СОДЕРЖАНИЕ

ПРЕДИСЛОВИЕ

Настоящее пособие отражает опыт автора в преподавании дисциплины «Вычислительная математика» для студентов, обучающихся по специальности «Системы автоматизированного проектирования» (САПР) в Томском государственном университете автоматизированных систем управления и радиоэлектроники. Техническая направленность специальности определила характер пособия – здесь основные положения и алгоритмы вычислительной математики часто иллюстрируются примерами, меньшее внимание уделяется строгому доказательству положений (например, в виде теорем).

В первой части пособия рассматриваются следующие темы:

- введение в численные методы;
- погрешности вычислений;
- вычисление элементарных функций.

В первой главе, являющейся вводной, даются понятия об основных этапах решения задач, характеристиках вычислительных задач и алгоритмов.

Во второй главе рассматриваются источники и причины погрешностей при вычислениях на ЭВМ. Изложение материала иллюстрируется примерами, основанными на десятичной арифметике (виртуальная ЭВМ), которые, однако, раскрывают механизм появления ошибок и в двоичной арифметике.

Третья глава пособия посвящена способам вычисления элементарных функций.

1. ВВЕДЕНИЕ В ЧИСЛЕННЫЕ МЕТОДЫ

1.1. Предмет вычислительной математики

Применение ЭВМ приобрело в настоящие время массовый характер. Вычислительная техника используется не только в инженерных и экономических науках, но и в таких «нематематических» специальностях, как медицина, лингвистика, психология и др. Круг специалистов – пользователей, встречающихся с вычислениями на ЭВМ, – весьма широк. Поэтому возникла необходимость в специальных дисциплинах, рассматривающих применение компьютеров для вычислений.

Основной такой дисциплиной является ***вычислительная математика***. Она изучает способы построения и исследования *численных* методов решения математических задач на ЭВМ. Для обозначения этой дисциплины используются также синонимы «прикладная математика», «вычислительные методы», «численные методы».

Численные методы разрабатывают и исследуют, как правило, высококвалифицированные специалисты – математики. Эффективные вычислительные алгоритмы реализованы в составе многих математических пакетов, а также в виде отдельных программ на алгоритмических языках. Что касается основной части студентов нематематических специальностей и инженеро-технических работников, то для них главной задачей является понимание основных идей методов, особенностей и областей их применения. Это позволяет осознанно применять имеющиеся пакеты и библиотеки вычислительных программ.

Таким образом, изучение настоящей дисциплины должно позволить студентам оценить *свойства методов* (такие, например, как точность, быстродействие, надежность и др.) и выбрать *наиболее подходящий численный метод* (или совокупность методов) для решения поставленной задачи.

1.2. Классическая и вычислительная математика

Можно выделить следующие группы математических методов, используемых для вычислений:

- качественные;
- графические;
- аналитические;
- численные.

Качественные методы обычно основаны на теоремах, устанавливающих некоторые важные свойства решаемых задач.

Примеры:

а) основная теорема алгебры, устанавливающая число корней алгебраического уравнения *n*-й степени;

б) теоремы, определяющие условия существования решения системы линейных алгебраических уравнений.

Качественные методы помогают выявить свойства и особенности вычислительных задач, но, как правило, не дают непосредственно их решения.

Графические методы основаны на графических построениях и обычно служат для выявления свойств и приближенной оценки решений задачи.

Пример: решение уравнения $f(x)= 0$ путем нахождения точек пересечения графика функции $f(x)$ с осью абсцисс. Такой способ позволяет оценить число и приближенные значения вещественных корней уравнения.

Графические методы наглядны, но грубы, они могут быть применены к сравнительно простым задачам. Следует отметить, что в связи с появлением в современных математических пакетах развитых средств машинной графики трудоемкость реализации графических методов существенно уменьшилась.

Аналитические методы предполагают представление решения задачи в виде конечного числа *формул*. Помимо *чисел*, они оперируют с *функциями, операторами* и т.д.

Для *классической математики* характерно использование качественных и аналитических методов вычислений. В настоящие время возможности аналитических методов решения задач существенно расширились благодаря появлению пакетов символьных вычислений (компьютерной алгебры), таких как Matematica, Maple V, Derive и др. К сожалению, даже применения всего арсенала методов классической математики недостаточно для решения многих практических задач.

Приведем **примеры.**

1. Вычисление определенных интегралов. Как известно, для вычисления интегралов существуют таблицы, разработаны также специальные приемы. Однако есть функции (например, e^{x^2}), интегралы от которых в принципе существуют, но не могут быть выражены в аналитическом виде.

2. Нахождение корней полинома *n*-й степени (решение алгебраического уравнения с одним неизвестным):

$$a_n x^n + a_{n-1} x^{n-1} + \ldots + a_1 x + a_0 = 0. \qquad (1.1)$$

Формула для корней квадратного уравнения (n=2), которая приводится в школьном курсе математики, была известна еще в XII веке. В XVI веке были найдены аналитические выражения для корней уравнений 3-й и 4-й

степени (формулы Кардано и Феррари). Однако дальнейшие попытки математиков получить аналогичные формулы для корней уравнений 5-й степени и выше не увенчались успехом. Только в 1824 г. норвежский математик Н.Абель доказал, что таких формул в принципе не существует, т.е. в общем случае корни алгебраического уравнения с $n \geq 5$ не могут быть выражены с использованием обычных арифметических операций. Таким образом, аналитически решить алгебраические уравнения выше 4-й степени не представляется возможным!

3. Решение систем линейных уравнений. Изучаемые в курсе линейной алгебры методы Крамера и Гаусса являются аналитическими и теоретически пригодны для систем любого порядка. Однако попытка непосредственного воплощения этих методов в вычислительных процедурах для ЭВМ, без учета влияния погрешностей вычислений, приведет к неэффективным алгоритмам, которые могут давать неверные результаты уже при числе уравнений $n=20\div40$!

Таким образом, непосредственное использование результатов классической математики для вычислений часто является неэффективным и не приводит к цели. Поэтому появилась потребность в специальной дисциплине - *вычислительной математике*, позволяющей построить более эффективные методы и расширить класс решаемых задач. Если классическая математика не рассматривает вычисления как самоцель, то в вычислительной математике акцент делается именно на практической стороне вычислений. Вычислительная математика изучает численные методы, *специально* предназначенные для *эффективных* вычислений, при этом рассматриваются такие характеристики, как точность и быстродействие метода, вычислительная сложность задачи и т.д. Здесь предлагаются модификации известных (классических) методов и новые методы вычислений, рассматриваются особенности реализации вычислительных алгоритмов на ЭВМ, вплоть до особенностей их программирования, и т.д.

Численные методы – это методы, основанные на сведении решения задачи к выполнению конечного числа *арифметических действий над числами* (сложение, вычитание, умножение, деление). Эти методы оперируют *только с числами*. Иначе говоря, численные методы – это такие методы, которые могут быть *непосредственно* реализованы на *цифровой* ЭВМ.

При практических вычислениях численные методы во многих случаях имеют преимущества перед аналитическими методами. Продемонстрируем это на рассмотренных выше **примерах.**

1. При численном интегрировании определенный интеграл интерпретируется как площадь под кривой $f(x)$, а сама площадь находится как сумма площадей S_i некоторых элементарных фигур (прямоугольников, трапеций и т.д.):

$$\int_a^b f(x)dx \approx \sum_{i=1}^{m} S_i , \qquad (1.2)$$

где m – число элементарных фигур. В отличие от аналитического способа вычисления интеграла, формула (1.2) пригодна для любых функций $f(x)$. Она может быть применена даже тогда, когда функция $f(x)$ задана не аналитически, а таблицей значений или в форме графика.

2. В вычислительной математике разработаны эффективные алгоритмы, которые позволяют найти численные значения корней полинома (1.1) степени n=50÷100 и выше.

3. Небольшая модификация метода Гаусса, учитывающая механизм образования ошибок при вычислениях на ЭВМ, позволяет увеличить порядок надежно решаемых систем линейных уравнений в несколько раз (примерно до значений n=100÷200) по сравнению с классическим методом. Специальные итерационные численные методы дают возможность решать системы линейных уравнений с числом неизвестных $n=10^5 \div 10^6$!

Приведенные примеры (а их можно дать гораздо больше) ясно показывают большие возможности вычислительной математики.

1.3. Математическое моделирование и вычислительный эксперимент

Появление и совершенствование ЭВМ привело к подлинно революционному преобразованию науки и техники. Изменилась технология исследования, колоссально увеличились возможности теоретического изучения и прогноза сложных процессов, проектирования сложных технических систем и т.д. Решение крупных научно-технических проблем стало возможным лишь благодаря применению математического моделирования и новых численных методов для ЭВМ.

Приведем некоторые **примеры таких проблем**.

1. Овладение ядерной энергией и построение атомных реакторов требует решения комплекса сложных задач физики и механики (проектирование и управление работой реактора, изучение тепловых полей и напряжения в стенках, расчеты на прочность, защита от излучения и т.д.).

2. Освоение космоса и создание летательных аппаратов (ракет, самолетов, вертолетов) связано с решением многих задач аэродинамики и баллистики (например, расчет движения ракеты и управление ее полетом), а также комплекса сложных задач механики, физики и техники.

3. Прогнозирование погоды ведет к решению сложных физических и метеорологических задач с большим числом влияющих факторов.

Применение при исследовании подобных проблем физического моделирования (например, с использованием макетов реальных объектов) обычно является трудоемким, дорогостоящим, а иногда и вообще невозможным. Поэтому в настоящие время для решения сложных задач выработалась новая технология – вычислительный эксперимент.

Вычислительный эксперимент – это технология решения сложных проблем, исследования физических и технических объектов и процессов, основанная на построении и анализе с помощью ЭВМ математических моделей объектов и процессов.

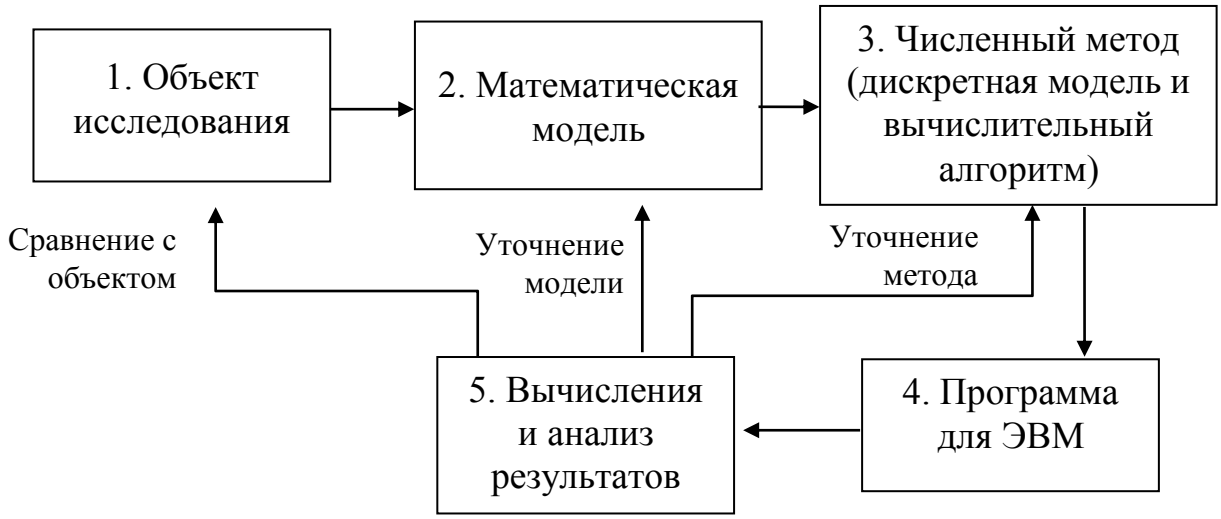

Рис. 1.1. **Схема вычислительного эксперимента**

Схема вычислительного эксперимента при исследовании некоторого объекта, явления или процесса приведена на рис. 1.1. Рассмотрим основные этапы такого эксперимента.

На первом этапе формулируется задача исследования в содержательной (физической) постановке. Здесь выбирается физическая модель объекта, устанавливаются физические законы, которым он подчиняется, определяется, какие свойства объекта нужно исследовать, какие физические факторы необходимо учесть, а какими можно пренебречь.

На втором этапе физической модели ставится в соответствие математическая модель объекта.

Математическая модель физического или технического объекта – это совокупность математических величин (чисел, переменных, матриц, множеств и т.д.) и отношений между ними, отражающая с необходимой точностью интересующие пользователя свойства объекта.

В качестве отношений могут выступать уравнения, неравенства, включения одних множеств в другие и т.д.

Очевидно, математическая модель должна правильно (адекватно) описывать физический объект. Обычно такая модель представляет собой запись физических законов (например, сохранения массы, энергии и т.д.) в виде систем уравнений – алгебраических, дифференциальных, интегральных и др. Большинство реальных объектов и процессов описывается *нелинейными* уравнениями. В некоторых условиях (при малых значениях параметров, малых отклонениях от состояния равновесия и т.д.) эти условия можно заменить линейными.

На третьем этапе строится численный метод решения задачи. Поскольку ЭВМ способна выполнять лишь простейшие арифметические операции, она «не понимает» постановки задачи даже в математической формулировке. Для ее решения должен быть найден численный метод, позволяющий свести задачу к некоторому вычислительному алгоритму, реализуемому на ЭВМ.

Обычно построение численного метода для заданной математической модели включает два шага:

а) переход к дискретной модели объекта (***дискретизация*** математической задачи);

б) выбор или разработка вычислительного алгоритма для дискретной задачи.

Полученное решение дискретной задачи принимается за приближенное решение исходной математической задачи.

Под ***дискретной моделью*** понимается такая интерпретация математической модели, которая доступна для реализации на ЭВМ. Чаще всего дискретизация достигается переходом от функции непрерывного аргумента к функции дискретного аргумента (аргумент принимает дискретное множество значений). Например, если математическая модель представляет собой систему дифференциальных уравнений, то дискретной моделью может быть аппроксимирующая ее система разностных уравнений. В последней системе производные каждой функции заменены отношениями конечного приращения функции к конечному приращению аргумента, в связи с чем уравнения переходят в алгебраические. Далее подбирается вычислительный алгоритм для решения системы разностных уравнений.

На четвертом этапе алгоритм реализуется в виде программы для ЭВМ. Мы не будем останавливаться на вопросах, связанных с программированием, так как это выходит за рамки данного пособия. Отметим лишь, что разработка конкретных численных алгоритмов и их программирование должны быть тесно связаны. Необходимыми составляющими разработки программы являются ее отладка и тестирование.

На последнем*, пятом этапе* проводятся вычисления на ЭВМ, и результаты сопоставляются с реальным объектом. Такое сопоставление позволяет судить о корректности проведенного вычислительного эксперимента и вносит в процесс вычислений «обратную связь». В случае несовпадения результатов вычислений с поведением объекта исследователь принимает решение о внесении исправлений и уточнений в программу, численный метод и математическую модель.

1.4. Численный метод, алгоритм и программа

Основу вычислительного алгоритма составляет следующая цепочка (триада):

Модель
↓
Численный метод
(алгоритм)
↓
Программа
(вычисления)

Рассмотрим более подробно отличие между понятиями «численный метод», «алгоритм» и «программа». Определение численного метода было дано ранее в п.1.2. Приведем определения алгоритма и программы.

Вычислительный алгоритм – последовательность арифметических и логических операций, при помощи которых находится приближенное численное решение задачи.

Программа – запись алгоритма решения задачи на понятном ЭВМ языке в виде точно определенной последовательности операций.

Сравнение определений для численного метода и алгоритма показывает, что эти понятия очень близки. Отличия между ними весьма тонкие и состоят больше в традициях использования данных терминов.

Обычно под ***численным методом*** понимают некоторую последовательность *алгоритмических блоков* (часто типовых) для решения поставленной задачи. Иначе говоря, задать численный метод – это значит определить необходимый набор «кубиков» - блоков, обеспечивающих получение решения. При этом сами «кубики», как правило, рассматриваются со степенью подробности, необходимой для понимания только идеи метода, или вообще не детализируются, если они типовые.

Например, при решении системы нелинейных уравнений методом Ньютона типовым «кубиком» является блок решения системы линейных алгебраических уравнений (СЛАУ). При описании метода способ решения системы линейных уравнений либо указывается без деталей (например, метод Гаусса), либо вообще не конкретизируется.

Очевидно, выбор численного метода существенно зависит от вида математической модели объекта.

Напротив, *алгоритм детально и однозначно определяет все вычислительные операции*. Можно считать, что ***алгоритм – это конкретная реализация численного метода.*** Один и тот же метод может быть реализован различными алгоритмами. При неудачном выборе алгоритма (т.е. деталей) можно «испортить» численный метод. Например, несмотря на потенциальную эффективность метода Ньютона, алгоритм решения системы нелиней-

ных уравнений будет весьма неэффективным, если в блоке решения СЛАУ применить метод Гаусса без учета влияния погрешностей вычислений.

Иногда понятия «численный метод» и «алгоритм» используются как синонимы, и между ними не делается различий. Зачастую сложный алгоритм описывают не очень подробно, с выделением недетализируемых (типовых) блоков. Все же, строго говоря, алгоритм – это описание процесса вычислений во всех подробностях.

Отличие понятий «алгоритм» и «программа» более выраженное. Алгоритм – это указание последовательности операций *безотносительно средств его реализации* (машинного или алгоритмического языка, транслятора, ЭВМ). Программа – это *запись* алгоритма на *конкретном языке*.

Точно так же, как алгоритм – это конкретная реализация численного метода, так и ***программа - это конкретная реализация алгоритма***. Один и тот же алгоритм можно осуществить различными программами. Например, можно использовать или не использовать подпрограммы, процедуры и функции, применить различные операторы цикла, выделить динамическую или статическую память, употребить разные языки программирования, учесть или не учесть особенности конкретной ЭВМ и т.д. От всех этих факторов будет зависеть эффективность программы. Неудачная реализация программы может «испортить» и алгоритм, и в целом численный метод.

Вывод настоящего раздела состоит в том, что математическая модель (исходные математические соотношения), численный метод (общая идея вычислений), алгоритм (детализация идеи) и программа (реализация на определенном языке) должны быть *согласованы* между собой. Численный метод должен учитывать особенности модели, алгоритм – особенности метода, а программа – особенности алгоритма. Численный метод не должен существенно уменьшать точность модели. Конкретное осуществление алгоритма и программы не должно ухудшить потенциальных свойств численного метода. Вычислительная математика и дает знания, необходимые для достижения такой согласованности.

1.5. Погрешности вычислительного эксперимента

Процесс исследования объекта методом математического моделирования и вычислительного эксперимента неизбежно имеет приближенный характер, потому что каждая составляющая в цепочке *модель* $\rightarrow$ *численный метод* (*алгоритм*) $\rightarrow$ *программа* (*вычисления*) вносит свою долю погрешностей. Таким образом, **полная (суммарная) погрешность вычислительного эксперимента включает в себя:**

1) погрешность математической модели;
2) погрешность численного метода;
3) вычислительную погрешность.

Рассмотрим более подробно источники возникновения этих погрешностей.

Погрешность математической модели имеет несколько причин. *Во-первых*, в модели, как правило, вынужденно или преднамеренно учитываются не все влияющие физические факторы. Общепринятой является практика, когда сложность модели выбирается из компромисса между точностью и скоростью вычислений. Таким образом, иногда модель специально упрощают для повышения быстродействия.

Во-вторых, исходные данные задачи обычно известны неточно, к ним относятся численные коэффициенты модели, начальные условия и другие входные данные. На практике многие параметры физических и технических объектов и процессов носят случайный характер, поэтому конкретные образцы одного и того же объекта (процесса) неизбежно будут иметь некоторые отличия. Все это ведет к неточности модели.

Погрешность вычислительного эксперимента, обусловленную погрешностью математической модели, называют *неустранимой*, поскольку она неизбежна в рамках данной модели. Несмотря на то, что эту погрешность нельзя исключить, следует *оценить ее влияние* на результирующую ошибку вычислений и в соответствии с этим *выбрать численный метод*, устойчивый к ошибкам в исходных данных.

Погрешность численного метода **(*методическая погрешность*)** обусловлена тем, что, как правило, численный метод воспроизводит исходную математическую модель приближенно. Ошибки могут возникать при переходе от точных формул к приближенным, от непрерывных функций к дискретным и т.д. Источниками первого вида ошибок являются, например, замена (аппроксимация) функции или ее табличных значений полиномом; замена бесконечного ряда, описывающего некоторую функцию, конечным, и т.д.

Специально выделяют ***ошибки дискретизации***, возникающие, в частности, при замене производной разностным отношением, замене определенного интеграла суммой и др. Например, используя вместо точного значения производной $f'(x)$ разностное отношение

$$f'(x) \approx \frac{f(x+\Delta x) - f(x)}{\Delta x},$$

мы допускаем погрешность дискретизации, имеющую при $\Delta x \to 0$ порядок Δx.

В зависимости от наличия методической погрешности различают *два вида* численных методов. Существуют методы, которые в предположении, что исходные данные точны, ошибки округления (см. ниже) отсутствуют и вычислительный процесс конечен (т.е. реализуется за конечное число элементарных арифметических операций), приводят к точным результатам. Такие численные методы называются ***прямыми точными методами***. Примерами являются методы Гаусса, Крамера и обратной матрицы для ре-

шения СЛАУ, нахождение корней уравнений 2-й – 4-й степени по аналитическим формулам и др.

Численные методы, которые даже при отсутствии ошибок в исходной информации и ошибок округления приводят к погрешности вычислений (независимо от того, конечен вычислительный процесс или бесконечен), называются ***приближенными методами***. К ним относятся, например, методы, основанные на дискретизации исходной математической задачи, итерационные численные методы и т.д.

Важно указать, что, как правило, погрешность приближенного численного метода *регулируема*, т.е. она может быть уменьшена до любого разумного значения путем изменения некоторого параметра (например, шага интегрирования, числа членов усеченного ряда и т.д.).

Вычислительная погрешность – это погрешность, возникающая из-за округления чисел в ЭВМ. Таким образом, этот вид погрешностей возникает непосредственно в процессе вычислений на ЭВМ. Из-за ограниченной разрядной сетки числа в ЭВМ округляются, т.е. представляются приближенно. В ограниченной сетке невозможно точно представить многие рациональные числа (т.е. натуральные дроби, например, 1/3), константы π, e и т.д.

При решении вычислительных задач выполняются миллионы и миллиарды операций с приближенными числами, при этом ошибки округления накапливаются. В результате решение, полученное на ЭВМ, будет отличаться от точного решения даже при отсутствии ошибок в исходных данных и методических ошибок. Величина вычислительной погрешности определяется двумя факторами: точностью представления вещественных чисел в ЭВМ и чувствительностью (устойчивостью) используемого алгоритма к погрешностям округления.

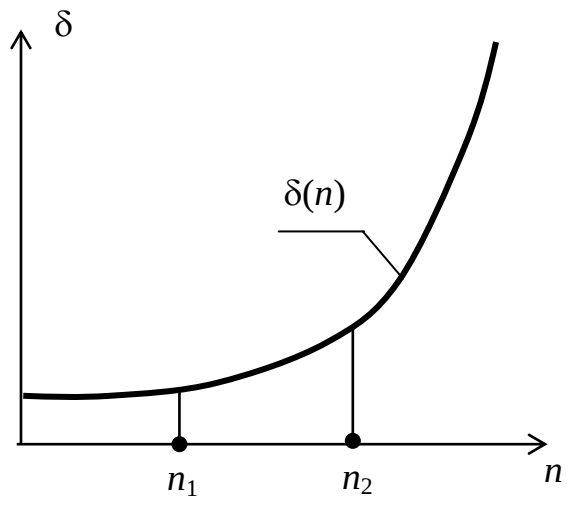

Рис. 1.2. **Зависимость вычислительной погрешности δ от порядка сложности задачи *n***

Обычно зависимость вычислительной погрешности δ некоторого алгоритма от порядка сложности задачи n (например, от числа уравнений и неизвестных) имеет характерный вид (рис. 1.2). При увеличении порядка сложности до некоторого значения n_1 погрешность мала и незначительно изменяется. При превышении другого значения n_2 она начинает резко возрастать, приводя к неправдоподобным результатам. Поэтому использовать данный алгоритм при порядке сложно-

сти задачи выше граничного значения n_2 нецелесообразно, нужно переходить к другому алгоритму.

Сказанное можно иллюстрировать на примере метода Гаусса, используемого для решения СЛАУ. В методе Гаусса методическая погрешность отсутствует, так как при точных исходных данных и отсутствии ошибок округления он дает точное решение СЛАУ. Таким образом, единственный источник погрешности здесь – это ошибки округления. При реализации классического варианта метода Гаусса возрастание вычислительной погрешности может наблюдаться уже при порядках СЛАУ $n=n_2=20\div40$. Для модификации метода Гаусса с выбором главного элемента, учитывающей влияние ошибок округления, порядок надежно решаемых СЛАУ увеличивается до $n_2=100\div200$. Для решения СЛАУ гораздо больших порядков (например, с $n>1000$) метод Гаусса непригоден, необходимо использовать другие алгоритмы.

Итак, следует различать ***погрешности модели, метода*** и ***вычислительную погрешность***. Рассмотрим вопрос о целесообразном выборе величин и соотношений этих погрешностей. На практике обычно считается достаточной точность моделей физических и технических объектов от долей процента до нескольких процентов. Погрешности исходных данных, если они получены путем эксперимента (измерений), имеют следующий порядок:

– точность прецизионных физических измерений – до 10^{-10} %;

– типичная точность астрономических и геофизических измерений – 10^{-4} %;

– точность измерения величин, характеризующих многие физические и технические объекты и процессы – (0,1÷10) %.

Последняя цифра и определяет достаточную точность модели во многих случаях.

В литературе приводятся следующие ***рекомендации по выбору соотношения между составляющими общей ошибки вычислений***:

1. Погрешность численного метода должна быть существенно меньше погрешности модели (рекомендуется взять ее в 2÷5 раз меньше). Дальнейшее повышение точности численного метода нецелесообразно, так как оно связано с ухудшением других характеристик метода (быстродействие, объем требуемой оперативной памяти и т.д.).

2. Погрешность округления должна быть заметно меньше или хотя бы сравнима с погрешностью метода. Если берутся «устойчивые» алгоритмы и порядок сложности задачи не превышает критической величины, это условие обычно выполняется.

Проиллюстрируем сказанное на примере задачи моделирования радиоэлектронных устройств. Точность измерения токов и напряжений, которые обычно интересуют разработчика, составляет (0,1÷3) %. Точность измерения параметров элементов радиоэлектронных устройств (резисторов,

конденсаторов, транзисторов, микросхем и т.д.) лежит в этих же пределах. Однако разброс самих параметров элементов весьма значителен и может достигать, например, для резисторов (5÷10) %, а для активных элементов (транзисторов, микросхем) – даже (30÷50) %. Поэтому при моделировании радиоэлектронных устройств нет смысла использовать очень точные (соответственно весьма сложные) модели элементов и очень точные алгоритмы. В применяемых на практике программах моделирования точность моделей элементов и точность численных методов лежат в пределах (0,1÷1) %.

1.6. Характеристики вычислительных задач

По своему содержанию вычислительные задачи могут быть самыми разнообразными. Однако, с точки зрения решения их на ЭВМ, они обладают некоторыми общими свойствами, которые необходимо учитывать при выборе или построении численного метода. Рассмотрим эти свойства.

1.6.1. Устойчивые и неустойчивые задачи

Как уже отмечалось, погрешности исходных данных являются неустранимыми, т.е. вычислитель не может с ними бороться. Однако нужно иметь представление об их влиянии на точность конечных результатов. Казалось бы, можно надеяться на то, что погрешность результатов имеет порядок погрешности исходных данных. К сожалению, это не всегда так. Некоторые задачи весьма чувствительны к погрешностям в исходных данных. Эта чувствительность характеризуется так называемой устойчивостью.

Пусть в результате решения задачи по исходному значению величины x определяется значение искомой величины y. Это можно обозначить как $x \to y$. Если исходная величина имеет абсолютную погрешность Δx, то решение имеет погрешность Δy, т.е. $x+\Delta x \to y+\Delta y$.

Задача называется ***устойчивой*** по исходному параметру x, если решение непрерывно зависит от x, т.е. малое приращение исходной величины Δx приводит к малому приращению искомой величины Δy. Иначе говоря, малые погрешности в исходной величине приводят к малым погрешностям в результате.

Если провести аналогию с классической математикой, то величину y можно считать некоторой функцией аргумента x. Тогда понятие устойчивости близко к понятию непрерывности функции $y(x)$.

В случае нескольких искомых величин и нескольких исходных (входных) параметров суть понятия устойчивости такая же.

Если задача ***неустойчива***, тогда даже незначительные погрешности в исходных данных приводят к большим погрешностям в решении или вовсе к неверным результатам. О таких задачах также говорят, что они сверхчувствительны к ошибкам в исходных данных, плохо обусловлены.

Заметим, что в данном случае природа погрешностей в исходных данных не имеет значения. Эти погрешности могут быть вызваны как ошибками измерения (эксперимента), так и приближенным представлением чисел в ЭВМ за счет округления. Результат одинаково реагирует на указанные источники ошибок. Поэтому при исследовании точности вычислений на ЭВМ часто используют обычную теорию погрешностей, применяемую, например, для анализа точности эксперимента.

Таким образом, в неустойчивых задачах ошибки в исходных данных и ошибки округления могут привести к катастрофической потере точности результата. Важно отметить, что в рассматриваемом случае устойчивость – это *внутреннее свойство самой задачи*, а не метода решения.

Приведем примеры устойчивых и неустойчивых задач.

Пример 1.1. Вычисление корней полинома (пример предложен известным английским специалистом по численным методам Д. Уилкинсоном).

Рассмотрим полином

$$P(x) = (x–1)\,(x–2)\ldots(x–20) = x^{20} – 210x^{19} + \ldots$$

Очевидно, что корнями этого полинома являются $x_1 = 1$, $x_2 = 2$, … , $x_{20} = 20$.

Предположим, что один из коэффициентов полинома задан (вычислен) с некоторой малой погрешностью. Например, вместо коэффициента $a_{19} = –210$ при x^{19} возьмем коэффициент $a'_{19} = –210 + 10^{–7}$. При этом относительная ошибка задания этого коэффициента составляет около $0{,}5\cdot10^{–7}$ %.

Вычислим теперь корни полинома с измененным коэффициентом. Они существенно изменятся по сравнению с исходным полиномом:

$x_1 = 1{,}00$, $x_9 = 8{,}92$,
$x_2 = 2{,}00$, $x_{10,\,11} = 10{,}1 \pm 0{,}644$ i,
$x_3 = 3{,}00$, $x_{12,\,13} = 11{,}8 \pm 1{,}65$ i,
$x_4 = 4{,}00$, $x_{14,\,15} = 14{,}0 \pm 2{,}52$ i,
$x_5 = 5{,}00$, $x_{16,\,17} = 16{,}7 \pm 2{,}81$ i,
$x_6 = 6{,}00$, $x_{18,\,19} = 19{,}5 \pm 1{,}94$ i,
$x_7 = 7{,}00$, $x_{20} = 20{,}8$.
$x_8 = 8{,}01$,

Относительная ошибка вычисления корня x_9 равна примерно 1%, а корня x_{20} – 4%. Таким образом, при ошибке в исходных данных (всего в одном коэффициенте полинома) $0{,}5\cdot10^{–7}$ % ошибка результата составила 4%, т.е. возросла почти в 10^8 раз! Кроме того, результат неверен даже качественно, так как появились комплексные корни, отсутствующие у исходного полинома.

Заметим, что в данном случае вычисления корней велись на ЭВМ с высокой точностью (с 11 десятичными разрядами) для того, чтобы исклю-

чить влияние ошибок округления. Абсолютная ошибка 10^{-7} в коэффициенте полинома может имитировать погрешность, возникающую за счет округления в ЭВМ с 7 десятичными разрядами.

Можно сделать вывод, что зависимость корней от коэффициентов полинома – «плохая», очень чувствительная функция. Малое изменение в коэффициентах ведет к большим изменениям корней, это внутреннее свойство самой задачи.

Таким образом, задача вычисления корней полинома является неустойчивой, плохо обусловленной. Уменьшить погрешность вычисления корней можно, только уменьшая ошибки задания исходных коэффициентов полинома и ошибки округления (используя большее число разрядов для представления чисел).

Пример 1.2. Задача интегрирования.

Покажем, что задача вычисления определенного интеграла является устойчивой. Пусть необходимо найти интеграл

$$I = \int_0^1 f(x)dx,$$

в этом случае исходной является функция $f(x)$, а результатом – значение интеграла I. Предположим, что исходные данные заданы с некоторой погрешностью $\delta f(x)$, т.е. вместо функции $f(x)$ фигурирует функция $\tilde{f}(x) = f(x) + \delta f(x)$. Погрешность в задании $f(x)$ приведет к тому, что значение интеграла изменится:

$$\tilde{I} = \int_0^1 \tilde{f}(x)dx.$$

Рассмотрим разность $\delta I = \tilde{I} - I = \int_0^1 \left[\tilde{f}(x) - f(x)\right]dx = \int_0^1 \delta f(x)dx.$

В силу известного математического неравенства $|\delta I| \le \max_{x\in[0;1]} |\delta f(x)|$.

Таким образом, если погрешность задания $f(x)$ не превышает некоторого значения ε, т.е. $|\delta f(x)| \le \varepsilon$, то и погрешность вычисления интеграла не превышает этого же значения: $|\delta I| \le \varepsilon$. Итак, интеграл I непрерывно зависит от функции $f(x)$ и, следовательно, задача интегрирования устойчива.

Таким же способом можно показать, что устойчивой является и задача численного интегрирования, когда интеграл I представляется в виде суммы площадей элементарных фигур. Устойчивость интегрирования является следствием того, что площадь под кривой $f(x)$ изменяется мало, если малы изменения функции $f(x)$.

Пример 1.3. Задача дифференцирования.

Задача дифференцирования функции $f(x)$, заданной приближенно, является неустойчивой. Покажем это на примере функции погрешности одно-

го частного вида. Допустим, что функция погрешности равна $\delta f(x) = \frac{1}{N}\sin N^2 x$, где N достаточно велико. Тогда вместо функции $f(x)$ рассматривается функция $\tilde{f}(x) = f(x) + \delta f(x) = f(x) + \frac{1}{N}\sin N^2 x$. Функция $\tilde{f}(x)$ представляет собой исходную функцию $f(x)$, на которую наложены синусоидальные колебания с амплитудой $1/N$ и периодом $T = 2\pi/N^2$ (рис. 1.3). Рассмотрим разность функций $\tilde{f}(x)$ и $f(x)$ на интервале $x \in [0; a]$, длина которого больше половины периода, т.е. $a > \pi/N^2$. Очевидно, максимальная величина этой разности не превышает значения $\varepsilon = 1/N$ и при большом N весьма мала.

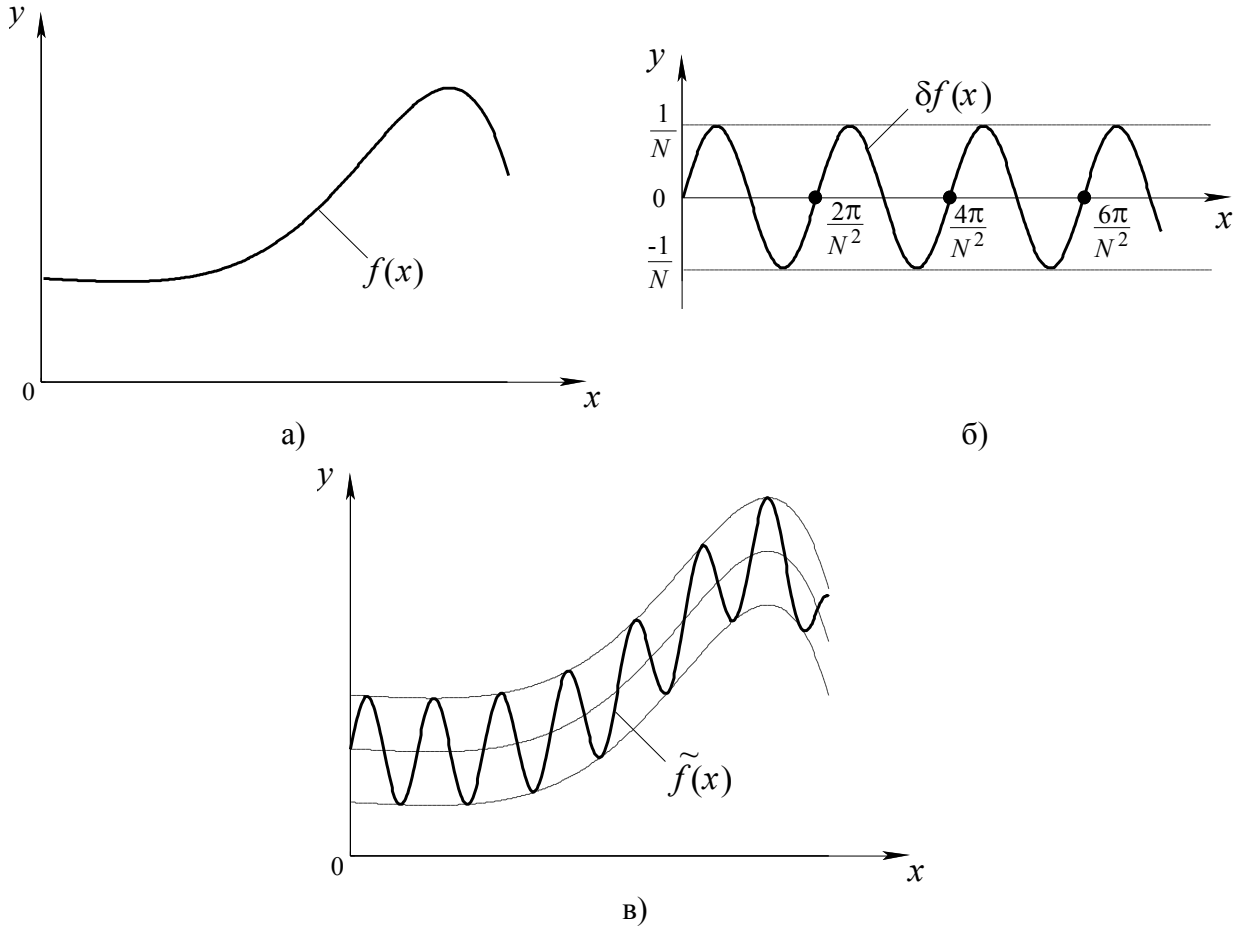

Рис. 1.3. **Иллюстрация неустойчивости задачи дифференцирования: а) график исходной функции *f*(*x*); б) график функции погрешности δ*f*(*x*); в) график результирующей функции** $\tilde{f}(x) = f(x) + \delta f(x)$

Вследствие влияния погрешности вместо точного значения производной $f'(x)$ получим приближенное значение $\tilde{f}'(x) = f'(x) + N\cos N^2 x$. Очевидно, максимальная величина разности $\tilde{f}'(x) - f'(x)$ на интервале $x \in [0; a]$ равна $N = \frac{1}{\varepsilon}$ и тем больше, чем больше N. Причина такого поведения $\tilde{f}'(x)$

ясна из рис. 1.3. При возрастании N период колебаний $\tilde{f}(x)$ уменьшается и график $\tilde{f}(x)$ возрастает и спадает более круто. Поэтому производная $\tilde{f}'(x)$ (наклон касательной к кривой $\tilde{f}(x)$) сильно отличается от $f'(x)$.

Таким образом, малому изменению (ε) функции $f(x)$ соответствует большое изменение (1/ε) ее производной, и, значит, задача дифференцирования неустойчива. Отсюда, как следствие, неустойчивость и задачи численного дифференцирования. Это означает, что на точность численного дифференцирования сильно влияет точность задания самой функции.

1.6.2. Корректные и некорректные задачи

Помимо устойчивости, вводят также понятие корректности задачи.

Задача называется ***корректной*** (поставленной корректно), если для любых допустимых исходных данных ее решение существует, единственно и устойчиво по исходным данным.

Условие корректности более сильное, чем условие устойчивости, так как здесь дополнительно требуется существование решения при любых разумных исходных данных, а также единственность решения. Все рассмотренные ранее неустойчивые задачи одновременно являются и некорректными.

Некорректные (неустойчивые) задачи встречаются на практике, и их приходится решать. В силу своих специфических свойств эти задачи требуют повышенного внимания к выбору численного метода, решение их не всегда может быть успешным, особенно в случае большой размерности. Универсальным способом повышения эффективности решения таких задач является увеличение разрядности при представлении чисел в ЭВМ (например, эмуляция машинной арифметики с большим количеством разрядов). Развиваются также методы решения некоторых специальных классов некорректных задач. Это так называемые *методы регуляризации*, которые основаны на замене исходной некорректной задачи корректно поставленной.

1.7. Требования к вычислительным методам (алгоритмам)

Перечислим основные *общие требования*, предъявляемые к вычислительным методам (алгоритмам):

- ✓ *точность;*
- ✓ *быстродействие;*
- ✓ *экономичность;*
- ✓ *надежность.*

Указанные требования являются *противоречивыми*, поэтому при разработке алгоритмов указывается, на какие характеристики необходимо делать акцент – например, на быстродействие или экономичность.

Помимо перечисленных общих требований, к алгоритмам могут предъявляться также *специальные требования*, например, возможность решать задачи большой размерности, возможность реализации на конкретном языке и конкретной ЭВМ и т.д.

Рассмотрим требования к вычислительным алгоритмам более подробно.

Точность. Требование точности – это главное требование, предъявляемое к вычислительному алгоритму. Оно означает, что алгоритм должен давать решение исходной задачи *с заданной точностью* $\varepsilon > 0$ за конечное число действий $N(\varepsilon)$. Алгоритм должен быть *реализуемым*, т.е. давать решение задачи за допустимое машинное время (допустимое число действий). Для большинства алгоритмов время решения задачи (объем вычислений $N(\varepsilon)$) возрастает при повышении точности, т.е. при уменьшении ε. Поэтому обычно выбирают разумный компромисс между точностью и временем решения задачи на данной ЭВМ.

Быстродействие. Быстродействие *характеризуется машинным временем (числом действий)*, необходимым для решения задачи. Естественно добиваться, чтобы число действий $N(\varepsilon)$ (и тем самым требуемое машинное время) было *минимальным* для поставленной задачи. Для любой задачи можно предложить много алгоритмов, обеспечивающих одинаковую по порядку точность $\varepsilon > 0$, но за разное число действий $N(\varepsilon)$. Среди этих, как говорят, эквивалентных по порядку точности алгоритмов следует выбрать тот, который дает решение с затратой наименьшего машинного времени (числа действий $N(\varepsilon)$).

Экономичность. Требование экономичности означает, что объем оперативной памяти (ОП) ЭВМ, необходимый для работы алгоритма, не должен превышать заданной величины. Объем ОП является лимитирующим фактором при решении задач большой размерности (например, решении СЛАУ высокого порядка), а также при создании программ для специализированных (управляющих) ЭВМ, обладающих небольшим объемом памяти. В таких случаях требование экономичности часто входит в противоречие с точностью и быстродействием алгоритма, т.е. для уменьшения объема ОП приходится жертвовать указанными характеристиками.

Иногда быстродействие алгоритма (число операций) также включают в характеристику экономичности.

Надежность. Свойство надежности можно определить как *работоспособность алгоритма при любых исходных данных*, в том числе реагирование на ошибочные ситуации.

Надежность является интегральной (обобщенной) характеристикой алгоритма и включает в себя такие его свойства, как *сходимость* и *устойчивость (корректность),* а также *обработку особых ситуаций*. Рассмотрим указанные свойства подробнее.

Сходимость алгоритма (численного метода). Свойство сходимости означает, что алгоритм (метод) позволяет найти решение, близкое к истинному решению, в необходимом диапазоне изменения исходных данных.

Более конкретно понятие сходимости должно формулироваться применительно к конкретному классу численных методов. Рассмотрим два характерных случая. Распространенной группой численных методов являются ***методы дискретизации***. В них задача с непрерывными параметрами заменяется на дискретную задачу, в которой значения функций вычисляются в фиксированных точках. К этой группе относятся, в частности, методы численного интегрирования, решения дифференциальных уравнений и др.

Для методов дискретизации под *сходимостью метода* понимается стремление решения дискретной задачи к решению непрерывной задачи при уменьшении параметра дискретизации (шага интегрирования, шага сетки и т.д.).

Другой разновидностью численных методов являются ***итерационные методы***. В них используется *итерационный процесс*, в котором на каждом шаге строится очередное приближение x_i (i = 1, 2, …) к решению:

$$x_0 \to x_1 \to x_2 \to \ldots \to x_n \approx x^*, \qquad (1.3)$$

где x_0 – начальное приближение, x^* - истинное решение. Итерационные методы применяются при нахождении корней полиномов, решении линейных и нелинейных систем уравнений, оптимизации функций и т.д.

Говорят, что итерационный процесс (1.3) *сходится* к точному решению x^*, если при неограниченном возрастании числа итераций предел последовательности $x_0, x_1, x_2, \ldots, x_i, \ldots$ существует и равен x^*, т.е. $\lim\limits_{i\to\infty} x_i = x^*$. Соответствующий итерационный метод называется ***сходящимся***.

На практике требуется сходимость численных методов в широком диапазоне изменения входных данных. Например, метод численного интегрирования должен быть сходящимся для любых функций $f(x)$; итерационный метод вычисления корней полинома должен сходиться независимо от порядка полинома, значений коэффициентов, выбора начального приближения для корня и т.д. Однако для сложных задач нет абсолютно сходящихся методов. Эффективные сходящиеся численные методы созданы, в частности, для численного интегрирования, вычисления корней полиномов

(вплоть до порядка 50÷100), решения СЛАУ (с числом неизвестных до $n = 10^6$), решения дифференциальных уравнений. В то же время решение систем нелинейных уравнений, многомерных задач оптимизации и некоторых других задач сопряжено с существенными трудностями.

Устойчивость и корректность алгоритма (численного метода). Ранее было рассмотрено свойство устойчивости *вычислительной задачи*: малая погрешность в исходных данных Δx приводит к малой погрешности результата Δy. Близкое понятие вводится также для *вычислительного алгоритма* (*метода*), однако в данном случае под погрешностью Δx понимается погрешность округления в ЭВМ. При каждом акте вычислений (каждой элементарной арифметической операции) появляются ошибки округления. В зависимости от алгоритма эти ошибки округления могут либо нарастать, либо затухать.

Если в процессе вычислений погрешности округления неограниченно нарастают, такой алгоритм называют *неустойчивым (вычислительно неустойчивым)*. Если погрешности округления не накапливаются, то алгоритм является *устойчивым*.

Приведем пример анализа устойчивости алгоритма.

Пример 1.4. Исследование устойчивости алгоритма.

Пусть в алгоритме реализуется итерационный процесс в соответствии с формулой

$$y_{i+1} = q\, y_i, \text{ где } i = 0, 1, 2, \ldots ,$$

где y_0 и q заданы. Предположим, что при вычислении y_i внесена погрешность округления δ_i, т.е. вместо точного значения y_i получено приближенное значение $\tilde{y}_i = y_i + \delta_i$. Тогда вместо y_{i+1} получим

$$\tilde{y}_{i+1} = q\,(y_i + \delta_i) = y_{i+1} + q\,\delta_i .$$

Отсюда видно, что погрешность $\delta_{i+1} = \tilde{y}_{i+1} - y_{i+1}$, возникающая при вычислении y_{i+1}, связана с погрешностью δ_i соотношением

$$\delta_{i+1} = q\,\delta_i,\ i = 0, 1, 2, \ldots$$

Следовательно, если $|\,q\,| > 1$, то в процессе вычислений абсолютное значение погрешности будет возрастать. Например, если при вычислении числа y_0 (только одного!) допущена ошибка округления δ_0, то число y_1 будет вычислено с ошибкой $q\delta_0$, число y_2 – с ошибкой $q^2\,\delta_0$ и т.д.

Таким образом, при $|\,q\,| > 1$ данный алгоритм неустойчив. Если же $|\,q\,| \le 1$, то погрешность не возрастает, т.е. алгоритм устойчив.

Аналогично понятию корректности вычислительной задачи вводится соответствующие понятие для алгоритма (численного метода).

Алгоритм (численный метод) называется *корректным*, если при любых значениях исходных данных численное решение существует, единственно и устойчиво относительно погрешности исходных данных.

Следуют различать устойчивость (корректность) вычислительной задачи и устойчивость (корректность) алгоритма ее решения (численного метода). Если задача неустойчива, ее сложно решать *любым* численным методом. Если задача устойчива (корректна), это еще не означает, что она может быть хорошо решена любым методом. Для достижения успеха необходимо выбрать устойчивый метод решения данной задачи.

Итак, численный метод должен *сохранять* свойство устойчивости (корректности) задачи. Вычислительно неустойчивый метод может «испортить» решение устойчивой задачи.

Проиллюстрируем сказанное двумя примерами.

Пример 1.5. Вычисление интеграла.

В п.1.6. было показано, что задача интегрирования устойчива. Построим рекуррентный численный метод вычисления интеграла

$$I_n = \int_0^1 x^n e^{x-1} dx, \quad n = 1, 2, ...$$

Применив интегрирование по частям, можно получить следующие соотношения:

$$I_1 = \int_0^1 x e^{x-1} dx = \frac{1}{e};$$

$$I_2 = \int_0^1 x^2 e^{x-1} dx = 1 - 2I_1;$$

$$I_3 = \int_0^1 x^3 e^{x-1} dx = 1 - 3I_2;$$

$$\ldots \quad \ldots \quad \ldots \quad \ldots \quad \ldots$$

$$I_n = \int_0^1 x^n e^{x-1} dx = 1 - nI_{n-1},$$

т.е. каждый очередной «составляющий» интеграл I_i можно найти по предыдущему I_{i-1}, используя рекуррентное соотношение

$$I_i = 1 - i\, I_{i-1},\ i = 2, 3, \ldots, n;\ I_1 = \frac{1}{e}.$$

Применив полученное рекуррентное соотношение, вычисляем, округляя до 6 значащих цифр:

$I_1 = 0{,}367879;$ $I_6 = 0{,}127120;$
$I_2 = 0{,}263242;$ $I_7 = 0{,}110160;$
$I_3 = 0{,}207274;$ $I_8 = 0{,}118720;$
$I_4 = 0{,}170904;$ $I_9 = -0{,}0684800.$
$I_5 = 0{,}145480;$

Однако действительное значение интеграла I_9 не может быть отрицательным, поскольку подынтегральная функция $x^9 e^{x-1}$ на всем отрезке интегрирования [0; 1] неотрицательная. Исследуем источник погрешности. Нетрудно убедиться, что округление до шести знаков в I_1 дает погрешность, примерно равную лишь $4{,}4 \cdot 10^{-7}$ по сравнению с точным значением $1/e$. Однако на каждом шаге (при вычислении I_2, I_3 и т.д.) указанная погрешность умножается на число по модулю большее единицы (на 2, 3, … , 9), что в итоге дает $2 \cdot 3 \cdot 4 \cdot \ldots \cdot 9 = 9!$. Это и приводит к результату, не имеющему смысла. Здесь причиной является алгоритм решения задачи, который оказался вычислительно неустойчивым, несмотря на корректность (устойчивость) самой задачи интегрирования.

Пример 1.6. Вычисление значений функции $\sin x$.

Для вычисления многих элементарных функций используются ряды. Рассмотрим вычисление значений функции $\sin x$ с помощью ее ряда Тейлора:

$$\sin x = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \ldots \,. \qquad (1.4)$$

Исследуем в начале устойчивость самой задачи вычисления функции $y(x) = \sin x$. Очевидно, функция $y(x)$ непрерывна, величина ее производной $y'(x) = \cos x$ по модулю не превышает единицу. Отсюда, а также непосредственно из графика функции $\sin x$ следует, что малым погрешностям Δx в задании аргумента будут соответствовать малые погрешности Δy в величине функции. Таким образом, задача вычисления $\sin x$ устойчива.

Изучим теперь погрешности алгоритма вычисления $\sin x$, основанного на применении ряда (1.4). С этой целью вычислим с помощью (1.4) значения $\sin x$ при нескольких величинах x. Для наглядности вычисления проведем с четырьмя верными десятичными знаками, т.е. будем имитировать счет в ЭВМ с четырехразрядной десятичной арифметикой. Члены ряда, меньшие 10^{-4}, не будем учитывать, так как при использовании четырех разрядов они не будут влиять на результат. По признаку Лейбница остаток сходящегося знакочередующегося ряда (т.е. погрешность суммы конечного числа членов) не превышает по модулю значения первого из отброшенных членов. Таким образом, можно ожидать, что абсолютная погрешность вычисления $\sin x$ с помощью ряда (1.4) не превысит 10^{-4}, а относительная – 0,02%.

Возьмем вначале $x = \frac{\pi}{6} = 0{,}5236$ (30°). Получим:

$\sin 0{,}5236 \approx 0{,}5236 - 0{,}2393 \cdot 10^{-1} + 0{,}3281 \cdot 10^{-3} = 0{,}5000.$

Таким образом, четыре значащих цифры результата совпадают с точным значением $\sin \frac{\pi}{6} = \frac{1}{2}$.

Из курса математики известно, что разложение (1.4) для функции $\sin x$ справедливо при любом значении аргумента ($-\infty < x < \infty$). Поэтому используем его для вычисления $\sin x$ при $x = 2\pi + \frac{\pi}{6} = 6{,}807$ (390°):

$$\sin 6{,}807 \approx 6{,}807 - 52{,}56 + 121{,}7 - 134{,}3 + 86{,}46 - 36{,}42 + 10{,}81 - 2{,}386 + \\ +0{,}4066 - 0{,}05508 + 0{,}006077 - 0{,}0005565 \approx 0{,}5493$$

Абсолютная погрешность полученного результата равна около 0,05, а относительная – 10% вместо ожидаемого значения 0,02% по признаку Лейбница.

Улучшить точность вычислений можно, использовав большее число разрядов для представления чисел. Выполним те же вычисления с восемью значащими цифрами, учитывая члены ряда до 10^{-8}:

$$\sin (2\pi + \frac{\pi}{6}) = \sin 6{,}8067841 \approx 0{,}49999993.$$

В результате ошибка уменьшилась. Однако повышенная точность вычислений не всегда помогает. Вычислим значение $\sin x$ при $x = 4\pi + \frac{\pi}{6} = 25{,}656340$ с использованием восьми разрядов. В итоге получим результат, не имеющий смысла:

$$\sin (4\pi + \frac{\pi}{6}) = \sin 25{,}656340 \approx 24{,}254018.$$

Итак, рассмотренный алгоритм вычисления $\sin x$, основанный на непосредственном использовании ряда (1.4), является численно неустойчивым и может привести к катастрофическим ошибкам, несмотря на устойчивость поставленной задачи. Это объясняется сильным влиянием погрешностей округления при $|x| > 1$ на величину суммы, а также неудачным способом суммирования ряда (слева направо без учета величины членов – см. п.2.5.2).

В данном случае существует простой способ резко уменьшить погрешности вычислений и сделать алгоритм устойчивым. Он состоит в применении формул приведения, благодаря чему аргумент x будет всегда находиться в интервале $[0; \frac{\pi}{4}]$, т.е. $|x| < 1$. Такая модификация алгоритма при ограничении первыми пятью членами разложения (1.4) и вычислениях с точностью до восьми разрядов приводит к максимальной абсолютной погрешности $0{,}4 \cdot 10^{-6}$ (порядка первого отброшенного члена в соответствии с признаком Лейбница) независимо от величины x. Подобный способ обычно

используется при вычислении тригонометрических и других элементарных функций с помощью рядов (см. п.3.4.).

Приведенные примеры позволяют сделать ***важные выводы***. Один из выводов состоит в том, что непосредственное применение классических математических методов для вычислений, без учета их реализации на ЭВМ, может приводить к неудовлетворительным последствиям. Результаты классической математики справедливы только для *точных* вычислений, однако в реальных вычислениях на ЭВМ неизбежны ошибки округления, а также другие источники ошибок. Поэтому приходится модифицировать существующие или разрабатывать новые методы, специально предназначенные для вычислений на ЭВМ.

Второй, более общий вывод состоит в том, что нельзя слепо верить результатам, полученным на ЭВМ. Нужно быть готовым к присутствию ошибок в алгоритме и программе, возникающих не только по невнимательности, но и чисто методических, иногда самых неожиданных. Специалисты отмечают, что психологически человек склонен доверять расчетам на ЭВМ. Аккуратные колонки чисел с большим количеством значащих цифр, красивые графики усиливают это доверие. Однако даже в многократно проверенных математических пакетах, не говоря уже о вновь создаваемых программах, бывают ошибки.

Таким образом, всегда нужно оставлять место сомнениям. Если человек к существованию ошибок не готов, он может испытать буквально потрясение из-за невозможности объяснить полученные результаты. Рассмотренные примеры хорошо иллюстрируют эту ситуацию, так как применение, казалось бы, безупречно строгих математических подходов при отсутствии ошибок в программировании алгоритма приводит к бессмысленным результатам типа $\sin 25{,}66 \approx 24{,}25$.

Сказанное свидетельствует о большой важности анализа точности и устойчивости алгоритмов, выявления возможных источников погрешностей, а также о необходимости тщательного и всестороннего тестирования программ с использованием хорошо изученных задач. В приведенных примерах ошибочность результатов установить было очень легко, так как они не соответствуют классическим представлениям математики. Однако в реальных ситуациях определить, что результаты вычислений содержат ошибки, бывает весьма непросто.

Обработка ошибок и особых ситуаций. Остановимся на этом вопросе, хотя он скорее относится к культуре программирования, а не непосредственно к разработке вычислительных алгоритмов. В ходе вычислений на ЭВМ могут возникать ситуации, ведущие к переполнению разрядной сетки или невозможности выполнения операции. Примерами могут быть деление на нуль, логарифм нуля, тангенс от аргумента $\frac{\pi}{2}$, извлечение квадратного корня из отрицательного числа и т.д. Кроме подобных «стандарт-

ных» ошибок, могут быть ошибки или особые ситуации, связанные со спецификой алгоритма. Например, невозможно найти точку пересечения двух параллельных прямых, и если встретилась такая ситуация, нужно предусмотреть соответствующие действия программы. Еще один вид особых ситуаций – слишком долгие вычисления, так называемое «зависание»: неизвестно, нормально ли работает алгоритм или он «зациклился», когда закончится вычислительный процесс и закончится ли он вообще.

Многие трансляторы обрабатывают «стандартные» ошибки и извещают о них. Однако этого часто недостаточно, так как обычно процесс вычислений прерывается и управление передается операционной системе. Между тем желательно предоставить возможность самому пользователю, не выходя из программы, принять решение – прекратить вычисления или продолжить их, например, изменив исходные данные. Для получения надежного алгоритма (программы) следует по возможности учесть специфические ситуации, проверить допустимость значений входных данных, обратить внимание на информативность выдаваемых сообщений об ошибках, предусмотреть для пользователя возможность прерывать процесс вычислений в любое время и т.д. Общее правило состоит в том, что *алгоритм и программа* (а не транслятор и не операционная система) должны обрабатывать особые ситуации и ошибки при вычислениях.

Организация входа исходных данных в вычислительных программах. Хотя этот вопрос относится к построению интерфейсной части программы, мы рассмотрим его здесь в связи с тем, что в программах, предназначенных для вычислений, ввод исходных данных имеет определенную специфику. Начинающие программисты часто организуют ввод числовых данных с клавиатуры. Однако при этом не учитывается, что, как правило, вычисления с помощью программы выполняют многократно, обычно модифицируя только часть (иногда очень небольшую) исходных данных. Если заставить пользователя вводить каждый раз заново даже не очень большое количество чисел (например, больше 3÷5), уже после нескольких запусков программы он найдет этот процесс слишком утомительным.

Существует несколько способов преодоления такой ситуации. Первый – это ввод чисел из файла в обычном текстовом формате, который предварительно записывается в текстовом редакторе. При этом рекомендуется, помимо числовых данных, в файл включать содержательные обозначения переменных и комментарии, используя для разбора простейшие синтаксические анализаторы. Второй способ – это использование входных данных «по умолчанию», в качестве которых берутся типовые или наиболее вероятные значения переменных. Еще один способ – использование в программе собственного редактора входных данных с контролем значений переменных и возможностью записи чисел в файл и последующего считывания. На практике указанные способы могут комбинироваться. В любом случае ввод чисел с клавиатуры стараются минимизировать.

2. ПОГРЕШНОСТИ ОКРУГЛЕНИЯ ПРИ ВЫЧИСЛЕНИЯХ НА ЭВМ И ИХ НАКОПЛЕНИЕ

2.1. Основные понятия теории приближенных вычислений

2.1.1. Абсолютные и относительные ошибки

Понятие погрешности связано с измерением или вычислением определенных величин (например, физических), которые количественно характеризуются числами. Так как на практике никакие измерения и их обработка не могут быть выполнены абсолютно точно, то *точное значение* величины остается неизвестным и его приходится заменять *приближенным значением* этой величины

Пусть a^* – точное значение некоторого числа, a – приближенное его значение.

***Абсолютной погрешностью* (*ошибкой*)** числа называется разность между его истинным значением и приближенным значением, полученным в результате вычисления или измерения:

$$\Delta(a) = a^* - a. \qquad (2.1)$$

Для оценки точности величин понятия абсолютной ошибки недостаточно. Например, если абсолютная ошибка 1 мм получилась при измерении расстояний в 100 метров и 1 метр, то ясно, что в первом случае измерения выполнены с более высокой точностью. Более показательна в этом примере относительная ошибка.

***Относительной погрешностью* (*ошибкой*)** числа называется отношение абсолютной погрешности к точному значению числа:

$$\delta(a) = \frac{\Delta(a)}{a^*} = \frac{a^* - a}{a^*} = 1 - \frac{a}{a^*}. \qquad (2.2)$$

Относительную погрешность часто измеряют в процентах: $\delta(a)\ [\%] = \delta(a) \cdot 100\%$. В приводимом выше примере относительные погрешности измерений составляют соответственно 0,001% и 0,1%.

Недостатком применения относительной погрешности является то, что она не определена при $a^* = 0$ и очень велика, если значение a^* близко к нулю, хотя абсолютная ошибка может быть мала. На практике для оценки точности величин используют как абсолютную, так и относительную погрешности. Иногда под абсолютной и относительной ошибкой понимают модули величин $\Delta(a)$ и $\delta(a)$ соответственно.

Итак, если известны точное a^* и приближенное a значения некоторого числа, то по формулам (2.1), (2.2) можно вычислить абсолютную и относительную ошибки. Однако, как уже отмечалось, истинное значение определяемой величины обычно неизвестно, поэтому ошибку приближенного чис-

ла a определить нельзя. Тем не менее, почти всегда можно указать число, оценивающее эту ошибку. В результате приходим к понятиям предельной абсолютной и предельной относительной погрешностей.

Число $\overline{\Delta}(a)$, удовлетворяющее неравенству

$$| a^* - a | = | \Delta(a) | \le \overline{\Delta}(a), \qquad (2.3)$$

называется ***предельной абсолютной погрешностью*** приближенного числа a.

Очевидно, такое определение предельной абсолютной погрешности не является однозначным. Так, если $a^* = \pi$, а за приближенное значение взять $a = 3{,}14$, то, учитывая, что $3{,}140 < \pi < 3{,}142$, можно записать:

$$| \pi - a | < 0{,}002; \; | \pi - a | \le 0{,}01; \; | \pi - a | \le 0{,}1.$$

Каждое из чисел 0,002, 0,01, 0,1 будет предельной абсолютной погрешностью числа a. Но чем ближе между собой числа $| a^* - a |$ и $\overline{\Delta}(a)$, тем точнее предельная погрешность оценивает фактическую ошибку. Поэтому в качестве предельной абсолютной погрешности $\overline{\Delta}(a)$ приближенного числа a берут по возможности наименьшее из чисел, удовлетворяющих неравенству (2.3).

Если известна предельная погрешность $\overline{\Delta}(a)$, то можно утверждать, что точное значение числа a^* гарантированно содержится в интервале $[a - \overline{\Delta}(a), \; a + \overline{\Delta}(a)]$, т.е.

$$a - \overline{\Delta}(a) \le a^* \le a + \overline{\Delta}(a). \qquad (2.4)$$

Принята также следующая условная запись, эквивалентная (2.4):

$$a^* = a \pm \overline{\Delta}(a), \qquad (2.5)$$

т.е. a^* приближенно равно a с погрешностью $\overline{\Delta}(a)$. Например, запись $x = 3{,}14 \pm 0{,}02$ означает, что истинное значение числа x находится между 3,12 и 3,16.

При анализе ошибок вычислений на ЭВМ для оценки относительной погрешности приближенного числа a формулу (2.2) обычно видоизменяют и полагают

$$\delta(a) = \frac{\Delta(a)}{a} = \frac{a^* - a}{a}, \qquad (2.6)$$

такое определение удобнее, так как истинное значение числа a^* неизвестно. Очевидно, при малых $\Delta(a)$ справедливо $a \approx a^*$ и, таким образом, величины $\delta(a)$, вычисляемые по формулам (2.2) и (2.6), отличаются незначительно. В дальнейшем мы будем использовать определение относительной погрешности с помощью формулы (2.6).

Из (2.1) и (2.6) следует, что если известны значения погрешностей $\Delta(a)$ и $\delta(a)$, то точное значение числа a^* может быть записано как:

$$a^* = a + \Delta(a) = a\,(1 + \frac{\Delta(a)}{a}) = a\,(1 + \delta(a)). \qquad (2.7)$$

Предельной относительной погрешностью приближенного числа a называется величина

$$\bar{\delta}(a) = \frac{\bar{\Delta}(a)}{|a|}. \qquad (2.8)$$

Из (2.3), (2.6) и (2.8) следует, что величина $\bar{\delta}(a)$ удовлетворяет неравенству:

$$\left|\frac{a^* - a}{a}\right| = |\,\delta(a)\,| \le \bar{\delta}(a).$$

Иначе говоря, $\bar{\delta}(a)$ – это верхняя оценка модуля относительной погрешности числа.

Точное значение числа a^* гарантированно находится в интервале

$$a(1 - \bar{\delta}(a)) \le a^* \le a(1 + \bar{\delta}(a)),$$

что условно записывается как

$$a^* = a\,(1 \pm \bar{\delta}(a)). \qquad (2.9)$$

Например, если $x = 3{,}14 \cdot (1 \pm 0{,}01)$, то точное значение числа x находится в промежутке между числами $3{,}14 \cdot 0{,}99$ и $3{,}14 \cdot 1{,}01$.

Если предельная абсолютная погрешность $\bar{\Delta}(a)$ (предельная относительная погрешность $\bar{\delta}(a)$) каким-либо образом оценена, то в дальнейшем она принимается в качестве абсолютной (относительной) погрешности приближенного числа a при анализе точности измерений и вычислений.

2.1.2. Значащие, верные и сомнительные цифры числа

Рассмотрим понятия значащих и верных цифр числа. Пусть положительное число a представлено в виде конечной десятичной дроби:

$a = \alpha_m \cdot 10^m + \alpha_{m-1} \cdot 10^{m-1} + \ldots + \alpha_{m-n+1} \cdot 10^{m-n+1}$,

здесь α_i – цифры числа a (α_i =0, 1, …, 9), индекс i обозначает номер десятичного разряда числа, разряд с номером m является старшим. Например,

$307{,}69 = 3 \cdot 10^2 + 0 \cdot 10^1 + 7 \cdot 10^0 + 6 \cdot 10^{-1} + 9 \cdot 10^{-2}$.

Каждая единица, стоящая на определенном месте в числе a, написанном в виде десятичной дроби, имеет свое значение. Единица, стоящая на первом месте, равна 10^m, на втором – 10^{m-1}, на n-ом – 10^{m-n+1} и т.д.

Значащими цифрами числа называются все цифры в записи числа, начиная с первой ненулевой слева.

Пример: a=0,006070. Здесь первые три нуля не являются значащими цифрами, так как они служат для установки десятичных разрядов других цифр. Остальные цифры числа являются значащими. Последний (значащий) нуль является представителем сохраненного десятичного разряда.

Точность приближенного числа зависит не от того, сколько в этом числе цифр, а от того, сколько значащих цифр заслуживает доверия.

Значащую цифру приближенного числа называют ***верной*** в узком (широком) смысле, если абсолютная погрешность числа не превосходит половины единицы (единицы) разряда, соответствующего этой цифре. В противном случае цифра считается ***сомнительной***.

Пример. Определить количество верных значащих цифр приближенного числа a=12,48±0,07 в узком и широком смысле.

Решение. Представим число a в виде $12{,}48 = 1\cdot10^{1} + 2\cdot10^{0} + 4\cdot10^{-1} + 8\cdot10^{-2}$ и составим таблицу

Цифра	1	2	4	8
Единица разряда	10	1	0,1	0,01
Половина единицы разряда	5	0,5	0,05	0,005

Сравнивая для каждой цифры половину единицы (единицу) разряда с абсолютной погрешностью числа $\Delta(a)$=0,07, получим, что в узком смысле верными являются цифры 1 и 2, а сомнительными – цифры 4 и 8; в широком смысле верны цифры 1, 2, 4 и сомнительна цифра 8.

В соответствии с определением, если для приближенного числа $a = \alpha_m\cdot10^{m} + \alpha_{m-1}\cdot10^{m-1} + \ldots + \alpha_{m-n+1}\cdot10^{m-n+1}$, заменяющего точное число a^*, известно, что

$$|\Delta(a)| = |a^* - a| \le 0{,}5\cdot10^{m-n+1} \text{ в узком смысле,}$$

$$|\Delta(a)| = |a^* - a| \le 10^{m-n+1} \text{ в широком смысле,}$$

то первые n цифр $\alpha_m, \alpha_{m-1}, \ldots, \alpha_{m-n+1}$ этого числа являются верными.

Пример. Найти количество верных значащих цифр приближенного числа a=43,00, заменяющего точное число a^*=42,98.

Решение. Очевидно, что здесь m=1, так как $a = 4\cdot10^{1} + 3\cdot10^{0} + 0\cdot10^{-1} + 0\cdot10^{-2}$. Справедливы неравенства:

$$|\Delta(a)| = |a^* - a| = 0{,}02 \le 0{,}5\cdot10^{-1};$$

$$|\Delta(a)| = |a^* - a| = 0{,}02 \le 10^{-1}.$$

Имеем: $m-n+1=-1$, и при m=1 n=3, т.е. первые три цифры числа 4, 3, 0 являются верными в узком и широком смысле, последний нуль при этом является сомнительным.

Понятие верных цифр числа в узком смысле используется при записи приближенных чисел, а также при анализе погрешностей ручных и машинных вычислений с применением симметричного округления (п.2.3). Понятие верных цифр в широком смысле удобно для анализа погрешностей при округлении методом усечения (п.2.3).

2.1.3. Связь абсолютной и относительной погрешностей приближенного числа с количеством верных знаков этого числа

Если известно, что все значащие цифры числа верны в узком (широком) смысле, можно легко найти его предельную абсолютную погрешность, которая в соответствии с определением равна половине единицы (единице) последнего разряда.

Пример. a=0,087;

$\overline{\Delta}(a)$=0,0005=5·10^{-4} в узком смысле;

$\overline{\Delta}(a)$=0,001=10^{-3} в широком смысле.

a=0,0870;

$\overline{\Delta}(a)$=0,00005=5·10^{-5} в узком смысле;

$\overline{\Delta}(a)$=0,0001=10^{-4} в широком смысле.

Предельная относительная погрешность приближенного числа может быть найдена по числу его верных знаков с помощью следующей теоремы, которую мы приводим без доказательства.

Теорема. Если приближенное положительное число a имеет n верных десятичных знаков в узком смысле, то предельная относительная погрешность этого числа равна

$$\overline{\delta}(a)=\frac{1}{\alpha_m}\left(\frac{1}{10}\right)^{n-1},$$

где α_m – первая значащая цифра числа.

Если приближенное число a имеет n верных знаков в широком смысле, то приведенную оценку следует увеличить в 2 раза.

Пример. Какова предельная относительная погрешность при замене числа π числом a=3,14?

Решение. Очевидно, что здесь n=3 и α_m=3, поэтому

$$\overline{\delta}(a)=\frac{1}{3}\left(\frac{1}{10}\right)^{3-1}=\frac{1}{300}\approx 0{,}33\ \%.$$

Иногда приходится решать обратную задачу – определить количество n верных знаков числа a по заданной предельной относительной погрешности числа $\overline{\delta}(a)$. Для этого следует вначале найти предельную абсолютную

погрешность числа по формуле $\overline{\Delta}(a) = a \cdot \overline{\delta}(a)$ и затем воспользоваться определением, приведенным в п.2.1.2.

Пример. Приближенное число a=24253 имеет относительную погрешность 1%. Сколько в нем верных знаков?

Решение. Найдем вначале предельную абсолютную погрешность числа a:

$$\overline{\Delta}(a) = a \cdot \overline{\delta}(a) = 24253 \cdot 0{,}01 \approx 243 = 2{,}43 \cdot 10^2.$$

Теперь, пользуясь определением п.2.1.2, найдем, что верными (как в узком, так и в широком смысле) являются лишь первые две цифры числа a=24253, остальные цифры являются сомнительными. Поэтому число a предпочтительнее записать в виде a=2,43·10^4 (см. п. 2.1.4).

2.1.4. Запись приближенных чисел

Очевидно, сохранять при записи приближенного числа большое количество недостоверных значащих цифр не имеет смысла. Более того, это даже неправильно, так как создает иллюзию высокой точности числа, которой на самом деле нет.

Точную информацию о погрешностях приближенных чисел дают записи в виде $a^* = a \pm \Delta(a)$ или $a^* = a(1 \pm \delta(a))$, однако из-за громоздкости частое использование такой формы неудобно. Поэтому в вычислительной практике приняты такие правила представления приближенных чисел, которые позволяют непосредственно по виду числа судить о его точности.

Приближенные числа принято записывать таким образом, чтобы все значащие цифры числа были верными в узком смысле. При этом сам вид числа сразу показывает его предельную абсолютную погрешность, которая равна половине единицы последнего разряда, сохраняемого при записи.

Например, запись 3,1416 означает, что абсолютная погрешность этого приближенного числа не превосходит 0,0005, а запись 370 – что абсолютная погрешность не превосходит 0,5.

Таким образом, приближенные числа 3,7·10^2; 37·10; 370; 370,0; 370,00 имеют разную степень точности, их предельные абсолютные погрешности составляют соответственно 50; 5; 0,5; 0,05 и 0,005. Число 275 тысяч с абсолютной погрешностью меньше 500 следует записать в виде 275·10^3. Запись вида 270000 применять нельзя, так как она означает абсолютную погрешность меньше 0,5.

В математических таблицах приводятся только верные цифры соответствующих чисел. При расчетах с приближенными значениями констант или функций, взятыми из математических таблиц, следует считать, что абсолютная погрешность их равна половине единицы младшего разряда табличных значений этих констант или функций.

2.2. Представление чисел в ЭВМ

Для изучения особенностей и погрешностей машинных вычислений следует рассмотреть представление чисел в ЭВМ. В настоящее время во всех вычислительных устройствах принята запись чисел, основанная на позиционной системе счисления. Например, в десятичной системе счисления, которой мы обычно пользуемся, запись 103,67 определяет число

$$1 \cdot 10^2 + 0 \cdot 10^1 + 3 \cdot 10^0 + 6 \cdot 10^{-1} + 7 \cdot 10^{-2}.$$

Здесь 10 – основание системы счисления, запятая отделяет дробную часть числа от целой, 1, 0, 3, 6, 7 – числа из базисного набора {0, 1, 2, 3, … 9}, с помощью которого можно представить любое вещественное число.

ЭВМ работает, как правило, в двоичной системе, когда любое число записывается в виде последовательности нулей и единиц. Например, запись $11{,}0101_2$ в двоичной системе определяет число

$$1 \cdot 2^1 + 1 \cdot 2^0 + 0 \cdot 2^{-1} + 1 \cdot 2^{-2} + 0 \cdot 2^{-3} + 1 \cdot 2^{-4}.$$

В общем случае *в позиционной системе с основанием p* запись

$$a = \pm\, \alpha_n \alpha_{n-1} \ldots \alpha_0\, \alpha_{-1}\, \alpha_{-2} \ldots \qquad (2.10)$$

означает, что

$$a = \pm\, (\alpha_n\, p^n + \alpha_{n-1}\, p^{n-1} + \ldots + \alpha_0\, p^0 + \alpha_{-1}\, p^{-1} + \alpha_{-2}\, p^{-2} + \ldots),$$

здесь основание p – целое число, большее единицы. Каждое из чисел α_i может принимать одно из значений $\{0, 1, \ldots, p-1\}$.

Числа α_i называются ***разрядами***, например: α_3 – третий разряд перед запятой, α_2 – второй разряд после запятой.

Запись вида (2.10) называется ***представлением вещественного числа*** в форме *с фиксированной запятой*. Например, десятичные числа с фиксированной запятой – это привычные нам записи чисел 5; –10; 175,12; 0,0093 и т.д.

Распространена также форма представления чисел *с плавающей запятой*:

$$a = \mathrm{M} p^{\Pi},$$

здесь p – основание системы счисления, число M представляется в форме с фиксированной запятой и называется ***мантиссой числа*** a, число П называется ***порядком числа*** a.

Десятичное число в этой форме записи имеет вид $a = \mathrm{M} \cdot 10^{\Pi}$. Например, число –273,9 можно записать в виде: $-2739 \cdot 10^{-1}$; $-2{,}739 \cdot 10^2$; $-0{,}2739 \cdot 10^3$ и т.д. Таким образом, представление числа с плавающей запятой неоднозначно. Для однозначного представления вводят нормализацию, т.е. накладывают на мантиссу числа ограничение

$$\frac{1}{p} \le |\,\mathrm{M}\,| < 1. \qquad (2.11)$$

Полученную таким образом запись называют ***нормализованным представлением числа с плавающей запятой.*** Если представить мантиссу числа в виде $M = \pm 0, \alpha_{-1}, \alpha_{-2}, \ldots, \alpha_{-n}$, то при $\alpha_{-1} \neq 0$ имеем нормализованную форму числа.

Для десятичных чисел условие нормализации (2.11) имеет вид $0{,}1 \le |M| < 1$. Поэтому число – 273,9 в нормализованном представлении запишется как $-0{,}2739 \cdot 10^3$, мантисса $M = -0{,}2739$ и порядок $П = 3$.

Для двоичной системы условие нормализации (2.11) принимает форму $\frac{1}{2} \le |M| < 1$. Это означает, что в представлении мантиссы $M = \pm 0, \alpha_{-1}, \alpha_{-2}, \ldots, \alpha_{-n}$ первая цифра после нуля (разряд α_{-1}) всегда будет равна единице. Например, двоичное число $101{,}11_2$ в нормализованном виде равно $0{,}10111_2 \cdot 2^3$, мантисса $M = 0{,}10111$, порядок $П = 3_{10} = 11_2$.

В современных компьютерах при вычислениях используются два типа чисел – целые и с плавающей запятой. Целые числа – это числа с фиксированной запятой (запятая стоит после числа). В первых ЭВМ была реализована только целочисленная арифметика (т.е. арифметика с фиксированной запятой). В настоящее время в компьютерах, как правило, *аппаратно* реализуется арифметика с плавающей запятой.

Определим диапазон представления целых чисел в ЭВМ. В 32-разрядных процессорах для хранения целого числа отводится 4 байта, т.е. 32 двоичных разряда (бита), причем один разряд используется для хранения знака числа (0 – «плюс», 1 – «минус»). В общем случае диапазон изменения двоичных целых чисел определяется соотношением

$$-p^r \le a \le p^r - 1,$$

где r – число двоичных разрядов, отводимых для представления собственно значения числа (без знака). Увеличение диапазона для изображения отрицательных чисел на единицу по сравнению с положительными цифрами связано со спецификой их представления в ЭВМ (отрицательные числа представляются в дополнительном коде). Таким образом, для 32-разрядной арифметики ($r = 31$) имеем:

$$-2^{31} \le a \le 2^{31} - 1.$$

Число 2^{31} равно 2147483648, т.е. примерно 2 млрд.

В 16-разрядных процессорах под целое число отводятся 2 байта, т.е. 16 двоичных разрядов ($r = 15$). В этом случае диапазон изменения целых чисел –

$$-2^{15} \le a \le 2^{15} - 1, \text{ т.е. } -32768 \le a \le 32767.$$

Рассмотрим теперь представление в ЭВМ чисел с плавающей запятой. Приведем форму хранения двоичных чисел, рекомендуемую стандартом

Рис. 2.1. **Условное изображение разрядной сетки для 32-разрядной арифметики**

IEEE 1985 г., она взята за основу при создании популярных процессоров фирмы Intel и других фирм. В 32-разрядной арифметике числа с плавающей запятой представляются в памяти 32 двоичными разрядами (4 байта), из них 24 бита отводятся для хранения мантиссы М и 8 бит – для хранения порядка П (см. рис.2.1). В нормализованной форме самый левый бит мантиссы любого двоичного числа всегда равен единице, это позволяет его не хранить (это так называемый «фантомный» бит). В тоже время из 24 бит, отведенных под мантиссу, один бит используется для хранения знака мантиссы (знака числа).

Для удобства вместо порядка числа П в 8 отведенных битах хранится так называемая характеристика числа X, которая принимает целые значения от 0 до $255 = 2^8 - 1$. Она связана с порядком числа соотношением $X = П + 127$. Чтобы получить фактическое значение показателя, нужно вычесть 127 из хранимого числа: $П = X - 127$. Значение характеристики $X = 255$ резервируется для представления бесконечности, а также указания результатов некорректных операций (например, квадратный корень из отрицательного числа). Таким образом, реально для указания величины (и знака) порядка используются значения характеристики $X = 0 \div 254$, следовательно, диапазон изменения порядка числа равен $П = -127 \div 127$.

Наибольшее число с плавающей запятой, которое может быть представлено в данной ЭВМ, называется ***машинной бесконечностью*** и обозначается M_∞. Из сказанного выше следует, что для 32-разрядной арифметики в стандарте IEEE

$$M_\infty = +1{,}111...1_2 \cdot 2^{127} \approx 10^{38}.$$

Для указания машинной бесконечности в англоязычной литературе используется аббревиатура OFL (overflow level – уровень переполнения).

Наименьшее положительное число с плавающей запятой, которое может быть представлено в ЭВМ, называется ***машинным нулем*** и обозначается M_0. Для 32-разрядных процессоров:

$$M_0 = +1{,}000...0_2 \cdot 2^{-127} \approx 10^{-38}.$$

Для обозначения машинного нуля применяется аббревиатура UFL (underflow level).

Итак, все вещественные числа с плавающей запятой, которые могут быть представлены в ЭВМ, расположены по абсолютной величине в диапазоне от M_0 до M_∞:

$$M_0 \le |a| \le M_\infty.$$

Очевидно, что значения машинной бесконечности M_∞ и машинного нуля M_0 в конкретной ЭВМ определяются числом разрядов, отводимых под порядок числа с плавающей запятой. Если все числа, представимые в ЭВМ, нанести на числовую ось (рис. 2.2), то левее числа M_0 будет находиться ближайшее число 0. В записи с плавающей запятой – это число с нулевой мантиссой и любым порядком. Отметим, что нуль и целые числа представляются в ЭВМ особым образом.

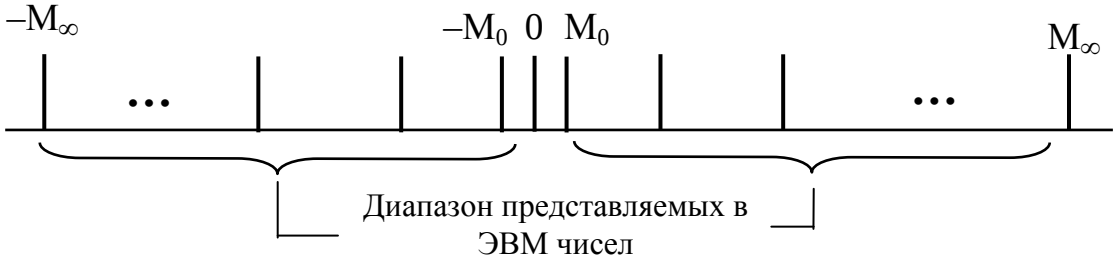

Рис. 2.2. **Изображение чисел, представляемых в ЭВМ, на числовой оси**

При выполнении вычислений в ЭВМ могут получаться числа по модулю меньше, чем M_0, или больше, чем M_∞. Поэтому в машинной арифметике вводятся определенные соглашения. В частности, если в процессе счета какой-либо задачи появится вещественное число, меньшее по модулю чем M_0, то ему присваивается нулевое значение:

$$\text{если } |a| < M_0, \text{ то } a = 0.$$

Так, в 32-разрядной арифметике при перемножении двух чисел 10^{-20} и 10^{-21} получим не привычный результат 10^{-41}, а нуль. Как видно, правила машинной и обычной арифметики отличаются.

При появлении в процессе счета вещественного числа, большего по модулю, чем M_∞, происходит так называемое переполнение разрядной сетки, после чего ЭВМ прекращает счет задачи. Эта ситуация обозначается также «авост» (автоматический останов) или OFL:

$$\text{если } |a| > M_\infty, \text{ то OFL.}$$

В результате, например, имеем: $10^{20} \cdot 10^{21}$ = OFL; 1,346/0 = OFL и т.д.

Еще одной важной особенностью машинных вычислений является *конечность* и *дискретность* множества чисел, с которыми оперирует ЭВМ. Как известно, множество вещественных чисел бесконечно и «плотно». Однако из-за ограниченности разрядной сетки в ЭВМ можно представить точно не все числа из диапазона $[M_0, M_\infty]$, а лишь конечное подмножество этого множества – именно те числа, которые являются комбинациями двоичных цифр (0 и 1) в пределах используемой разрядной сетки. Эта ситуация изображена условно на рис.2.2. Количество вещественных чисел в диапазоне $[M_0, M_\infty]$, с которыми оперирует ЭВМ, равно количеству всевозможных

комбинаций двоичных цифр в рамках, установленных разрядной сеткой, т.е. зависит от числа разрядов, отведенных под мантиссу и порядок числа. Для 32-разрядной арифметики в стандарте IEEE это количество составляет примерно 2^{31}, т.е. около 2 млрд.

Очевидно, интервал изменения порядка $П_{min} \div П_{max}$ определяет диапазон вещественных чисел по величине, а число разрядов мантиссы – дискретность («густоту») распределения их на отрезке числовой оси. Разность между двумя соседними числами равна единице последнего двоичного разряда.

Интересно, что числа с плавающей запятой в интервале $[M_0, M_\infty]$ распределены неравномерно. Между каждыми двумя соседними степенями двойки находится 2^{22} числа с плавающей запятой, например, 2^{22} чисела между 2^{-127} и 2^{-126} и столько же чисел между 2^{126} и 2^{127}. Таким образом, числа с плавающей запятой гуще расположены вблизи нуля.

Итак, диапазон изменения чисел с плавающей запятой гораздо шире и они расположены гораздо «гуще» на числовой оси, чем числа с фиксированной запятой. Этим и объясняется преимущественное применение таких чисел в современных ЭВМ.

В заключение этого раздела сделаем замечание, касающееся способа иллюстрации машинной арифметики. Как уже упоминалось, в ЭВМ внутреннее представление чисел – двоичное. Однако для человека такое представление чисел не вполне привычно. Поэтому, чтобы дальнейшее изложение материала стало более ясным и не было связано с деталями аппаратной реализации, мы будем демонстрировать особенности машинной арифметики, опираясь в основном на десятичную, а не на двоичную арифметику.

2.3. Округление чисел в ЭВМ

Число a^*, не представимое в ЭВМ точно, подвергается округлению, т.е. заменяется близким ему числом a, представимым в ЭВМ точно. Точность представления в ЭВМ чисел с плавающей запятой характеризуется абсолютной и относительной погрешностями (см. п. 2.1), которые в данном случае являются погрешностями (ошибками) округления. При этом под точным числом a^* понимается исходное (округляемое) число, а под приближенным a – округленное число.

Рассмотрим округление чисел в ЭВМ.

Округлением числа до n разрядов (n значащих цифр) в принятой системе счисления называется операция, в результате которой осуществляется замена исходного числа таким числом, все разряды которого, начиная с $(n+1)$-го, являются нулевыми. (Здесь используется нумерация разрядов слева направо, т.е. самый левый разряд является старшим и имеет первый номер.)

Величина погрешностей зависит от способа округления. В вычислительной практике нашли наибольшее применение два основных способа:

1) симметричное округление;

2) отбрасывание младших разрядов (усечение числа).

Симметричное округление рассматривается в школьном курсе математики и является общепринятым при «ручных» вычислениях.

Приведем *правила симметричного округления десятичных чисел*. Чтобы округлить число до n значащих цифр, отбрасывают все цифры, стоящие справа от n-ой 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 остаются 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, введенное Гауссом).

Таким образом, если при округлении числа отбрасывается меньше половины единицы последнего сохраняемого разряда, то цифры всех сохраненных разрядов остаются неизменными. Если же отброшенная часть числа составляет больше половины единицы последнего сохраненного десятичного разряда, то цифра этого разряда увеличивается на единицу. В исключительном случае, когда отброшенная часть в точности равна половине единицы последнего сохраненного десятичного разряда, для взаимной компенсации знаков ошибок округления используется правило четной цифры.

Например, при округлении до 4-х знаков (разрядов) десятичных чисел, содержащих 6 значащих цифр, получим следующие значения округленного числа a и абсолютной погрешности округления $\Delta(a)$:

$$a_1^* = 457{,}328;\ a_1 = 457{,}3;\ \Delta(a_1) = 0{,}028;$$

$$a_2^* = 1{,}23584;\ a_2 = 1{,}236;\ \Delta(a_2) = -\,0{,}00016.$$

Очевидно, что при симметричном округлении модуль абсолютной погрешности $|\Delta(a)|$ может достигать половины единицы последнего сохраняемого разряда числа. Последняя величина принимается за предельную абсолютную погрешность приближенного числа для указанного способа округления. Таким образом, для приведенных выше чисел имеем:

$$\overline{\Delta}(a_1) = 0{,}05;\ \overline{\Delta}(a_2) = 0{,}0005.$$

Например, значение $a_1 = 457{,}3$ могло быть получено округлением чисел 457,328; 457,347; 457,269 и т.д. Во всех случаях $|\Delta(a)| \le \overline{\Delta}(a) = 0{,}05$, т.е. все указанные числа гарантированно находятся в диапазоне $457{,}3 \pm 0{,}05$.

При округлении с точностью до n разрядов по способу усечения сохраняются все разряды числа слева до n-го включительно, остальные (младшие) разряды отбрасываются, т.е. число просто «усекается». Пользуясь этим способом, получим:

$$a_1^* = 457{,}328;\ a_1 = 457{,}3;\ \Delta(a_1) = 0{,}028;$$

$$a_2^* = 1{,}23584;\ a_2 = 1{,}235;\ \Delta(a_2) = 0{,}00094.$$

Как видно, абсолютная погрешность при отбрасывании младших разрядов числа a_2^* больше, чем при симметричном округлении этого числа.

Округленное значение $a_1 = 457{,}3$ является представлением всех чисел в диапазоне от 457,300 до 457,399. Таким образом, предельная абсолютная погрешность при усечении числа равна единице последнего сохраняемого разряда, т.е. в 2 раза превышает предельную погрешность симметричного округления:

$$\overline{\Delta}(a_1) = 0{,}1;\ \overline{\Delta}(a_2) = 0{,}001.$$

Из сравнения двух способов округления следует, что метод симметричного округления приводит к меньшим ошибкам вычисления. Однако техническая реализация способа усечения гораздо проще, не требует дополнительных программных ресурсов, а значит, и затрат машинного времени. Поэтому в современных ЭВМ обычно применяется именно этот способ. В некоторых алгоритмических языках предусмотрена возможность выбора способа округления самим программистом.

Как уже отмечалось, расстояние (разность) между двумя соседними числами с плавающей запятой на числовой оси равно единице последнего двоичного разряда, эта величина и принимается за предельную абсолютную погрешность при округлении (усечении) двоичного числа в ЭВМ.

Обычно больший интерес при вычислениях представляет относительная погрешность. Найдем предельную относительную погрешность при округлении чисел с плавающей запятой по способу усечения.

Пусть для записи мантиссы в ЭВМ (без учета знака) отводится t двоичных разрядов. Предположим, что исходное (точное) число представлено в виде бесконечной двоичной дроби

$$a^* = \pm 2^{\text{П}}\left(\frac{\beta_1}{2} + \frac{\beta_2 +}{2^2} + \ldots + \frac{\beta_t}{2^t} + \frac{\beta_{t+1}}{2^{t+1}} + \ldots\right),$$

где каждая из цифр мантиссы β_i равна 0 или 1. Отбрасывая все лишние разряды, получим округленное число

$$a = \pm 2^{\text{П}}\left(\frac{\beta_1}{2} + \frac{\beta_2 +}{2^2} + \ldots + \frac{\beta_t}{2^t}\right).$$

Очевидно, абсолютная погрешность округления равна:

$$\Delta(a) = a - a^* = \pm 2^{\Pi}\left(\frac{\beta_{t+1}}{2^{t+1}} + \frac{\beta_{t+2}}{2^{t+2}} + ...\right).$$

Так как значения β_{t+1}, β_{t+2} и т.д. не превышают 1, для модуля погрешности $\Delta(a)$ можно записать оценку

$$|\Delta(a)| \le 2^{\Pi}\frac{1}{2^{t+1}}\left(1 + \frac{1}{2} + \frac{1}{2^2} + ...\right) = 2^{\Pi - t},$$

здесь учитывается, что предел суммы $\left(1 + \frac{1}{2} + \frac{1}{2^2} + ...\right)$ равен 2.

Как отмечалось в п.2.2, из условия нормализации двоичного числа $\frac{1}{2} \le |\mathrm{M}| \le 1$ следует, что всегда β_1=1. Поэтому модуль числа a^* отвечает неравенству

$$|a^*| \ge 2^{\Pi} \cdot \frac{1}{2} = 2^{\Pi - 1}.$$

Теперь найдем оценку для $|\delta(a)|$:

$$|\delta(a)| = \frac{|\Delta(a)|}{|a^*|} \le \frac{2^{\Pi - t}}{2^{\Pi - 1}} = 2^{-t+1}.$$

Итак, при округлении чисел с плавающей запятой способом усечения модуль относительной погрешности удовлетворяет неравенству:

$$|\delta(a)| = \frac{|a^* - a|}{|a^*|} \le 2^{-t+1},$$

где t – число двоичных разрядов, используемых для хранения мантиссы (без учета знака). Это означает, что *предельная относительная погрешность*

$$\bar{\delta}(a) = 2^{-t+1} \qquad (2.12)$$

при округлении определяется только числом разрядов t, отводимых под мантиссу, и не зависит от самого числа. В связи с важностью числа 2^{-t+1} в машинной арифметике оно получило специальное наименование.

Определение. Число $\varepsilon_{маш} = 2^{-t+1}$ характеризует предельную относительную точность представления чисел в ЭВМ и называется ***машинным эпсилоном***.

Из (2.9) и (2.12) следует, что точное значение числа a^* находится в пределах

$$a^* = a\,(1 \pm \varepsilon_{маш}), \qquad (2.13)$$

где a – округленное число. Отсюда вытекает еще одно определение машинного эпсилона: величина $\varepsilon_{маш}$ – это такое пороговое малое число, которое ЭВМ еще «различает» по сравнению с единицей. В машинной арифметике справедливы правила:

$$1+a = 1, \text{ если } a < \varepsilon_{маш};$$
$$1+a > 1, \text{ если } a \geq \varepsilon_{маш}.$$

Иначе говоря, при $a \geq \varepsilon_{маш}$ число a достаточно велико, чтобы в сумме $1+a$ изменить последний двоичный разряд; в противном случае результат равен 1, т.е. остается без изменений. Указанные соотношения также показывают отличия машинной арифметики от обычной.

В 32-разрядной арифметике по стандарту IEEE для записи мантиссы используется t=23 разряда (см. п. 2.2), поэтому:

$$\varepsilon_{маш} = 2^{-22} \approx 1{,}2 \cdot 10^{-7} = 1{,}2 \cdot 10^{-5}\,\%.$$

Это означает, что при выполнении любых действий, использующих числа с плавающей запятой длиной 4 байта (с одинарной точностью),в результате могут быть верными не более чем семь десятичных знаков. Можно сделать следующий ***важный вывод***: *бессмысленно рассчитывать на большую относительную точность, чем* $\varepsilon_{маш}$*, в любом результате вычислений с плавающей запятой*. Многие вычислительные программы имеют входной параметр ε, задающий желаемую точность результата. Неразумно присваивать ему значение, меньшее $\varepsilon_{маш.}$

Точность в семь десятичных разрядов не всегда достаточна для вычислений. Для повышения точности стандарт IEEE рекомендует выполнять арифметику с большей разрядностью, несмотря на то что результаты операций записываются в память с 32 битами. В компьютерах, поддерживающих этот стандарт, арифметика с плавающей точкой часто реализована с внутренней разрядностью 80 битов.

Кроме того, во многих алгоритмических языках предусмотрены числа с плавающей запятой двойной точности. Для их представления используется вдвое большее число двоичных разрядов, чем для обычных чисел с плавающей запятой. В случае 32-разрядной арифметики числа с двойной точностью занимают 8 байт, при этом диапазон представления чисел составляет около $10^{-306} \div 10^{306}$, а относительная точность представления чисел – около 10^{-15}, т.е. верны первые 15 десятичных цифр.

Во многих ЭВМ для ускорения счета арифметика с удвоенной точностью реализована аппаратно. В частности, в компьютерах, совместимых с IBM PC, вычисления с удвоенной точностью реализованы в сопроцессоре (отдельном или встроенном). В некоторых алгоритмических языках допускается также арифметика с комплексными числами с плавающей запятой одинарной и удвоенной точности, при этом комплексное число представляется парой вещественных чисел. Комплексная арифметика редко поддерживается аппаратно; как правило, она реализуется программным путем, в этом случае вычисления с комплексными числами выполняются гораздо медленнее.

2.4. Предельные погрешности при арифметических действиях с приближенными числами

Одним из важнейших вопросов при оценке точности численных методов является вопрос о том, как ошибки, возникающие в определенном месте в ходе вычислений, распространяются дальше, т.е. становится ли их влияние больше или меньше по мере того, как производятся последующие операции. Как уже отмечалось раньше, при катастрофическом возрастании ошибок алгоритм является неустойчивым и некорректным.

В качестве первого шага при анализе распространения и накопления ошибок следует оценить погрешности, возникающие при выполнении арифметических действий. Определенные возможности для этого дает классическая теория погрешностей, которая устанавливает правила оценки ошибок при операциях с приближенными данными. Теория погрешностей (вычислений с приближенными числами) сформировалась задолго до появления ЭВМ и успешно применяется при обработке результатов вычислений и измерений. Следует отметить, однако, что классическая теория не учитывает особенностей машинной арифметики и ее результаты применительно к вычислениям на ЭВМ должны уточняться, об этом подробно будет сказано дальше.

Приведем основные правила классической теории для оценки предельных погрешностей при арифметических действиях с приближенными числами.

2.4.1. Сложение и вычитание

1. Предельная абсолютная погрешность *алгебраической суммы* (т.е. суммы или разности) нескольких чисел равна сумме предельных абсолютных погрешностей этих чисел:

$$\overline{\Delta}(a_1 + a_2 + \ldots + a_n) = \overline{\Delta}(a_1) + \overline{\Delta}(a_2) + \ldots + \overline{\Delta}(a_n). \qquad (2.14)$$

Доказательство. Пусть $a_1, a_2, \ldots, a_n$ – исходные (точные) числа, их алгебраическая сумма равна

$$S^* = \pm a_1^* \pm a_2^* \pm \ldots \pm a_n^*.$$

Аналогично алгебраическая сумма приближенных чисел

$$S = \pm a_1 \pm a_2 \pm \ldots \pm a_n.$$

Очевидно, что абсолютная погрешность суммы равна:

$$\Delta(S) = S^* - S = \pm\Delta(a_1) \pm \Delta(a_2) \pm \ldots \pm\Delta(a_n)$$

и, следовательно,

$$|\Delta(S)| \le |\Delta(a_1)| + |\Delta(a_2)| + \ldots + |\Delta(a_n)|.$$

Взяв в качестве оценок абсолютных погрешностей $|\Delta(a_i)|$ приближенных чисел их предельные погрешности $\overline{\Delta}(a_i)$, найдем:

$$|\Delta(S)| \le \sum_{i=1}^{n} \overline{\Delta}(a_i)$$

или

$$\overline{\Delta}(S) = \sum_{i=1}^{n} \overline{\Delta}(a_i).$$

Из (2.14) следует, что предельная абсолютная погрешность суммы не может быть меньше предельной абсолютной погрешности наименее точного из слагаемых (т.е. слагаемого, имеющего наибольшую погрешность). Таким образом, с какой бы точностью ни были определены остальные слагаемые, мы не можем за их счет увеличить точность суммы.

2. Из (2.14) вытекает, что предельная относительная погрешность *алгебраической суммы* чисел равна:

$$\overline{\delta}(a_1 + a_2 + ... + a_n) = \frac{\overline{\Delta}(a_1 + a_2 + ... + a_n)}{|(a_1 + a_2 + ... + a_n)|} = \frac{\overline{\Delta}(a_1) + \overline{\Delta}(a_2) + ... + \overline{\Delta}(a_n)}{|(a_1 + a_2 + ... + a_n)|}. \quad (2.15)$$

Учитывая, что в соответствии с (2.8) $\overline{\Delta}(a_i) = |a_i| \overline{\delta}(a_i)$, из (2.15) получим также:

$$\overline{\delta}(a_1 + a_2 + ... + a_n) = \frac{1}{|S|} \sum_{i=1}^{n} |a_i| \overline{\delta}(a_i), \quad (2.16)$$

где $S = a_1 + a_2 + \ldots + a_n$.

Обратим внимание на то, что в формулах (2.14)-(2.16) оцениваются погрешности алгебраической (а не арифметической) суммы, т.е. числа a_i могут быть разного знака.

На основании общих формул (2.14)-(2.16) легко могут быть записаны погрешности суммы и разности двух чисел одинакового знака:

$$\overline{\Delta}(a \pm b) = \overline{\Delta}(a) + \overline{\Delta}(b); \quad (2.17)$$

$$\overline{\delta}(a \pm b) = \frac{\overline{\Delta}(a) + \overline{\Delta}(b)}{|a \pm b|} = \frac{|a|}{|a \pm b|} \overline{\delta}(a) + \frac{|b|}{|a \pm b|} \overline{\delta}(b). \quad (2.18)$$

3. Если слагаемые a_i одного знака (т.е. сумма арифметическая), то предельная относительная погрешность суммы не превышает предельной относительной погрешности исходных чисел:

$$\min_{i=\overline{1,n}} \overline{\delta}(a_i) \le \overline{\delta}(a_1 + a_2 + ... + a_n) \le \max_{i=\overline{1,n}} \overline{\delta}(a_i). \quad (2.19)$$

Доказательство. Пусть $S = a_1 + a_2 + \ldots + a_n$ и для определенности положим $a_i > 0$. За предельную относительную погрешность суммы можно принять

$$\overline{\delta}(S) = \frac{\overline{\Delta}(S)}{S} = \frac{\overline{\Delta}(a_1) + \overline{\Delta}(a_2) + ... + \overline{\Delta}(a_n)}{a_1 + a_2 + ... + a_n}.$$

Так как $\overline{\delta}(a_i) = \overline{\Delta}(a_i)/a_i$, то

$$\overline{\Delta}(a_i) = a_i \cdot \overline{\delta}(a_i).$$

Пусть $\overline{\delta}_{\max} = \max\limits_i \overline{\delta}(a_i)$ и $\overline{\delta}_{\min} = \min\limits_i \overline{\delta}(a_i)$ – соответственно максимальная и минимальная из погрешностей $\overline{\delta}(a_i)$. Тогда имеем:

$$\overline{\delta}(S) = \frac{\sum\limits_{i=1}^{n} a_i \cdot \overline{\delta}(a_i)}{S} \le \frac{\overline{\delta}_{\max} \sum\limits_{i=1}^{n} a_i}{S} = \overline{\delta}_{\max};$$

$$\overline{\delta}(S) = \frac{\sum\limits_{i=1}^{n} a_i \cdot \overline{\delta}(a_i)}{S} \ge \frac{\overline{\delta}_{\min} \sum\limits_{i=1}^{n} a_i}{S} = \overline{\delta}_{\min}.$$

Приведем примеры применения данных формул.

Пример 2.1. Вычисление предельных погрешностей суммы чисел.

Пусть стороны треугольника a=10, b=15, c=20 измерены с одинаковыми предельными абсолютными погрешностями $\overline{\Delta}(a) = \overline{\Delta}(b) = \overline{\Delta}(c) = 0{,}1$, т.е. $a^* = 10 \pm 0{,}1$; $b^* = 15 \pm 0{,}1$; $c^* = 20 \pm 0{,}1$. Необходимо найти предельные абсолютную и относительную погрешности вычисления периметра треугольника P.

Вычисляем периметр: $P = a + b + c = 45$. По формуле (2.14) имеем:

$$\overline{\Delta}(a+b+c) = \overline{\Delta}(a) + \overline{\Delta}(b) + \overline{\Delta}(c) = 0{,}3.$$

Таким образом, $P = 45 \pm 0{,}3$.

Найдем теперь по формуле (2.15) предельную относительную погрешность вычисления периметра:

$$\overline{\delta}(a+b+c) = \frac{\overline{\Delta}(a) + \overline{\Delta}(b) + \overline{\Delta}(c)}{|a+b+c|} = \frac{0{,}3}{45} \approx 0{,}0067 = 0{,}67\%,$$

т.е. $P = 45(1 \pm 0{,}067)$.

Сравним полученную относительную погрешность периметра с относительными погрешностями измерения сторон треугольника:

$$\overline{\delta}(a) = \frac{\overline{\Delta}(a)}{|a|} = \frac{0{,}1}{10} = 0{,}01 = 1\%;\ \overline{\delta}(b) = \frac{\overline{\Delta}(b)}{|b|} = \frac{0{,}1}{15} \approx 0{,}067 = 0{,}67\%;$$

$$\overline{\delta}(c) = \frac{\overline{\Delta}(c)}{|c|} = \frac{0{,}1}{20} = 0{,}005 = 0{,}5\%.$$

Как видно, относительная погрешность суммы $P = a + b + c$ не превышает относительной погрешности каждого из слагаемых, что согласуется с формулой (2.19).

Пример 2.2. Относительная погрешность разности двух чисел.

Перепишем формулу (2.18) для относительной погрешности разности двух чисел:

$$\bar{\delta}(a-b)=\frac{\bar{\Delta}(a)+\bar{\Delta}(b)}{|a-b|}. \qquad (2.20)$$

Из нее следует, что при вычитании двух близких чисел относительная погрешность может сильно возрастать. Продемонстрируем это на примере.

Пусть заданы два десятичных числа с 7 значащими цифрами: $a^* = 1{,}001999$; $b^* = 1{,}000001$. Точное значение разности этих чисел $R^* = a^* - b^* = 0{,}001998$.

Предположим, что ЭВМ имеет 4 десятичных разряда и вследствие округления по способу усечения числа a^* и b^* записаны в память ЭВМ с погрешностями: $a = 1{,}001$; $b = 1{,}000$. В результате будет найдено приближенное значение разности $R = a - b = 0{,}001$.

Вычислим относительные погрешности исходных чисел и результата:

$$\delta(a)=\frac{a^*-a}{a}=\frac{0{,}000999}{1{,}001}\approx 0{,}001=0{,}1\%;$$

$$\delta(b)=\frac{b^*-b}{b}=\frac{0{,}000001}{1{,}000}\approx 10^{-6}=10^{-4}\%;$$

$$\delta(R)=\frac{R^*-R}{R}=\frac{0{,}000998}{0{,}001}\approx 1{,}0=100\,\%.$$

Итак, несмотря на то, что исходные числа округлены с точностью не хуже 0,1%, разность этих чисел вычислена с погрешностью 100%.

Заметим, что в данном случае мы нашли действительную, а не предельную ошибку вычислений. Оценим теперь по формуле (2.20) предельную относительную погрешность. Предельные абсолютные погрешности при округлении чисел a и b равны единице последнего сохраняемого разряда, т.е. $\bar{\Delta}(a)=\bar{\Delta}(b)=0{,}001$. Тогда по формуле (2.20) имеем:

$$\bar{\delta}(a-b)=\frac{0{,}001+0{,}001}{0{,}001}=2=200\%.$$

Таким образом, оценка предельной относительной погрешности дала еще худший результат. Отсюда следует вывод, что при вычитании двух близких чисел может наблюдаться катастрофическое ухудшение точности вычислений.

2.4.2. Умножение, деление, возведение в степень и извлечение корня

1. При умножении или делении чисел их предельные относительные погрешности складываются:

$$\bar{\delta}(a_1\cdot a_2\cdot\ldots\cdot a_n)=\bar{\delta}(a_1)+\bar{\delta}(a_2)+\ldots+\bar{\delta}(a_n); \qquad (2.21)$$

$$\bar{\delta}\left(\frac{a_1 \cdot a_2 \cdot \ldots \cdot a_n}{b_1 \cdot b_2 \cdot \ldots \cdot b_m}\right) = \sum_{i=1}^{n} \bar{\delta}(a_i) + \sum_{i=1}^{m} \bar{\delta}(b_i). \qquad (2.22)$$

Доказательство. Докажем формулу (2.21). Пусть $P = a_1\, a_2 \ldots a_n$, для простоты положим $a_i>0$. Тогда $\ln P = \sum_{i=1}^{n} \ln a_i$.

Применим приближенную формулу $\Delta(\ln x) \approx d(\ln x) = \frac{\Delta(x)}{x}$, где d – знак дифференцирования. Так как $d(\ln P) = \sum_{i=1}^{n} d(\ln a_i)$, имеем:

$$\frac{\Delta(P)}{P} = \sum_{i=1}^{n} \frac{\Delta(a_i)}{a_i}.$$

Взяв абсолютные значения, получим:

$$\left|\frac{\Delta(P)}{P}\right| \le \sum_{i=1}^{n} \left|\frac{\Delta(a_i)}{a_i}\right|.$$

Используем в качестве оценок абсолютных погрешностей $|\,\Delta(a_i)\,|$ предельные значения $\bar{\Delta}(a_i)$ и учтем, что $\frac{\bar{\Delta}(a_i)}{a_i} = \bar{\delta}(a_i)$, $\left|\frac{\Delta(P)}{P}\right| = |\,\delta(P)\,|$. В результате найдем:

$$|\,\delta(P)\,| \le \sum_{i=1}^{n} \bar{\delta}(a_i)$$

или

$$\bar{\delta}(P) = \sum_{i=1}^{n} \bar{\delta}(a_i).$$

Формула (2.21), очевидно, верна и в том случае, если сомножители a_i имеют разные знаки. Доказательство формулы (2.22) аналогично.

На основании (2.21) и (2.22) легко записать соотношения для погрешностей при умножении и делении двух чисел:

$$\bar{\delta}(a \cdot b) = \bar{\delta}(a) + \bar{\delta}(b); \qquad (2.23)$$

$$\bar{\delta}\left(\frac{a}{b}\right) = \bar{\delta}(a) + \bar{\delta}(b). \qquad (2.24)$$

2. Предельная относительная погрешность k-й степени числа в k раз больше предельной относительной погрешности числа.

Доказательство. Предельная относительная погрешность при возведении числа в степень может быть получена из (2.21), если положить все сомножители a_i одинаковыми:

$$\bar{\delta}(a^k) = k \cdot \bar{\delta}(a).$$

3. Предельная относительная погрешность корня k-й степени в k раз меньше предельной относительной погрешности подкоренного числа.

Доказательство.

Пусть $y = \sqrt[k]{a}$, тогда $y^k = a$. Отсюда $\delta(\overline{a}) = \overline{\delta}(y^k) = k \cdot \overline{\delta}(y)$ и, значит,

$$\overline{\delta}(y) = \overline{\delta}(\sqrt[k]{a}) = \frac{1}{k}\overline{\delta}(a).$$

Приведем пример.

Пример 2.3. Вычисление предельной относительной погрешности произведения чисел.

Пусть измерены стороны параллелепипеда a= 10, b = 15, c = 20 с предельными абсолютными погрешностями $\overline{\Delta}(a) = \overline{\Delta}(b) = \overline{\Delta}(c)$=0,1. Нужно найти предельную относительную погрешность вычисления объема параллелепипеда.

Вначале вычислим объем параллелепипеда: $V = abc$ = 4500.

Предельные относительные погрешности измерения сторон были найдены в примере 2.1:

$$\overline{\delta}(a) = 0{,}01 = 1\ \%;\ \overline{\delta}(b) = 0{,}0067 = 0{,}67\ \%;\ \overline{\delta}(c) = 0{,}005 = 0{,}5\ \%.$$

Теперь по формуле (2.21) найдем предельную относительную погрешность вычисления объема:

$$\overline{\delta}(abc) = \overline{\delta}(a) + \overline{\delta}(b) + \overline{\delta}(c) = 0{,}01 + 0{,}0067 + 0{,}005 \approx 0{,}022 = 2{,}2\%.$$

Итак, объем параллелепипеда равен $V = 4500(1 \pm 0{,}022)$.

Сформулированные правила позволяют оценить предельные погрешности при вычислении по сложным формулам.

Пример 2.4. Вычисление предельной относительной погрешности функции

$$y = \sqrt{\frac{a+b}{x^3(1-x)}}.$$

Используя приведенные выше формулы, получим:

$$\overline{\delta}(y) = \frac{1}{2}[\overline{\delta}(a+b) + 3\overline{\delta}(x) + \overline{\delta}(1-x)] = \frac{1}{2}\left[\frac{\overline{\Delta}(a) + \overline{\Delta}(b)}{|a+b|} + 3\frac{\overline{\Delta}(x)}{|x|} + \frac{\overline{\Delta}(1) + \overline{\Delta}(x)}{|1-x|}\right].$$

Полученная оценка относительной погрешности содержит в знаменателе выражение $|1\text{-}x|$. Ясно, что при $x \approx 1$ можно получить очень большую погрешность.

На основании приведенных формул, определяющих погрешности при операциях с приближенными числами, можно сделать ***следующие выводы***:

1. При суммировании чисел одного знака предельная относительная погрешность не увеличивается.

2. При умножении и делении чисел предельная относительная погрешность увеличивается (предельные относительные погрешности исходных чисел складываются).

3. При вычитании относительная погрешность может резко возрастать.

4. Величины погрешностей при сложении (вычитании) и умножении (делении) не зависят от порядка суммирования или перемножения.

Представленные выводы являются полезными при анализе источников погрешностей и построении точных алгоритмов. Подчеркнем, однако, что *эти выводы справедливы для классической теории погрешностей*; в ней предполагается, что ошибки в исходных данных обусловлены, например, ошибками измерений и т.п. Классическая теория погрешностей *не учитывает специфики выполнения арифметических операций в ЭВМ*. В разделе 2.5 сформулированные выводы будут уточнены применительно к вычислениям на ЭВМ.

2.4.3 Общая формула для погрешности

Основная задача теории погрешностей заключается в следующем. Пусть задана некоторая функция

$$u = f(a_1, a_2, \ldots, a_n)$$

и известны погрешности аргументов функции a_i. Необходимо определить погрешность функции u.

Предположим, что функция u дифференцируема и $\Delta(a_i)$ – абсолютные погрешности аргументов a_i. Если считать погрешности $\Delta(a_i)$ малыми величинами, можно приближенно записать

$$|\Delta(u)| \approx |df(a_1, a_2, \ldots, a_n)| = \left|\sum_{i=1}^{n}\frac{\partial f}{\partial a_i}\Delta(a_i)\right| \le \sum_{i=1}^{n}\left|\frac{\partial f}{\partial a_i}\right|\cdot\left|\Delta(a_i)\right|.$$

Взяв в качестве оценок $|\Delta(a_i)|$ предельные абсолютные погрешности $\overline{\Delta}(a_i)$, имеем:

$$|\Delta(u)| \le \sum_{i=1}^{n}\left|\frac{\partial f}{\partial a_i}\right|\cdot\overline{\Delta}(a_i)$$

или

$$\overline{\Delta}(u) = \sum_{i=1}^{n}\left|\frac{\partial f}{\partial a_i}\right|\cdot\overline{\Delta}(a_i). \qquad (2.25)$$

Предельная относительная погрешность функции u равна:

$$\overline{\delta}(u) = \frac{\overline{\Delta}(u)}{|u|} = \frac{1}{|u|}\sum_{i=1}^{n}\left|\frac{\partial f}{\partial a_i}\right|\cdot\overline{\Delta}(a_i). \qquad (2.26)$$

Формулы (2.25) и (2.26) позволяют найти предельные погрешности функции, исходя из предельных погрешностей аргументов.

Пример. Найти предельные абсолютную и относительную погрешности вычисления объема шара $V=\frac{1}{6}\pi d^3$, если диаметр d=3,7±0,05 см, а $\pi\approx 3{,}14$.

Решение. В соответствии с формулой (2.25) предельная абсолютная погрешность объема равна:

$$\overline{\Delta}(V)=\left|\frac{\partial V}{\partial \pi}\right|\overline{\Delta}(\pi)+\left|\frac{\partial V}{\partial d}\right|\overline{\Delta}(d)=\frac{1}{6}d^3\overline{\Delta}(\pi)+\frac{1}{2}\pi\, d^2\overline{\Delta}(d).$$

Так как $\overline{\Delta}(\pi)$=0,005 и $\overline{\Delta}(d)$=0,05, то отсюда найдем:

$$\overline{\Delta}(V)=\frac{1}{6}\cdot 3{,}7^3\cdot 0{,}005+\frac{1}{2}\cdot 3{,}14\cdot 3{,}7^2\cdot 0{,}05=0{,}042+1{,}075=1{,}117\approx 1{,}1 \text{ см}^3.$$

Таким образом,

$$V=\frac{1}{6}\pi d^3 = 27{,}4 \pm 1{,}1 \text{ см}^3.$$

Предельная относительная погрешность объема шара равна

$$\overline{\delta}(V)=\frac{\overline{\Delta}(V)}{V}=\frac{1{,}1}{27{,}4}\approx 0{,}004=4\,\%.$$

Пример. Формулы (2.25), (2.26) можно использовать для вывода соотношений пп. 2.4.1, 2.4.2, определяющих погрешности при выполнении элементарных операций. Получим, например, выражение для предельной абсолютной погрешности разности двух чисел. Пусть $f(a, b)=a-b$, тогда по формуле (2.25) имеем:

$$\overline{\Delta}(a-b)=\left|\frac{\partial f}{\partial a}\right|\overline{\Delta}(a)+\left|\frac{\partial f}{\partial b}\right|\overline{\Delta}(b)=\overline{\Delta}(a)+\overline{\Delta}(b),$$

что совпадает с формулой (2.17).

2.4.4. Обратная задача теории погрешностей

На практике важна также ***обратная задача***: каковы должны быть абсолютные погрешности аргументов функции, чтобы абсолютная погрешность самой функции не превышала заданной величины?

Эта задача математически не определена, так как заданную предельную погрешность $\overline{\Delta}(u)$ функции $u = f(a_1, a_2, \ldots, a_n)$ можно обеспечить при различных соотношениях предельных погрешностей аргументов $\overline{\Delta}(a_i)$.

Простейшее решение обратной задачи дается ***принципом равных влияний***. Согласно этому принципу предполагается, что все слагаемые $\left|\frac{\partial f}{\partial a_i}\right|\cdot\overline{\Delta}(a_i)$ одинаково влияют на образование общей абсолютной погрешности $\overline{\Delta}(u)$ функции u.

Пусть величина предельной абсолютной погрешности $\overline{\Delta}(u)$ задана. В соответствии с (2.25) она равна

$$\overline{\Delta}(u)=\sum_{i=1}^{n}\left|\frac{\partial f}{\partial a_i}\right|\cdot\overline{\Delta}(a_i).$$

Предположим, что все слагаемые равны между собой, тогда получим

$$\left|\frac{\partial f}{\partial a_1}\right|\overline{\Delta}(a_1)=\left|\frac{\partial f}{\partial a_2}\right|\overline{\Delta}(a_2)=...=\left|\frac{\partial f}{\partial a_n}\right|\overline{\Delta}(a_n)=\frac{\overline{\Delta}(u)}{n}.$$

Отсюда найдем предельную абсолютную погрешность каждого из аргументов:

$$\overline{\Delta}(a_i)=\frac{1}{n}\cdot\frac{\overline{\Delta}(u)}{\left|\frac{\partial f}{\partial a_i}\right|};\ i=\overline{1,n}.$$

Пример. Радиус основания цилиндра $r\approx 2$ м, высота цилиндра $h\approx 3$ м. С какими абсолютными погрешностями нужно определить r, h и взять число π, чтобы объем цилиндра можно было вычислить с точностью до 0,1 м^3?

Решение. Объем цилиндра определяется формулой $V=\pi r^2 h$. Полагая $r = 2$, $h = 3$, $\pi\approx 3{,}14$, приближенно получим:

$$\frac{\partial V}{\partial \pi}=r^2h=12;\ \frac{\partial V}{\partial r}=2\pi rh=37{,}7;\ \frac{\partial V}{\partial h}=\pi r^2=12{,}6.$$

Так как число аргументов функции равно n=3 и $\overline{\Delta}(u)$=0,1, имеем:

$$\overline{\Delta}(\pi)=\frac{1}{3}\cdot\frac{\overline{\Delta}(V)}{\frac{\partial V}{\partial \pi}}=\frac{0{,}1}{3\cdot 12}\approx 0{,}003;$$

$$\overline{\Delta}(r)=\frac{1}{3}\cdot\frac{\overline{\Delta}(V)}{\frac{\partial V}{\partial r}}=\frac{0{,}1}{3\cdot 37{,}7}\approx 0{,}001;$$

$$\overline{\Delta}(h)=\frac{1}{3}\cdot\frac{\overline{\Delta}(V)}{\frac{\partial V}{\partial h}}=\frac{0{,}1}{3\cdot 12{,}6}\approx 0{,}003.$$

2.4.5. Общие правила вычислений с приближенными числами

При вычислениях с приближенными числами, очевидно, нецелесообразно использовать излишнее число значащих (недостоверных) цифр, так как это приводит к увеличению объема вычислений. Приведенные в пп. 2.4.1-2.4.4 формулы классической теории погрешностей позволяют оценивать ошибки вычислений при выполнении различных арифметических действий и определять требуемое число знаков приближенных чисел. Од-

нако при массовых вычислениях, часто выполняемых на практике, когда число действий велико, использование таких формул нецелесообразно по двум причинам. *Во-первых*, в этом случае детальный анализ погрешностей, появляющихся при выполнении каждой арифметической операции, весьма трудоемок. *Во-вторых*, при достаточно большом числе действий формулы классической теории погрешностей дают *слишком завышенную оценку ошибок*.

Это объясняется тем, что при выводе формул делаются два допущения: а) все погрешности $\Delta(a_i)$ исходных чисел a_i равны предельным значениям $\overline{\Delta}(a_i)$; б) все предельные погрешности $\overline{\Delta}(a_i)$ (отклонения приближенных чисел от точных значений) имеют одинаковый знак и, таким образом, «усиливаются». Между тем, на практике обычно $\Delta(a_i) < \overline{\Delta}(a_i)$, причем отклонения $\Delta(a_i)$ имеют разный знак и, следовательно, компенсируют друг друга (подробнее см. п. 2.5.7).

Поэтому при осуществлении массовых вычислений (например, в ручных расчетах или при вычислениях на калькуляторе) не учитывают погрешность каждого отдельного результата, а пользуются определенными практическими правилами. Выполнение этих правил обеспечивает разумную точность вычислений при экономии труда.

Рассмотрим вначале правила выполнения отдельных арифметических операций, вытекающие из результатов теории погрешностей.

1. При ***сложении и вычитании*** абсолютная погрешность результата не может быть меньше абсолютной погрешности наименее точного слагаемого. Поэтому не имеет смысла сохранять излишние знаки в более точных слагаемых.

Сложение чисел различной точности выполняется *по следующему правилу*:

а) выделяют число (или числа), имеющее наибольшую абсолютную погрешность (с наименьшим количеством верных значащих цифр);

б) остальные числа округляют по образцу выделенного, сохраняя один запасной знак (т.е. оставляют на один знак больше, чем в выделенном числе);

в) производят сложение и результат округляют на один знак, т.е. оставляют столько значащих цифр, сколько их имеет число с наименьшим количеством верных цифр.

2. При ***вычитании близких чисел*** предельная относительная погрешность может быть весьма большой. В этом случае руководствуются *следующими правилами*:

а) следует по возможности избегать вычитания двух почти равных приближенных чисел, преобразуя соответствующим образом математические выражения (примеры подобных преобразований приведены в п.2.6);

б) если же приходится вычитать такие числа, то следует уменьшаемое и вычитаемое брать с достаточным запасом верных знаков (если такая возможность имеется); например, если известно, что при вычитании чисел a_1 и a_2 первые m их значащих цифр пропадут, а результат необходимо иметь с n верными значащими цифрами, то следует взять числа a_1 и a_2 с $m+n$ верными значащими цифрами.

3. В теории погрешностей показано, что при вычислении ***произведения*** небольшого числа сомножителей (порядка десяти) число верных знаков произведения на одну или две единицы меньше числа верных знаков в наименее точном из сомножителей. Отсюда вытекает следующее *правило вычисления произведения нескольких чисел* различной точности:

а) выделяют число с наименьшим количеством верных значащих цифр;

б) оставшиеся сомножители округляют таким образом, чтобы каждый из них содержал на одну (или две) значащую цифру больше, чем количество верных цифр в выделенном числе;

в) в произведении сохраняют столько значащих цифр, сколько верных цифр имеет наименее точный из сомножителей (выделенное число).

Деление чисел производят по аналогичному правилу.

4. ***При возведении в степень*** приближенного числа в результате следует сохранить столько значащих цифр, сколько верных знаков имеет основание степени.

5. ***При извлечении корня*** из приближенного числа в результате следует брать столько значащих цифр, сколько верных цифр имеет подкоренное число.

Приведенные правила можно обобщить следующим образом:

- при сложении, вычитании, умножении и делении приближенных чисел в результате следует сохранять столько значащих цифр, сколько их имеет приближенное число с наименьшим количеством верных цифр;
- если некоторые данные имеют излишние десятичные младшие разряды (при сложении или вычитании) или больше значащих цифр, чем другие (при умножении и делении), то их предварительно следует округлить, сохраняя одну запасную цифру.

При выполнении ***массовых вычислений*** руководствуются следующими *дополнительными правилами*:

- для уменьшения накопления погрешностей во всех *промежуточных* результатах следует сохранять на одну цифру больше, чем рекомендуют предыдущие правила;
- в окончательном результате эта «запасная» цифра отбрасывается.

2.5. Погрешности округления при выполнении арифметических операций в ЭВМ

В этом разделе изучим возникновение погрешностей округления с учетом особенностей выполнения арифметических операций на ЭВМ.

2.5.1. Сложение двух чисел

Рассмотрим вначале образование ошибок округления при сложении в ЭВМ двух чисел. Пусть a^* и b^* - точные значения этих чисел, $c^* = a^* + b^*$ - точное значение их суммы. Как отмечалось в п.2.2, из-за конечной разрядной сетки числа в ЭВМ представляются с определенной погрешностью. Пусть a – округленное (записанное в память ЭВМ) значение числа a^*, $\Delta(a)$ – погрешность округления числа a. Тогда

$$a^* = a + \Delta(a). \qquad (2.27)$$

Аналогично для числа b^*

$$b^* = b + \Delta(b). \qquad (2.28)$$

Так как в памяти ЭВМ вместо точных значений чисел a^* и b^* хранятся их округленные значения a и b, то в операции суммирования будут участвовать именно последние величины. Их сумма равна:

$$c' = a + b. \qquad (2.29)$$

Однако результат c' операции суммирования может опять не войти в разрядную сетку и будет записан в память ЭВМ как округленное число c с ошибкой округления $\Delta_{op}(c)$:

$$c' = c + \Delta_{op}(c). \qquad (2.30)$$

Специальное обозначение подчеркивает, что природа появления ошибки Δ_{op} связана с *округлением результата*.

На основании (2.28)-(2.30) можно записать цепочку соотношений:

$$c^* = a^* + b^* = a + \Delta(a) + b + \Delta(b) = c' + \Delta(a) + \Delta(b) =$$
$$= c + \Delta(a) + \Delta(b) + \Delta_{op}(c). \qquad (2.31)$$

Окончательно, из (2.31) найдем абсолютную погрешность при сложении на ЭВМ двух чисел:

$$\Delta(c) = c^* - c = \Delta(a) + \Delta(b) + \Delta_{op}(c). \qquad (2.32)$$

Итак, результирующая ошибка состоит из ошибок представления в ЭВМ исходных чисел $\Delta(a)$, $\Delta(b)$ и ошибки округления результата $\Delta_{op}(c)$.

Приведем пример.

Пример 2.5. Погрешность операции суммирования двух чисел в ЭВМ

Пусть точные значения суммируемых чисел с плавающей запятой равны $a^* = 0{,}92687 \cdot 10^1$ и $b^* = 0{,}71679 \cdot 10^1$. Точное значение суммы равно $c^* = a^* + b^* = 0{,}164366 \cdot 10^2$.

Предположим, что операция суммирования выполняется в ЭВМ с 4 десятичными разрядами. Тогда на первом шаге исходные числа a^* и b^* будут округлены с погрешностями $\Delta(a)$ и $\Delta(b)$:

$$a = 0{,}9268 \cdot 10^1;\ \Delta(a) = 0{,}7 \cdot 10^{-3};$$
$$b = 0{,}7167 \cdot 10^1;\ \Delta(b) = 0{,}9 \cdot 10^3.$$

Сложение округленных чисел a и b дает результат:

$$c' = a + b = 0{,}16435 \cdot 10^2.$$

После округления числа c' с погрешностью $\Delta_{\text{ор}}(c)$ окончательно будет получено значение суммы чисел c:

$$c = 0{,}1643 \cdot 10^2;\ \Delta_{\text{ор}}(c) = 5 \cdot 10^{-3}.$$

Результирующая ошибка вычисления суммы равна $\Delta(c) = c^* - c = 6{,}6 \cdot 10^{-3}$ и складывается из следующих погрешностей:

$$\Delta(c) = \Delta(a) + \Delta(b) + \Delta_{\text{ор}}(c) = 0{,}7 \cdot 10^{-3} + 0{,}9 \cdot 10^{-3} + 5 \cdot 10^{-3}.$$

Очевидно, в данном случае погрешность из-за округления результата $\Delta_{\text{ор}}(c)$ является преобладающей.

Возможна ситуация, когда числа a^* и b^* представляются в ЭВМ точно (т.е. $\Delta(a) = \Delta(b) = 0$), однако их сумма не «входит» в разрядную сетку. Пусть, например, складываются числа $a^* = 0{,}1624 \cdot 10^3$ и $b^* = 0{,}1769 \cdot 10^{-1}$ на ЭВМ с 4 десятичными разрядами, в этом случае $a = a^*$, $b = b^*$ и $\Delta(a) = \Delta(b) = 0$. При сложении на ЭВМ предварительно выравниваются порядки чисел до большего, а затем мантиссы суммируются. В результате имеем:

$$c' = a + b = 0{,}1624 \cdot 10^3 + 0{,}001769 \cdot 10^3 = 0{,}164169 \cdot 10^3.$$

Последнее число будет округлено до значения $c = 0{,}1641 \cdot 10^3$, погрешность операции суммирования $\Delta(c) = \Delta_{\text{ор}}(c) = 0{,}069$.

Заметим, что рассмотренная ситуация с $\Delta(a) = \Delta(b) = 0$ нехарактерна, так как исходные числа a^* и b^* обычно являются результатом предыдущих машинных операций и, как правило, должны округляться.

Формула (2.32) выведена для случая, когда известны точные значения суммируемых чисел и их погрешности. На практике чаще приходится оценивать предельные погрешности вычислений. Предельная абсолютная погрешность при сложении двух чисел на ЭВМ определяется из выражения (2.32), если в нем составляющие ошибки заменить на соответствующие предельные величины:

$$\overline{\Delta}(c) = \overline{\Delta}(a + b) = \overline{\Delta}(a) + \overline{\Delta}(b) + \overline{\Delta}_{\text{ор}}(c). \qquad (2.33)$$

В частности, для примера 2.5 предельные погрешности $\overline{\Delta}(a)$, $\overline{\Delta}(b)$ и $\overline{\Delta}_{\text{ор}}(c)$ равны единице последнего сохраняемого разряда: $\overline{\Delta}(a) = \overline{\Delta}(b) = 10^{-3}$; $\overline{\Delta}_{\text{ор}}(c) = 10^{-2}$. Из (2.33) найдем предельную абсолютную погрешность операции суммирования:

$$\overline{\Delta}(c) = 1 \cdot 10^{-3} + 1 \cdot 10^{-3} + 1 \cdot 10^{-2} = 1{,}2 \cdot 10^{-2}.$$

Как и следовало ожидать, предельная оценка погрешности $\overline{\Delta}(c)$ больше реальной погрешности $\Delta(c)=6{,}6\cdot10^{-3}$.

Получим теперь соотношение для предельной относительной погрешности операции суммирования двух чисел. Для этого воспользуемся выражением (2.33) и учтем, что

$$\overline{\Delta}(c)=\overline{\delta}(c)\cdot|\,c\,|=\overline{\delta}(c)\cdot|\,a+b\,|;$$
$$\overline{\Delta}_{\text{ор}}(c)=\overline{\delta}_{\text{ор}}(c)\cdot|\,c\,|=\varepsilon_{\text{маш}}|\,c\,|,$$

где $\overline{\delta}(c)$ – результирующая предельная относительная погрешность; $\overline{\delta}_{\text{ор}}(c)$ – предельная относительная погрешность за счет округления результата, которая, как было показано в п. 2.3, равна величине машинного эпсилона $\varepsilon_{\text{маш}}$. Таким образом, имеем:

$$\overline{\delta}(c)\cdot|\,a+b\,|=\overline{\Delta}(a)+\overline{\Delta}(b)+\varepsilon_{\text{маш}}\cdot|\,c\,|$$

или

$$\overline{\delta}(c)=\overline{\delta}(a+b)=\frac{\overline{\Delta}(a)+\overline{\Delta}(b)}{|\,a+b\,|}+\varepsilon_{\text{маш}}. \qquad (2.34)$$

Аналогично может быть получена формула для предельной относительной погрешности при вычислении на ЭВМ разности двух чисел:

$$\overline{\delta}(a-b)=\frac{\overline{\Delta}(a)+\overline{\Delta}(b)}{|\,a-b\,|}+\varepsilon_{\text{маш}}. \qquad (2.35)$$

Полезным является также вариант формулы (2.34), когда относительная погрешность суммы выражается через о*тносительную* погрешность исходных величин. Учитывая, что $\overline{\Delta}(a)=\overline{\delta}(a)\cdot|\,a\,|$ и $\overline{\Delta}(b)=\overline{\delta}(b)\cdot|\,b\,|$, из (2.34) найдем:

$$\overline{\delta}(c)=\overline{\delta}(a+b)=\frac{|\,a\,|}{|\,a+b\,|}\overline{\delta}(a)+\frac{|\,b\,|}{|\,a+b\,|}\overline{\delta}(b)+\varepsilon_{\text{маш}}. \qquad (2.36)$$

Сравним полученные формулы (2.33)-(2.36) с их аналогами (2.17) и (2.18) в классической теории погрешностей. Как видно, в (2.33) и (2.34)-(2.36) появились добавки соответственно $\overline{\Delta}_{\text{ор}}(c)$ и $\varepsilon_{\text{маш}}$, обусловленные округлением результата. Эти добавки, не учитываемые в классической теории, связаны со спецификой вычислений в ЭВМ. Пример 2.5 показывает, что они могут быть весьма существенными.

Указанные добавки меняют и качественное соотношение погрешностей. В классической теории на основании рассмотрения формулы (2.18) утверждается, что предельная относительная погрешность арифметической суммы двух чисел не может превышать соответствующей погрешности каждого из слагаемых (см. соотношение (2.19)). Однако формула (2.36) свидетельствует о том, что в реальных вычислениях на ЭВМ из-за добавки $\varepsilon_{\text{маш}}$ величина $\overline{\delta}(a+b)$ может быть больше любой из величин $\overline{\delta}(a)$ и $\overline{\delta}(b)$.

Формула (2.35) показывает, что при вычислении на ЭВМ разности двух близких чисел может наблюдаться катастрофическое ухудшение точности, это совпадает с выводом классической теории погрешностей.

Замечание. *Для уменьшения влияния погрешностей округления во многих ЭВМ арифметические операции выполняются с увеличенным (по сравнению со стандартным) количеством разрядов, окончательные результаты затем округляются до стандартной длины.*

2.5.2. Суммирование последовательности чисел

Рассмотрим погрешности при вычислении на ЭВМ суммы нескольких чисел:

$$S = a_1 + a_2 + \ldots + a_n. \qquad (2.37)$$

Классические формулы (2.14), (2.15) для предельных абсолютной и относительной погрешностей суммы симметричны относительно каждого из слагаемых a_i. Иначе говоря, числа a_i могут быть перенумерованы любым образом, а формулы (2.14), (2.15) от этого не изменятся. Это означает, что в классической теории предельные погрешности не зависят от порядка суммирования чисел.

При вычислениях на ЭВМ расчет суммы производится поэтапно, обычно в форме следующего цикла:

S: = a_1;
для i: = 2 шаг 1 до n цикл
начало
S: = $S + a_i$;
конец *цикла*.

В результате получаем следующую цепочку вычисления частичных сумм S_i, $i = \overline{1,n}$:

$$S_1 = a_1;$$
$$S_2 = S_1 + a_2 = a_1 + a_2;$$
$$S_3 = S_2 + a_3 = a_1 + a_2 + a_3;$$
$$\ldots \quad \ldots \quad \ldots \quad \ldots \quad \ldots$$
$$S = S_n = S_{n-1} + a_n = a_1 + a_2 + \ldots + a_n. \qquad (2.38)$$

Заметим, что, в отличие от записи (2.37), в (2.38) однозначно определяется порядок суммирования чисел.

Рассмотрим вначале сложение трех чисел:

$$S = a_1 + a_2 + a_3.$$

В этом случае цепочка (2.38) будет включать две операции суммирования:

$$S_2 = a_1 + a_2; \tag{2.39}$$

$$S = S_2 + a_3. \tag{2.40}$$

Применим к каждой из этих операций формулу (2.36) для предельной относительной погрешности суммы двух чисел. Для операции суммирования (2.39):

$$\bar{\delta}_2(S) = \frac{|a_1|}{|S_2|}\bar{\delta}(a_1) + \frac{|a_2|}{|S_2|}\bar{\delta}(a_2) + \varepsilon_{\text{маш}}. \tag{2.41}$$

Для операции суммирования (2.40):

$$\bar{\delta}(S) = \frac{|S_1|}{|S|}\bar{\delta}(S_2) + \frac{|a_3|}{|S|}\bar{\delta}(a_3) + \varepsilon_{\text{маш}}. \tag{2.42}$$

Подставим соотношение (2.41) для $\bar{\delta}(S_2)$ в формулу (2.42), после некоторых преобразований получим предельную относительную погрешность вычисления на ЭВМ суммы трех чисел:

$$\bar{\delta}(S) = \frac{|a_1|}{|S|}\bar{\delta}(a_1) + \frac{|a_2|}{|S|}\bar{\delta}(a_2) + \frac{|a_3|}{|S|}\bar{\delta}(a_3) + \varepsilon_{\text{маш}} \cdot \left[1 + \frac{|a_1 + a_2|}{|S|}\right]. \tag{2.43}$$

Первая часть формулы (2.43), включающая три члена вида $\frac{|a_i|}{|S|}\bar{\delta}(a_i)$, симметрична относительно каждого из чисел a_i и совпадает с классической формулой для случая $n = 3$ (см. (2.16)). Таким образом, эта составляющая общей погрешности вычислений описывает ошибки представления исходных чисел a_i и не зависит от порядка их суммирования.

Последний член формулы (2.43) обусловлен округлением частичных сумм S_2 и S в процессе вычислений и несимметричен относительно чисел a_i. Его анализ приводит к интересному выводу: результирующая погрешность суммы трех чисел $\bar{\delta}(S)$ минимальна, если частичная сумма первых двух складываемых чисел является наименьшей по модулю, т.е. если $|a_1 + a_2| \le |a_1 + a_3|$ и $|a_1 + a_2| \le |a_2 + a_3|$. В частности, при положительных числах a_1, a_2, a_3 погрешность суммы $\bar{\delta}(S)$ минимальна, если $a_1 \le a_3$ и $a_2 \le a_3$, т.е. если сначала складывать наименьшие числа a_i.

Аналогично, методом индукции может быть выведена формула для предельной относительной погрешности вычисления на ЭВМ суммы n чисел:

$$\bar{\delta}(S) = \frac{1}{|S|}\sum_{i=1}^{n}|a_i|\,\bar{\delta}(a_i) + \varepsilon_{\text{маш}}\frac{|S_2| + |S_3| + \ldots + |S_n|}{|S|}. \tag{2.44}$$

Если все числа a_i одного знака, то формула (2.44) примет более наглядный вид:

$$\bar{\delta}(S) = \frac{1}{|S|}\sum_{i=1}^{n}|a_i|\,\bar{\delta}(a_i) + \varepsilon_{\text{маш}}\frac{|a_1(n-1) + a_2(n-1) + a_3(n-2) + \ldots + a_n|}{|S|}. \quad (2.45)$$

Анализ формул (2.44), (2.45) позволяет сделать ***следующие выводы***:

1. Относительная погрешность суммы увеличивается с ростом количества слагаемых и даже при числах одного знака может превысить относительную погрешность отдельных слагаемых.

2. Погрешность вычисления суммы зависит от порядка суммирования чисел. Ошибка $\bar{\delta}(S)$ минимальна, если вести суммирование в таком порядке, при котором частичные суммы являются наименьшими по абсолютной величине. В случае чисел одного знака ошибка $\bar{\delta}(S)$ минимальна, если начать суммирование с наименьших по модулю чисел a_i, т.е. если расположить (пронумеровать) числа в порядке возрастания их модуля:

$$|a_1| \le |a_2| \le |a_3| \le \ldots \le |a_n|.$$

Оба вывода отличаются от соответствующих положений классической теории погрешностей (см. формулы (2.16) и (2.19)). Причину неравноправности слагаемых a_i в образовании результирующей ошибки можно понять, рассмотрев итерационный процесс суммирования (2.38). Формально каждое слагаемое участвует в процессе суммирования лишь один раз. Однако в образовании ошибок каждое слагаемое участвует столько раз, сколько раз суммируются частичные суммы, зависящие от этого слагаемого. Сказанное ясно видно из формул (2.44) и (2.45).

Формулы (2.44) и (2.45) полезны для анализа, однако их сложно использовать на практике для определения ошибки $\bar{\delta}(S)$, так как требуется знать все числа a_i. Для приближенной оценки предельной относительной погрешности при вычислении на ЭВМ суммы n чисел можно использовать следующую простую формулу, справедливую, если все числа a_i $(i = \overline{1,n})$ одного знака:

$$\bar{\delta}(S) = \max_{i=\overline{1,n}} \bar{\delta}(a_i) + \varepsilon_{\text{маш}}\frac{n+2}{2}. \quad (2.46)$$

Из (2.46) следует, что относительная погрешность округления при вычислении суммы растет примерно пропорционально числу слагаемых.

Механизм влияния порядка суммирования чисел на величину погрешности суммы покажем на следующем примере.

Пример 2.6. Суммирование последовательности чисел.

Пусть требуется найти сумму пяти чисел:

S = 0,3 +0,5 +1,5 +25,9 + 1001,

точное значение суммы равно S = 1029,2.

Предположим, что суммирование выполняется в ЭВМ с 4 десятичными разрядами, после каждой операции результат округляется методом усечения. Все исходные числа точно представляются с помощью четырех десятичных разрядов. Проследим вначале вычисление на машине суммы чисел от наименьшего к наибольшему, т.е. в порядке их записи (округленное значение числа указывается после стрелки):

$$0{,}3 + 0{,}5 = 0{,}8;$$
$$0{,}8 + 1{,}5 = 2{,}3;$$
$$2{,}3 + 25{,}9 = 28{,}2;$$
$$28{,}2 + 1001 = 1029{,}2 \Rightarrow 1029.$$

Полученный результат S= 1029 совпадает с точным значением суммы, округленным до 4-х значащих цифр.

Изменим теперь порядок вычислений и начнем складывать числа от последнего к первому:

$$1001 + 25{,}9 = 1026{,}9 \Rightarrow 1026;$$
$$1026 + 1{,}5 = 1027,\ 5 \Rightarrow 1027;$$
$$1027 + 0{,}5 = 1027{,}5 \Rightarrow 1027;$$
$$1027 + 0{,}3 = 1027{,}3 \Rightarrow 1027.$$

В этом случае окончательный результат S= 1027, он менее точный.

Анализ процесса вычислений показывает следующее. Потеря точности в последнем случае происходит из-за того, что прибавления малых чисел к большому числу не происходит, так как они выходят за рамки разрядной сетки ($a+b = a$ при $a >> b$, т.е. величина b слишком мала для того, чтобы увеличить младший разряд числа a). Этих малых чисел может быть очень много, но на результат они не повлияют, поскольку прибавляются по одному. Ошибка вычисления суммы при этом может быть значительна.

Если сначала сложить малые числа, они «накопятся» и их суммарная величина станет достаточной для увеличения младшего разряда большого числа.

Из приведенного анализа вытекает правило, важное для практической организации вычислений: *для повышения точности при вычислении алгебраической суммы последовательности чисел суммирование нужно начинать с наименьших чисел***.**

Материал настоящего раздела приводит также к более общему выводу. Если погрешность результата зависит от порядка суммирования, тогда при сложении чисел в разном порядке получим разные результаты, т.е., например,

$$a + b + c \neq c + b + a.$$

Иначе говоря, при машинных вычислениях не выполняется свойство коммутативности операции сложения, являющееся основополагающим в математике. Этот вывод будет развит в п. 2.5.6.

2.5.3. Умножение двух чисел

Выведем формулу для погрешностей при вычислении на ЭВМ произведения двух чисел. Пусть, как и ранее, a^* и b^* – точные значения сомножителей, a и b – округленные значения, тогда

$$a^* = a + \Delta(a),\ b^* = b + \Delta(b), \qquad (2.47)$$

где $\Delta(a)$ и $\Delta(b)$ – погрешности округления сомножителей. Через $c^* = a^* b^*$ обозначим точное значение произведения.

В операции умножения будут участвовать округленные числа a и b, их произведение равно:

$$c' = ab. \qquad (2.48)$$

В общем случае результат c' операции умножения из-за конечной разрядной сетки будет округлен до значения c с ошибкой $\Delta_{op}(c)$, т.е.:

$$c' = c + \Delta_{op}(c). \qquad (2.49)$$

Учитывая (2.47)-(2.49), найдем:

$$c^* = a^* b^* = (a + \Delta(a))\,(b + \Delta(b)) = c' + a\,\Delta(b) + b\,\Delta(a) + \Delta(a)\Delta(b) =$$
$$= c + \Delta_{op}(c) + a\,\Delta(b) + b\,\Delta(a) + \Delta(a)\Delta(b). \qquad (2.50)$$

Из (2.50), пренебрегая малым членом $\Delta(a)\Delta(b)$, получим абсолютную погрешность произведения двух чисел:

$$\Delta(c) = c^* - c \approx a\,\Delta(b) + b\,\Delta(a) + \Delta_{op}(c). \qquad (2.51)$$

Для определения относительной погрешности разделим (2.51) на c:

$$\delta(c) = \frac{c^* - c}{c} = \frac{a\Delta(b)}{c} + \frac{b\Delta(a)}{c} + \delta_{op}(c),$$

где $\delta_{op}(c) = \Delta_{op}(c)/c$.

Будем полагать $\Delta_{op}(c)$ малой величиной, тогда $c^* \approx c' = ab$ и

$$\frac{a\Delta(b)}{c} \approx \frac{a\Delta(b)}{ab} = \frac{\Delta(b)}{b} = \delta(b);$$

$$\frac{b\Delta(a)}{c} \approx \frac{b\Delta(a)}{ab} = \frac{\Delta(a)}{a} = \delta(a).$$

Окончательно найдем:

$$\delta(c) = \delta(a) + \delta(b) + \delta_{op}(c). \qquad (2.52)$$

Формулу для предельной относительной погрешности можно получить, если в (2.52) перейти к предельным величинам составляющих ошибок и учесть, что $\overline{\delta}_{\text{ор}}(c) = \varepsilon_{\text{маш}}$:

$$\overline{\delta}(c) = \overline{\delta}(a) + \overline{\delta}(b) + \varepsilon_{\text{маш}}. \qquad (2.53)$$

Формула (2.53) отличается от соответствующего соотношения (2.23) в классической теории погрешностей. Как и в случае операции сложения, при вычислении на ЭВМ произведения двух чисел к ошибкам, обусловленным приближенным представлением (округлением) исходных чисел, добавляется ошибка $\varepsilon_{\text{маш}} = \overline{\delta}_{\text{ор}}(c)$, связанная с округлением результата.

2.5.4. Умножение последовательности чисел

Изучим погрешности при умножении на ЭВМ последовательности чисел:

$$P = a_1 a_2 \ldots a_n. \qquad (2.54)$$

Как и при суммировании, вычисление произведения нескольких чисел на ЭВМ выполняется в форме цикла:

P: = a_1;
для i: = 2 шаг 1 до n цикл
начало
    P: = $P \cdot a_i$;
конец цикла.

Таким образом, имеем цепочку вычисления частичных произведений $P_i,\ i = \overline{1,n}$:

$$\begin{aligned} P_1 &= a_1; \\ P_2 &= P_1 a_2 = a_1 a_2; \\ P_3 &= \mathrm{P}_2 a_3 = a_1 a_2 a_3; \\ &\ldots \quad \ldots \quad \ldots \quad \ldots \quad \ldots \\ P_n &= P_{n-1} a_n = a_1 a_2 \ \ldots a_n. \end{aligned} \qquad (2.55)$$

Рассмотрим вначале умножение трех чисел: $P = a_1 a_2 a_3$.

Цепочка включает две операции умножения:

$$P_2 = a_1 a_2; \qquad (2.56)$$

$$P = P_2 a_3. \qquad (2.57)$$

К каждой из операций (2.56), (2.57) применим формулу (2.53) для предельной относительной погрешности произведения двух чисел:

$$\overline{\delta}(P_2) = \overline{\delta}(a_1) + \overline{\delta}(a_2) + \varepsilon_{\text{маш}}; \qquad (2.58)$$

$$\overline{\delta}(P) = \overline{\delta}(P_2) + \overline{\delta}(a_3) + \varepsilon_{\text{маш}}. \qquad (2.59)$$

Подставив в (2.59) величину $\bar{\delta}(P_2)$ из уравнения (2.58), найдем предельную относительную погрешность при умножении трех чисел:

$$\bar{\delta}(P) = \bar{\delta}(a_1) + \bar{\delta}(a_2) + \bar{\delta}(a_3) + 2\varepsilon_{\text{маш}}. \qquad (2.60)$$

Аналогично методом индукции может быть получена формула для предельной относительной погрешности вычисления на ЭВМ произведения n чисел:

$$\bar{\delta}(P) = \sum_{i=1}^{n} \bar{\delta}(a_i) + (n-1)\,\varepsilon_{\text{маш}}. \qquad (2.61)$$

Анализ формулы (2.61) приводит к ***следующим выводам***:

1. Погрешность произведения n чисел состоит из погрешностей представления в ЭВМ (округления) исходных чисел и погрешностей округления n–1 частичных произведений.

2. Относительная погрешность произведения увеличивается с ростом числа сомножителей, даже если исходные числа точно представляются в разрядной сетке ЭВМ.

Заметим, что, в отличие от операции суммирования, в формуле (2.61) сомножители a_i равноправно участвуют в образовании ошибок (формула симметрична относительно всех чисел a_i). Это означает, что при вычислении на ЭВМ предельная относительная погрешность произведения последовательности чисел не зависит от порядка перемножения этих чисел. Однако сказанное относится лишь к предельной (максимально возможной) величине погрешности. На значения погрешности, реально получающиеся при умножении в ЭВМ нескольких чисел, порядок перемножения будет все равно влиять. Таким образом, *операция умножения на ЭВМ также некоммутативна,* т.е., например, $abc \neq cba$.

В противоположность операции суммирования, в общем случае нельзя дать рекомендации по выбору порядка сомножителей, при котором погрешность вычисления произведения будет минимальной.

2.5.5. Алгоритм перемножения последовательности чисел и масштабирование величин

Выше при выводе формул (2.60) и (2.61) предполагалось, что при перемножении не возникает чисел, меньших машинного нуля или больших машинной бесконечности. Однако может оказаться, что на каком-то этапе вычислений в качестве промежуточного результата будет получен либо машинный нуль M_0, либо машинная бесконечность M_∞. Оба указанных случая приводят к неверному окончательному результату, поэтому необходимо изменить вычислительный алгоритм. Оказывается, что здесь существенным является порядок перемножения чисел. Приведем пример.

Пример 2.7. Вычисление произведения последовательности чисел.

Пусть необходимо вычислить произведение чисел $a_1 = 10^{-21}$, $a_2 = 10^{-20}$, $a_3 = 10^{20}$ и $a_4 = 10^{21}$. Очевидно, значение произведения равно

$$P = a_1 a_2 a_3 a_4 = 10^{-21} \cdot 10^{-20} \cdot 10^{20} \cdot 10^{21} = 1.$$

Предположим, что вычисления выполняются на ЭВМ, использующей 32-разрядную арифметику, при представлении чисел с плавающей запятой. Тогда, как указывалось в п. 2.2, значение машинного нуля равно $M_0 = 10^{-38}$, а машинной бесконечности – $M_\infty = 10^{38}$. Каждое из чисел a_i принадлежит допустимому диапазону значений $[\, M_0, M_\infty \,]$.

Рассмотрим вычисление произведения при различном порядке сомножителей.

1. Числа перемножаются в порядке нумерации, т.е. $P = a_1 a_2 a_3 a_4$. В этом случае уже после первой операции перемножения будет получен машинный нуль, в дальнейшем результат не изменится.

$$P_2 = a_1 a_2 = 10^{-21} \cdot 10^{-20} = 10^{-41} \Rightarrow P_2 = 0;$$
$$P_3 = P_1 a_3 = 0 \cdot 10^{20} = 0;$$
$$P = P_3 a_4 = 0 \cdot 10^{21} = 0.$$

Итак, получен неверный результат $P = 0$.

2. Числа перемножаются в обратном порядке: $P = a_4 a_3 a_2 a_1$. Здесь после первой операции умножения имеет место переполнение разрядной сетки:

$$P_2 = a_4 a_3 = 10^{21} \cdot 10^{20} = 10^{41} \Rightarrow P_2 = \text{OFL}.$$

3. Верный результат получится, если числа перемножить, например, в таком порядке: $P = a_1 a_4 a_2 a_3$. В этом случае имеем:

$$P_2 = a_1 a_4 = 10^{-21} \cdot 10^{21} = 1;$$
$$P_3 = P_2 a_2 = 1 \cdot 10^{-20} = 10^{-20};$$
$$P = P_3 a_3 = 10^{-20} \cdot 10^{20} = 1.$$

Итак, в рассмотренном примере получение верного результата зависит от порядка перемножения чисел.

Приведем ***алгоритм вычисления произведения***, при котором в случае произвольного числа n сомножителей на промежуточных этапах не возникает машинного нуля или машинной бесконечности.

Шаг 1. В последовательности (2.54) перенумеруем сомножители таким образом, чтобы $| a_1 | \le | a_2 | \le \ldots \le | a_n |$.

Предполагается, что $| a_1 | \le 1$, $| a_n | \ge 1$.

Шаг 2. Вычисляем произведение чисел в порядке $\Pi_1 = a_1 a_n a_{n-1} a_{n-2} \ldots$ до тех пор, пока впервые не получим $| \Pi_1 | > 1$.

Шаг 3. Полученное частичное произведение Π_1 далее умножаем последовательно на a_2, a_3 и т.д., т.е. вычисляем произведение $\Pi_2 = \Pi_1 a_2 a_3 \ldots$ до тех пор, пока не получим $| \Pi_2 | < 1$.

Шаг 4. Шаги 1 и 2 повторяются до тех пор, пока все оставшиеся сомножители будут только большими единицы по модулю ($|a_i| > 1$), либо только меньшими ($|a_i| < 1$). Далее умножение производится в произвольном порядке.

Рассмотрим применение алгоритма для перемножения чисел в примере 2.7. Здесь числа уже пронумерованы в нужном порядке. Далее имеем следующую последовательность операций.

Шаг 2: $\Pi_1 = a_1 a_4 = 10^{-21} \cdot 10^{21} = 1$;

$\Pi_1 = a_1 a_4 a_3 = 10^{-21} \cdot 10^{21} \cdot 10^{20} = 10^{20} > 1$.

Шаг 3: $\Pi_2 = \Pi_1 a_2 = 10^{20} \cdot 10^{-20} = 1$.

На этом работа алгоритма прекращается, получили верный результат $P = \Pi_2 = 1$.

Еще один прием может использоваться, если все сомножители a_i слишком малы или слишком велики по сравнению с единицей. Такая ситуация встречается в технических расчетах, когда параметры технических устройств указываются непосредственно в удобных для инженера физических единицах измерения. (Например, частота $f = 10^9$ Гц, емкость $C = 10^{-12}$ Ф и т.д.) В этом случае исходные числа a_i масштабируют (нормируют), т.е. умножают или делят на подходящий постоянный множитель. После вычисления произведения нормированных чисел результат денормируют. Рассмотрим пример.

Пример 2.8. Масштабирование величин.

Пусть необходимо вычислить произведение чисел $a_1 = 1 \cdot 10^{-12}$, $a_2 = 2 \cdot 10^{-12}$, $a_3 = 3 \cdot 10^{-12}$, $a_4 = 4 \cdot 10^{-12}$. Очевидно, значение произведения

$$P = a_1 a_2 a_3 a_4 = 24 \cdot 10^{-48}$$

меньше машинного нуля в 32-разядной арифметике, т.е. непосредственное перемножение чисел приведет к неверному результату $P = 0$.

Выполним вначале нормировку исходных чисел в соответствии с соотношением $a_i' = a_i \cdot C$, где C – постоянный коэффициент. Удобно выбрать $C = 10^{12}$, в этом случае нормированные сомножители равны $a_1' = 1$, $a_2' = 2$, $a_3' = 3$, $a_4' = 4$. Легко установить связь произведения исходных чисел P с произведением нормированных чисел $P' = a_1' a_2' a_3' a_4'$:

$$P = a_1 a_2 a_3 a_4 = \frac{1}{C^4} a_1' a_2' a_3' a_4' = \frac{P'}{C^4}. \qquad (2.62)$$

Теперь найдем на ЭВМ произведение нормированных сомножителей a_i', это не связано с вычислительными трудностями:

$$P' = a_1' a_2' a_3' a_4' = 1 \cdot 2 \cdot 3 \cdot 4 = 24.$$

В заключение, выполнив разнормировку в соответствии с (2.62), найдем истинное значение произведения чисел:

$$P = \frac{P'}{C^4} = \frac{24}{10^{48}} = 24 \cdot 10^{-48}.$$

Обычно операции нормировки исходных чисел и денормировки результата выполняются самим пользователем. Заметим, что изменение порядка действий и масштабирование величин могут использоваться не только при вычислении произведения чисел. Эти приемы часто применяются при построении разнообразных алгоритмов и позволяют при вычислениях исключить появление машинного нуля и переполнения разрядной сетки.

2.5.6. Зависимость погрешности вычислений от порядка выполнения операций. Правила выполнения арифметических операций в ЭВМ

В этом разделе подведем некоторые итоги. В математике операции сложения (вычитания) и умножения (деления) являются коммутативными, ассоциативными и связанными между собой законом дистрибутивности. Напомним эти понятия.

***Коммутативный закон для сложения*:**

$a + b = b + a$.

Отсюда, например, вытекает, что

$a + b + c = a + c + b = b + a + c = \ldots = c + b + a$.

***Ассоциативный (сочетательный) закон для сложения*:**

$(a + b) + c = a + (b + c)$.

Коммутативный закон для умножения:

$ab = ba$,

отсюда следует

$abc = acb = bac = \ldots = cba$.

Ассоциативный закон для умножения:

$(a \cdot b) \cdot c = a \cdot (b \cdot c)$.

Дистрибутивный закон:

$a(b + c) = ab + ac$.

Из этих законов вытекает ***важное следствие***: вычисление арифметических выражений может осуществляться в любом порядке, т.е. независимо от способа записи выражения и расстановки скобок получим один и тот же результат.

В предыдущих разделах (п.2.5.2, п.2.5.4) было показано, что при вычислениях на ЭВМ вследствие ошибок округления операции сложения (вычитания) и умножения (деления) указанным законам не подчиняются, т.е.,

например, $a+b+c \neq c+b+a$, $a\ b\ c \neq c\ b\ a$ и т.д. Таким образом, какими бы малыми ни были ошибки округления, их появление существенно меняет математические свойства самих операций.

Из сказанного следуют ***два практических вывода***:

1. *При вычислениях на ЭВМ порядок операций и расстановки скобок в арифметических выражениях могут существенно влиять на результат*.

2. *Оптимизируя порядок операций, можно улучшить точность вычисления арифметических выражений*.

Приведем пример.

Пример 2.9. Вычисление арифметического выражения.

Пусть требуется вычислить значение выражения

$$K = (a-b)\,c, \tag{2.63}$$

где $a = 0{,}6382$, $b = 0{,}6371$, $c = 0{,}9364\cdot 10^2$. Вычисления выполняются с 4 десятичными разрядами, результаты операций округляются методом усечения. Нетрудно убедиться, что точное значение K, округленное до 4-х значащих цифр, равно $K = 0{,}1030$.

Вычислим K в соответствии с записью (2.63), т.е. вначале выполним операцию вычитания, затем – умножения. Получим:

$$a-b = 0{,}1100\cdot 10^{-2};$$

$$K_1 = (a-b)\,c = 0{,}1100\cdot 10^{-2}\cdot 0{,}9364\cdot 10^2 = 0{,}10300|040 \Rightarrow 0{,}1030,$$

где после знака | следуют отбрасываемые цифры, знак $\Rightarrow$ означает операцию округления. Итак, величина K_1 с точностью до 4-х знаков совпадает с точным значением K.

Теперь раскроем в (2.63) скобки и вычислим K в соответствии с выражением

$$K = ac - bc, \tag{2.64}$$

т.е. вначале найдем произведения ac и bc и затем выполним их вычитание. В результате определим:

$$ac = 59{,}76|1048 \Rightarrow 59{,}76;$$

$$bc = 59{,}65|8044 \Rightarrow 59{,}65;$$

$$K_2 = ac - bc = 59{,}76 - 59{,}65 = 0{,}1100.$$

Относительная ошибка вычисления K по формуле (2.64) составляет около 7%.

Итак, результаты вычислений по формулам (2.63), (2.64) отличаются друг от друга, т.е. при вычислениях на ЭВМ $(a-b)\,c \neq ac - bc$. При этом вычисления по формуле (2.63) гораздо точнее. Анализ показывает, что в первом случае (формула (2.63)) разность чисел a и b входит в разрядную сетку, т.е. вычисляется точно. Поэтому и величина K_1 находится с хорошей точностью. Во втором случае (формула (2.64)) произведения ac и bc не входят в разрядную сетку и округляются, это приводит к погрешности вычисления K_2. Кроме того, вычисления по формуле (2.64) требуют трех операций, тогда как по формуле (2.63) – только двух.

К аналогичным выводам приводит анализ вычисления выражения $K = (a - b)/c$. При положительных a, b, c и $a \approx b$ вычисление выражения по формуле $K_2 = a/c - b/c$ может привести к гораздо большей погрешности, чем использование исходной записи $K_1 = (a - b)/c$.

В заключение на основании рассмотренного материала сформулируем ***общие правила выполнения арифметических операций в ЭВМ***:

1. Сложение и вычитание последовательности чисел следует начинать с *наименьших* по модулю чисел.

2. Если возможно, алгоритм следует преобразовать таким образом, чтобы избежать *вычитания близких чисел* (один из примеров такого преобразования приведен в п.2.6).

3. При вычислении выражений вида $(a - b)\,c$ и $(a - b)/c$ в случае близких значений a и b нужно сначала выполнить операцию вычитания, а затем – операцию умножения или деления. Это приводит к меньшим ошибкам округления, чем при раскрытии скобок (см. пример 2.9).

4. Необходимо сводить *к минимуму* общее число операций.

2.5.7. Статистический характер погрешностей округления

На основании формул (2.46) и (2.61) можно получить оценку зависимости погрешности вычисления суммы и произведения от числа n слагаемых или сомножителей, т.е. фактически от числа операций сложения или умножения:

$$\bar{\delta}(a_1 + a_2 + \ldots + a_n) \approx \frac{n}{2}\varepsilon_{\text{маш}}; \qquad (2.65)$$

$$\bar{\delta}(a_1\, a_2 \ldots a_n) \approx 2n\,\varepsilon_{\text{маш}}. \qquad (2.66)$$

Формулы (2.65) и (2.66) получены соответственно из (2.46) и (2.61) при следующих допущениях: 1) количество слагаемых или сомножителей достаточно велико; 2) при вычислении суммы слагаемые a_i одного знака; 3) при вычислении произведения предельные относительные погрешности $\bar{\delta}(a_i)$ округления исходных чисел a_i приняты равными $\varepsilon_{\text{маш}}$ (см. п.2.3). Из (2.65), (2.66) вытекает, что при сложении и умножении предельная относительная погрешность результата примерно *пропорциональна числу операций*.

Следует отметить, однако, что формулы (2.65), (2.66) характеризуют именно предельные (максимально возможные) погрешности суммы и произведения. При попытке воспользоваться этими формулами для оценки реальных ошибок, возникающих при суммировании и умножении достаточно большого количества чисел, они дают слишком завышенные (пессимистические) значения.

В п.2.4.5 уже указывалось, что в классической теории погрешностей формулы для предельных погрешностей (п.2.4.1–2.4.3) выводятся в предположении наихудшего случая: считается, что все составляющие погрешности

(погрешности исходных данных) максимальные и имеют одинаковый знак, т.е. все числа отличаются от точных значений в одну сторону. В частности при оценке предельной погрешности суммы по формуле $\overline{\Delta}(a+b)=\overline{\Delta}(a)+\overline{\Delta}(b)$ предполагается, что погрешности $\overline{\Delta}(a)$ и $\overline{\Delta}(b)$ одного знака, т.е., например, одновременно $a < a^*$ и $b < b^*$. Получаемые таким образом оценки погрешностей являются гарантированными (т.е. реальная погрешность вычисления суммы $a + b$ никогда не превысит предельной величины $\overline{\Delta}(a+b)=\overline{\Delta}(a)+\overline{\Delta}(b)$), однако при большом количестве чисел очень грубыми.

В качестве примера применим формулу (2.66) для оценки допустимого числа операций умножения при использовании 32-разрядной арифметики. Пусть требуется вычислить произведение последовательности чисел с относительной погрешностью, не превышающей $\overline{\delta}(a_1\, a_2 \ldots a_n)$ = 1% = $= 1\cdot 10^{-2}$. Для 32-разрядной арифметики $\varepsilon_{маш}\approx 10^{-7}$, поэтому из формулы (2.66) получим максимально допустимое число операций умножения: $n \le \overline{\delta}(a_1\, a_2 \ldots a_n)/2\varepsilon_{маш} \approx 50000$. Между тем известно, что при вычислениях на современных ЭВМ число операций умножения может достигать миллионов и миллиардов. Таким образом, оценка по формуле (2.66) является слишком пессимистической.

Реально погрешности при вычислениях носят не детерминированный (полностью определенный), а статистический (случайный) характер. Поэтому при выполнении последовательности арифметических операций происходит частичная компенсация ошибок. Например, при вычислении суммы $a + b$ могут быть ситуации $a > a^*$ и $b < b^*$ или $a < a^*$ и $b > b^*$, в результате составляющие погрешности «гасят» друг друга. Как следствие, результирующая погрешность оказывается значительно меньше предсказываемой по формулам (2.65), (2.66).

Для более точной оценки погрешностей вычислений можно воспользоваться формулами, полученными в теории вероятности. Пусть складывается n чисел:

$$S = a_1 + a_2 + \ldots + a_n,$$

будем полагать, что предельные абсолютные погрешности всех слагаемых равны, т.е. $\overline{\Delta}(a_i)=\overline{\Delta}(a),\ i=\overline{1,n}$.

Аналогично, при вычислении произведения n чисел

$$P = a_1\, a_2 \ldots a_n,$$

полагаем равными предельные относительные погрешности всех сомножителей: $\overline{\delta}(a_i)=\overline{\delta}(a),\ i=\overline{1,n}$.

При сделанных предположениях классическая теория погрешностей дает следующие оценки предельных ошибок суммы и произведения (см. формулы (2.14) и (2.21)):

$$\overline{\Delta}(a_1 + a_2 + ... + a_n) = n\overline{\Delta}(a), \qquad (2.67)$$

$$\overline{\delta}(a_1\, a_2 \ldots a_n) = n\,\overline{\delta}(a). \qquad (2.68)$$

Как уже отмечалось, детерминированные оценки (2.67), (2.68) не учитывают случайного характера погрешностей и при больших n являются завышенными.

В теории вероятностей доказывается, что при n>10 в качестве оценок реальных погрешностей вычислений можно принять:

$$\widetilde{\Delta}(a_1 + a_2 + ... + a_n) = \sqrt{3n}\,\overline{\Delta}(a), \qquad (2.69)$$

$$\widetilde{\delta}(a_1\, a_2 \ldots a_n) = \sqrt{3n}\,\overline{\delta}(a), \qquad (2.70)$$

Формулы (2.69), (2.70) называются ***формулами Чеботарева***. В отличие от (2.67), (2.68), оценки (2.69), (2.70) носят *вероятностный*, а не гарантированный характер. Иначе говоря, погрешность результата будет меньше величины $\widetilde{\Delta}$ (или $\widetilde{\delta}$) с некоторой достаточно большой (но не равной 100%) вероятностью.

Формулы (2.67), (2.68) для предельных погрешностей суммы и произведения в классической теории по виду аналогичны формулам (2.65), (2.66), учитывающим ошибки округления результатов в ЭВМ, если принять в (2.67) $\overline{\Delta} = \overline{\delta}$, $\overline{\Delta}(a) = \varepsilon_{\text{маш}}/2$ и в (2.68) $\overline{\delta}(a) = 2\varepsilon_{\text{маш}}$. Это дает основание применить для оценки погрешностей вычислений на ЭВМ формулы Чеботарева (2.69) и (2.70) с учетом указанной замены величин:

$$\widetilde{\delta}(a_1 + a_2 + ... + a_n) = \frac{1}{2}\sqrt{3n}\,\varepsilon_{\text{маш}}; \qquad (2.71)$$

$$\overline{\delta}(a_1\, a_2 \ldots a_n) = 2\sqrt{3n}\,\varepsilon_{\text{маш}}\ . \qquad (2.72)$$

Формулы (2.71), (2.72) позволяют гораздо точнее оценить ошибки вычислений на ЭВМ при большом числе операций.

Приведем пример. Пусть перемножаются $n = 10^9$ чисел, т.е. выполняются $10^9 - 1$ операций умножения. Величина машинного эпсилона в 32-разрядной арифметике равна приблизительно $\varepsilon_{\text{маш}} \approx 1 \cdot 10^{-7}$. Формула (2.66) дает оценку относительной погрешности произведения:

$$\widetilde{\delta}(a_1\, a_2 \ldots a_n) = 2n\,\varepsilon_{\text{маш}} = 2 \cdot 10^9 \cdot 10^{-7} = 200 = 20000\%,$$

т.е. предсказывает, что при перемножении миллиарда чисел будет получен неправдоподобный результат. В то же время, применяя формулу Чеботарева (2.72), получим:

$$\overline{\delta}(a_1\, a_2 \ldots a_n) = 2\sqrt{3n}\,\varepsilon_{\text{маш}} = 2\sqrt{3 \cdot 10^9} \cdot 10^{-7} \approx 1{,}1 \cdot 10^{-2} = 1{,}1\%.$$

Это означает, что перемножение на ЭВМ миллиарда чисел может быть выполнено с приемлемой точностью. Последний результат гораздо ближе к действительности.

2.6. Организация вычислений: примеры

В предыдущих разделах мы определили основные источники погрешностей при вычислениях на ЭВМ и сформулировали некоторые правила вычислений. Однако вычислительные задачи очень разнообразны, поэтому универсальных рекомендаций по снижению ошибок, пригодных для любых типов задач, дать невозможно. Для получения малых погрешностей приходится подробно анализировать алгоритм и применять специальные приемы, которые могут быть индивидуальными для данной задачи или данного класса задач.

В приводимых ниже примерах мы проведем анализ ошибок вычислений и рассмотрим некоторые приемы их снижения.

Пример 2.10. Вычисление значений функции sin *x*.

Задача состоит в вычислении значений функции sin *x* с помощью ряда Тейлора:

$$\sin x = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \dots \ . \qquad (2.73)$$

Эта задача рассматривалась ранее в п. 1.7 (пример 1.6) при иллюстрации неустойчивых алгоритмов. Здесь мы подробно исследуем причину неустойчивости алгоритма, основанного на непосредственном использовании ряда (2.73).

Вычислим по формуле (2.73) $\sin x$ для нескольких значений x, отличающихся на число, кратное 2π: $x = \frac{\pi}{6} + 2\pi n$, где $n = 0, 1, 2, \dots, 8$. Очевидно, величина $\sin x$ для всех этих значений аргумента одинакова и равна $\sin x = 0{,}5$. Вычисления выполним с точностью 8 десятичных знаков, т.е. используем числа с одинарной точностью. Члены ряда, меньшие по модулю, чем 10^{-8}, будем отбрасывать.

Результаты вычислений приведены в табл. 2.1.

Таблица 2.1

Результаты вычислений функции sin *x*

n	*x*, °	*x*, радиан	sin *x*, одинарная точность	sin *x*, двойная точность	sin *x*, стандартная подпрограмма
0	30	0,52359878	0,49999999	0,49999999	0,49999999
1	390	6,8067841	0,49999993	0,50000006	0,50000005
2	750	13,089969	0,50013507	0,50000011	0,50000010
3	1110	19,373154	0,51655849	0,50000016	0,50000016
4	1470	25,656340	24,254018	0,50000143	0,50000010
5	1830	31,939525	14380,237	0,49953845	0,50000025
6	2190	38,222711	25902480	0,79868912	0,50000040
7	2550	44,505896	-130402500	29,539914	0,50000013
8	2910	50,789081	-83272283	-142982,02	0,50000029

Как видно, значение sin x для $x = 30°$ очень точно соответствует тому, что ожидалось, абсолютная погрешность составляет не более 10^{-8}. При увеличении угла ошибка растет, и при углах $x \geq 1470°$ вычисленное значение синуса получается совершенно бессмысленным.

Выполним анализ вычислений, для этого запишем несколько первых членов ряда (2.73) при $x = 1470°$:

$$\sin x = 25{,}656340 - 2789{,}0484 + 89849{,}610 - 1362035{,}9 + \ldots$$

Сумма первых двух членов ряда равна –2763,392060, при округлении до 8 значащих цифр имеем –2763,3920. При сложении, таким образом, потеряны последние две значащие цифры первого члена ряда. Так как отброшенные цифры далее восстановить невозможно, это означает, что ожидаемая погрешность вычисления sin x (суммы ряда) уже не может быть меньше 10^{-5}.

Суммирование первой частичной суммы –2763,3920 с третьим членом ряда дает 87086,2180, после округления 87086,218. На этом этапе значащие цифры суммы не теряются.

После суммирования с четвертым членом ряда получается –1274949,682, округление дает –1274949,6, и две последние значащие цифры третьей суммы снова потеряны. Число потерянных значащих цифр первого члена ряда возросло уже до пяти, т.е. ожидаемая ошибка вычисления суммы ряда составляет не менее 0,01.

Итак, картина постепенной потери точности становится ясной. Наибольший по модулю член ряда равен $55037680 \cdot 10^2$, после прибавления его к предыдущей сумме будут потеряны *все* значащие цифры первого члена ряда. Предполагаемая ошибка вычисления sin x теперь составляет не менее 10, это и приводит к бессмысленному результату!

Для следующего в таблице значения аргумента $x = 1830°$ наибольший член ряда составляет $0{,}26553689 \cdot 10^{13} \approx 2{,}7 \cdot 10^{12}$, что ведет к потере всех значащих цифр первого и второго членов ряда при суммировании.

Одна из причин погрешностей в данном случае состоит в том, что суммирование производится далеко не в лучшей последовательности. Как уже указывалось в п.2.5.2, лучше всего начинать сложение с наименьших членов или, что более правильно, вести сложение в таком порядке, чтобы частичные суммы были наименьшими по абсолютной величине.

К сожалению, эта причина не главная, и реализация правильной последовательности не позволит существенно уменьшить ошибку вычислений. Главная причина заключается в использовании при суммировании недостаточного количества значащих цифр (разрядов) чисел. Рассмотрим, например, максимальный по модулю член ряда Тейлора для $x = 1830°$, равный $0{,}26553689 \cdot 10^{13}$. Если написать это число без порядка, оно будет равно 2 655 368 900 000, причем нули не имеют никакого смысла, они просто по-

ставлены вместо тех цифр, которые не уместились в разрядной сетке ЭВМ. Очевидно, округленное значение данного члена ряда может отличаться от истинного на величину до 100000. Естественно, что при таких громадных ошибках совершенно безнадежно ожидать удовлетворительную точность окончательного результата, который наверняка не превысит 1.

Для того чтобы убедиться, что затруднения действительно происходят от недостаточного количества значащих цифр, можно произвести вычисления с удвоенной точностью, т.е. с 16 десятичными разрядами. Результаты таких вычислений приведены в табл. 2.1. Нетрудно убедиться, что погрешность определения $\sin x$ теперь достаточна мала, вплоть до значения аргумента $x = 1830°$. Однако при больших углах точность опять сильно падает.

Как уже отмечалось в п.1.7, в данном случае существует простой прием, позволяющий резко снизить ошибки вычислений. Вычитая из значения аргумента число, кратное 2π, можно привести угол к интервалу $[0; 2\pi]$. Далее, применяя известные тригонометрические формулы приведения углов, можно в конце концов получить интервал изменения x от 0 до $\frac{\pi}{4}$. Так как при этом приведенная величина угла $x<1$, все члены ряда (2.73) будут меньше единицы и погрешность суммы ряда будет мала независимо от первоначального значения угла. Прием, состоящий в приведении аргумента к так называемому основному интервалу, является стандартным при вычислении элементарных функций (см. п. 3.4).

Указанный способ уменьшения погрешностей использован в стандартной подпрограмме вычисления $\sin x$, входящей в состав одного из трансляторов. Результаты вычисления значений $\sin x$ с помощью этой подпрограммы приведены в последней колонке табл. 2.1. Как видно, подпрограмма в любом случае дает не меньше шести достоверных значащих цифр.

Пример 2.11. Вычисление значений функции e^x.

Рассмотрим теперь вычисление экспоненты с помощью ряда Тейлора:

$$e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \ldots . \qquad (2.74)$$

Известно, что ряд (2.74) сходится для всех конечных значений аргумента x: $-\infty < x < \infty$. Поэтому в соответствии с результатами классической математики его можно использовать для вычисления e^x при любых x.

Вычисления проведем с точностью в 7 десятичных разрядов. Если при суммировании величина текущего члена ряда по модулю такова, что она не изменяет цифру последнего разряда накопленной суммы, этот член будем отбрасывать. Результаты вычисления функции e^x для разных значений x приведены в табл. 2.2. В последней колонке таблицы приведены 7 верных значащих цифр экспоненты.

Таблица .2.2

Результаты вычисления функции e^x

x	e^x, непосредственное использование ряда	e^x, вычисление по (2.75)	e^x, точное значение
1	2,718282	2,718282	2,718282
5	148,4132	148,4132	148,4132
10	22026,47	22026,47	22026,46
15	3269017	3269017	3269017
–1	0,3678794	0,3678794	0,3678794
–5	$6{,}737784\cdot10^{-3}$	$6{,}737946\cdot10^{-3}$	$6{,}737947\cdot10^{-3}$
–10	$-1{,}640861\cdot10^{-4}$	$4{,}539992\cdot10^{-5}$	$4{,}539993\cdot10^{-5}$
–15	$-2{,}237700\cdot10^{-2}$	$3{,}059023\cdot10^{-7}$	$3{,}059023\cdot10^{-7}$

Как видно, при $x>0$ непосредственное применение ряда (2.74) дает вполне хорошую точность вычислений e^x. Однако при отрицательных значениях x точность резко снижается, и при $x \le -10$ вычисления дают даже качественно неверный результат (значения экспоненты не могут быть меньше нуля).

Исследуем причину появления такой ошибки. Для этого запишем значения некоторых членов ряда Тейлора при вычислении e^{-15}:

$e^{-15} \approx 1 – 15 + 112{,}5 – 562{,}5 + \ldots – 312540{,}3 + 334864{,}6 + \ldots – 0{,}0000062 + \ldots$

Более подробно величины членов ряда (2.74) при вычислениях приведены в табл. 2.3.

Таблица 2.3

Вычисленные члены ряда Тейлора при $x = –1$

n	n-й член ряда Тейлора
0	1,000000
1	–15,00000
2	112,5000
3	–562,5000
...	…
13	–312540,3
14	334864,6
15	–334864,6
16	313935,5
17	–277001,9
...	…
51	$-6{,}166081\cdot10^{-7}$
52	$1{,}778677\cdot10^{-7}$
53	$5{,}033992\cdot10^{-8}$

Рассмотрение значений членов ряда в табл. 2.3 позволяет понять механизм образования погрешности. Некоторые из этих членов (а следовательно, и промежуточные суммы) много больше, чем окончательный результат $e^{-15} \approx 3{,}06 \cdot 10^{-7}$. Когда один из таких больших членов вычитается из накопленной к этому моменту суммы, разность становится малой и имеет большую относительную погрешность. Таким образом, рассматриваемый пример хорошо иллюстрирует то обстоятельство, что при вычитании близких чисел относительная погрешность результата весьма велика (см. п.2.4.1).

Указанный механизм отсутствует при $x>0$, так как в этом случае все члены ряда (2.74) положительны (происходит только суммирование чисел). В результате погрешности вычисления e^x при положительных x весьма малы.

Сказанное позволяет предложить очень простой способ снижения погрешностей для $x<0$ – следует воспользоваться соотношением $e^{-x} = \dfrac{1}{e^x}$, т.е. вначале вычислить e^x при положительном x и затем взять обратную величину. В итоге вычисление e^x при $x<0$ заменяется вычислением e^{-x} при $x>0$ по следующей формуле:

$$e^{-x} = \frac{1}{1 + x + \dfrac{x^2}{2!} + \dfrac{x^3}{3!} + \ldots}. \qquad (2.75)$$

Результаты вычислений e^x с использованием (2.75) при $x<0$ приведены в табл. 2.2 и свидетельствуют об эффективности такого способа.

Пример 2.12. Решение квадратного уравнения.

Даже при решении простых задач с небольшим числом действий погрешности округления могут значительно исказить результат. В качестве иллюстрации рассмотрим решение квадратного уравнения

$$ax^2 + bx + c = 0. \qquad (2.76)$$

Как известно, корни уравнения определяются формулами:

$$x_1 = \frac{-b + \sqrt{D}}{2a}; \qquad x_2 = \frac{-b - \sqrt{D}}{2a}, \qquad (2.77)$$

где дискриминант равен $D = b^2 - 4ac$.

При некоторых соотношениях между величинами коэффициентов a, b и c появляется возможность появления погрешностей в вычислении корней уравнения x_1 и x_2 по формулам (2.77). Пусть, например, $b>0$. Если коэффициент b значительно превышает по абсолютной величине один или оба ос-

тальных коэффициента таким образом, что $b^2 >> 4ac$, тогда $D \approx b^2$ и $\sqrt{D} \approx b$. В результате при вычислении числителя корня x_1 в (2.77) вычитаются два близких числа и погрешность нахождения x_1 может быть велика.

Приведем численный пример. Пусть $a = 1$; $b = 0{,}4002$; $c = 0{,}8 \cdot 10^{-4}$. Точное значение корня x_1 при этом равно $x_1^* = -0{,}2000 \cdot 10^{-3}$. Вычислим корень x_1 по формуле (2.77), используя для наглядности 4 десятичных разряда (округление выполняется по способу «усечения» чисел и обозначается знаком $\Rightarrow$, отбрасываемая часть числа указывается после знака |).

Имеем:

$$\sqrt{D} = \sqrt{0{,}1601|\,6004 - 0{,}00032} \Rightarrow \sqrt{0{,}1598} = 0{,}3997|\,4992 \Rightarrow 0{,}3977;$$

$$x_1 = \frac{-0{,}4002 + 0{,}3997}{2} = -0{,}2500 \cdot 10^{-3}.$$

Как видно, в полученном результате неверна уже вторая значащая цифра. Влияние погрешности округления чисел привело к относительной погрешности вычисления корня x_1:

$$|\delta(x_1)| = \left|\frac{x_1^* - x_1}{x_1^*}\right| = \frac{0{,}05}{0{,}2} = 0{,}25 = 25\%.$$

С целью уменьшения ошибки вычисления преобразуем формулу (2.77) для x_1 таким образом, чтобы разность двух близких чисел заменить их суммой:

$$x_1 = \frac{\sqrt{D} - b}{2a} \cdot \frac{\sqrt{D} + b}{\sqrt{D} + b} = \frac{D - b^2}{2a(\sqrt{D} + b)} = \frac{-4ac}{2a(\sqrt{D} + b)} = \frac{-2c}{\sqrt{D} + b}.$$

Для рассмотренного примера вычисления по последней формуле дают:

$$x_1 = \frac{-0{,}00016}{0{,}3997 + 0{,}4002} = -0{,}2000|\,2500 \cdot 10^{-3} \Rightarrow -0{,}2000 \cdot 10^{-3},$$

т.е. теперь верны все четыре значащих цифры вычисленного корня x_1. В случае $b<0$ аналогичный прием можно применить для вычисления корня x_2.

В приведенных примерах 2.11 и 2.12 уменьшить погрешности вычислений удалось благодаря преобразованию формул, содержащих разности близких чисел, в эквивалентные формулы, не содержащие таких разностей.

Общий вывод состоит в том, что для многих задач существуют индивидуальные приемы, позволяющие повысить точность вычислений. Однако поиск способов снижения погрешностей невозможен без тщательного анализа задачи и выявления источников ошибок. Необходимы также знания общих положений, касающихся образования ошибок при вычислениях на ЭВМ, и некоторая изобретательность.