольку фактически в $O\left(h^{k+m}\right)$ скрывается производная функции).

Необходимо еще отметить, что при слишком малом шаге h на точность оценки начинает влиять конечность используемой для вычислений арифметики, и оценка (52) вследствие этого может показать погрешность численного интегрирования существенно меньшую, чем есть на самом деле (просто потому, что значения I^h и $I_{\frac{h}{2}}$ стали слишком мало различаться из-за конечности используемой для вычислений арифметики).

Кроме того, возможен случай, когда для интегрируемой функции константа C в (49) равна нулю (как видно из приведённых в этом пособии квадратурных формул, константа C определяется именно интегрируемой функцией). В этом случае правило Рунге даст неверную оценку просто потому, что порядок малости остаточного члена k определён неверно для конкретной интегрируемой функции.

На практике для оценки возможности применения правила Рунге и поправки Ричардсона часто используют эмпирическое соотношение

$$\left| 1 - \frac{1}{2^k} \cdot \frac{I_{2h} - I_{\frac{h}{2}}}{I_h - I_{\frac{h}{2}}} \right| < 0.1 \tag{54}$$

Фактически это соотношение проверяет, что на трех вложенных сетках погрешность падает в соответствии с порядком метода хотя бы в первой значащей цифре.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Писсанецки С. Технология разреженных матриц. – М.: Мир, 1988.

2. Вержбицкий В.М. Численные методы. – М.: Высшая школа, 2001.

3. Демидович Б.П., Марон И.А. Основы вычислительной математики. – М.: Наука, 1966.

4. Суетин П.К. Классические ортогональные многочлены. – М.: Наука, 1979.

ПРЕДМЕТНЫЙ УКАЗАТЕЛЬ