Понятие математического алгоритма решения. Формы представления чисел. Анализ ошибок при практической реализации алгоритмов решения. Виды погрешностей. Источники ошибок .

Никакие компьютеры сами по себе не могут решить ни одной содержательной задачи. Они способны выполнять лишь небольшое число очень простых действий. Вся их интеллектуальная сила определяется программами, составленными человеком. Программы также реализуют последовательности простых действий. Но эти действия целенаправленные. Поэтому искусство решать задачи на компьютере есть искусство превращения процесса поиска решения в процесс выполнения последовательности простых действий. Называется такое искусство **разработкой алгоритмов**.

Понятие алгоритма – одно из основных понятий программирования и математики. **Алгоритм — это последовательность действий, направленных на получение определённого результата за конечное число шагов.**

Алгоритм записывается на формальном языке, исключающем неоднозначность толкования. Исполнитель – это человек, компьютер, автоматическое устройство и т.п. Он должен уметь выполнять все команды, составляющие алгоритм, причем механически, "не раздумывая".

Запись алгоритма на формальном языке называется программой. Иногда само понятие алгоритма отождествляется с его записью, так что слова "алгоритм" и "программа" – почти синонимы. Небольшое различие заключается в том, что при упоминании алгоритма, как правило, имеют в виду основную идею его построения, общую для всех алгоритмических языков. Программа же всегда связана с записью алгоритма на конкретном формальном языке.

Пример 1. Составить алгоритм нахождения значения функции $y = 1/x$.

1) Ввод переменной x.

2) Если $x = 0$, то шаг 5, иначе шаг 3.

3) Вычислить $y = 1/x$.

4) Вывод y.

5) Конец.

В математике рассматриваются различные виды алгоритмов – программы для машин Тьюринга, алгоритмы Маркова (нормальные алгоритмы), частично рекурсивные функции и т.п.

Различные определения алгоритма в явной или неявной форме содержат следующий **ряд общих требований**:

Дискретность (от лат. discretus — разделенный, прерывистый) – это разбиение алгоритма на ряд отдельных законченных действий (шагов).

Данное свойство указывает, что любой алгоритм должен состоять из конкретных действий, следующих в определенном порядке. Так как выполнение алгоритма может быть перепоручено компьютеру, то, следовательно, для реализации алгоритма на компьютере необходимо выполнение требования дискретности вычислительных операций, т.е. возможно разделить задачи на элементарные операции для выполнения их счета.

Детерминированность (от лат. determinate — определенность, точность) - любое действие алгоритма должно быть строго и недвусмысленно определено в каждом случае.

Например, если к остановке подходят автобусы разных маршрутов, то в алгоритме должен быть указан конкретный номер маршрута — 5. Кроме того, необходимо указать точное количество остановок, которое надо проехать, — скажем, три.

Конечность - каждое действие в отдельности и алгоритм в целом должны иметь возможность завершения.

Массовость - один и тот же алгоритм можно использовать с разными исходными данными.

Свойство массовости означает, что алгоритм должен быть применен к целому классу однотипных задач, для которых меняются исходные условия. Например, не имеет смысла строить алгоритм нахождения наибольшего общего делителя только для одной пары чисел 10 и 15.

Результативность - в алгоритме не было ошибок.

Результативность алгоритма означает, что данный процесс является пошаговым и должен быть завершен через конечное число шагов, т.е. в алгоритме не должно быть ошибок.

Понятие результативности алгоритма связано с областью его применения. В том случае, если условия задачи взяты из области применения, то алгоритм перерабатывает их в конечное решение, после чего наступает остановка вычислительного процесса; если условия взяты не из области применения, то либо никогда не наступит остановка, либо наступает, но нельзя узнать, какое из прлученных чисел является результатом.

Пример 2. Рассмотрим алгоритм нахождения большего из двух заданных чисел A и B:

1. Из числа A вычесть число B.
2. Если получилось отрицательное значение, то сообщить, что число B больше.
3. Если получилось положительное значение, то сообщить, что число A больше.

При всей простоте и очевидности алгоритма, не каждый сразу поймет его ошибочность. Ведь если оба числа равны, то не получится никакого сообщения. Значит, надо обязательно предусмотреть это вариант, например:

1. Из числа A вычесть число B.
2. Если получилось отрицательное значение, то сообщить, что число B больше.
3. Если получилось положительное значение, то сообщить, что число A больше.
4. Если получился ноль, то сообщить, что числа равны.

Поясним эти свойства на простом примере.

Пример 3. Рассмотрим следующую формулу вычисления числа π:

$$\pi = 4\left(1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \ldots\right) = 4\sum_{i=0}^{\infty}\frac{(-1)^i}{2i+1}$$

Является ли эта формула алгоритмом вычисления числа π? Ответ на этот вопрос — «нет», так как здесь нет ни свойства массовости (нет входных данных), ни свойства конечности (сумма бесконечного количества чисел)

Важную роль играют рекурсивные алгоритмы (алгоритмы, вызывающие сами себя до тех пор, пока не будет достигнуто некоторое условие возвращения).

Для каждого алгоритма есть некоторое множество объектов, допустимых в качестве исходных данных.

Пример 4. В алгоритме деления вещественных чисел делимое может быть любым, а делитель не может быть равен нулю**.**

Алгоритм служит, как правило, для решения не одной конкретной задачи, а некоторого класса задач.

Пример 5. Алгоритм сложения применим к любой паре натуральных чисел. В этом выражается его свойство массовости, то есть одного класса.

Если математическая модель построена, то следующее, что надо сделать, – это разработать алгоритм решения задачи, описывающей модель.

Алгоритмы строятся для решения тех или иных вычислительных задач. Формулировка задачи описывает, каким требованиям должно удовлетворять решение задачи, а алгоритм, решающий эту задачу, находит объект, этим требованиям удовлетворяющий.

Принято различать два вида алгоритмов:

Комбинаторным (нечисленным) алгоритмом называется алгоритм, объектом обработки которого являются комбинаторные объекты (сочетания, размещения, перестановки), построенные из элементов конечного множества. К данному виду алгоритмов относится алгоритмы генерации комбинаторных объектов, графовые алгоритмы, потоковые алгоритмы, алгоритмы кодирования и т.д. .

Численным алгоритмом называется алгоритм, объектом обработки которого служат числа. В данный класс относят различные вычислительные алгоритмы, алгоритмы численного решения дифференциальных уравнений и т.д. .

Анализ алгоритмов и их порядок роста

Для алгоритмов существует два основных критерия:

Правильность

Сложность (трудоемкость).

Алгоритм считается правильным если он дает верный результат для решаемой задачи.

Что касается трудоемкости алгоритмов, то она составляется из четырех позиций:

- **Логическая сложность** оценивается числом человеко-месяцев, затраченных на выполнение алгоритма. Основной недостаток - человеческий фактор.

- **Статическая сложность** это длина описания алгоритма, измеряемая числом операторов языка программирования.
- **Временная сложность** это математическое время выполнения алгоритма, измеряемое в условных единицах времени.
- **Емкостная сложность** это число условных еденицы памяти, необходимых для выполнения алгоритма.

Важно отметить, что временная и емкостная сложности не привязываются к конкретным языкам программироания и к конкретным компьютерам, и поэтому является основными критериями в оценке алгоритмов и заключаются в построении функиций сложности и ассимптотической оценке этих функций.

Функции временной и емкостной сложностей строятся по аналогии. Для современной машины время выполнения алгоритма наиболее важно нежели занимаемая память.

Раздел математики, изучающий общие свойства алгоритмов, называется теорией алгоритмов.

Представление чисел

Научное обозначение

Стандартный метод представления действительного числа, называемый научным обозначением, получается посредством сдвига десятичной точки и присвоения соответствующей степени 10, например:

$$0{,}0000575 = 5{,}75 \times 10^{-5},$$

$$230000000 = 2{,}3 \times 10^{8}.$$

В химии, к примеру, важной константой является число Авогадро, равное $6{,}02252 \times 10^{23}$. Это число атомов в грамм-атоме любого химически простого вещества. В компьютерной терминологии $1K = 1{,}024 \times 10^{3}$, а скорость света в вакууме $c = 2{,}99792458 \times 10^{8}$ м/с, заряд электрона $e = 1{,}6021892 \times 10^{-19}$ Кл.

Машинные числа

В компьютерах для действительных чисел используется нормированное двоичное представление с плавающей точкой. Это означает, что математическая вевеличина x на самом деле не хранится в компьютере. Вместо нее в компьютере хранится двоичное приближение x:

$$x \approx \pm q \times 2^{n} \qquad (1)$$

Число q является мантиссой, и это конечное двоичное представление, которое удовлетворяет неравенству $1/2 < q < 1$. Целое число n называется показателем степени.

В компьютере используется только небольшое подмножество системы представления действительных чисел. Типично, что это подмножество содержит только часть двоичного числа, предложенного в (1). **Количество двоичных знаков ограничено двумя числами q и n.**

Пример 6:

133,21 = 10^2*1.3321, 10^2- порядок, 1.3321- мантисса.

1332.1 = 10^3*1.3321

0.13321 = 10^{-1}*1.3321

Важно понять, что когда мантисса и показатель степени в (1) ограничены, то у компьютера имеется ограниченное количество выбора из памяти вариантов приближений действительного числа x.

Компьютерные числа с плавающей точкой

В компьютере имеются как целая форма представления чисел, так и форма с плавающей точкой. Целая форма используется для вычислений с целыми числами и имеет ограниченное применение в численном анализе. Должно быть понятно, что любой компьютер, выполняя (1), ограничивает количество цифр, используемых в мантиссе q, и возможный диапазон показателя степени n должен быть ограничен.

Компьютеры с 32 двоичными разрядами, представляя действительное число с обычной точностью, использует 8 двоичных разрядов для показателя степени и 24 двоичных разряда для мантиссы. Они могут представлять действительные числа в интервале

$$\text{от } 2{,}938736E-39 \text{ до } 1{,}701412E+38$$

(т. е. от 2^{-128} до 2^{127}) с шестью десятичными знаками точности.

Компьютеры с 48 двоичными разрядами, представляя действительное число с обычной точностью, могут использовать 8 двоичных разрядов для показателя степени и 40 двоичных разрядов для мантиссы. Они могут представлять действительные числа в интервале от

$$2{,}938735877E-39 \text{ до } 1{,}7014118346E+38$$

(т. е. от 2^{-128} до 2^{127}) с 11 десятичными знаками точности.

Анализ ошибок

В практике численного анализа важно сознавать, что численное решение — это не точное математическое решение. Точность численного решения уменьшается по многим причинам несколькими тонкими способами. Понимание этих трудностей часто может привести профессионала к правильному выполнению и/или усовершенствованию численного алгоритма.

Определение. Предположим, что $\bar{p}$ является приближением p. Абсолютная ошибка равна $E_p = |p - \bar{p}|$, относительная ошибка равна $R_p = \dfrac{|p - \bar{p}|}{|p|}$ при условии, что $p \neq 0$.

Ошибка — это простая разность между истинным и приближенным значениями, тогда как относительная ошибка — это доля истинного значения.

Пример 7. Найдем абсолютную и относительную ошибки в следующих трех случаях. Пусть $x = 3{,}141592$ и $x = 3{,}14$, тогда абсолютная ошибка равна

$$E_x = |x - \bar{x}| = |3{,}141592 - 3{,}14| = 0{,}001592$$

и относительная ошибка равна

$$R_x = \frac{|x - \bar{x}|}{|x|} = \frac{|3{,}141592 - 3{,}14|}{|3{,}141592|} = 0{,}00507.$$

Пусть $y = 1000000$ и $\bar{y} = 999996$, тогда ошибка равна

$$E_y = |y - \bar{y}| = |1000000 - 999996| = 4$$

и относительная ошибка равна

$$R_y = \frac{|y - \bar{y}|}{|y|} = \frac{|1000000 - 999996|}{|1000000|} = 0{,}000004.$$

Пусть $z = 0{,}000012$ и $\bar{z} = 0{,}000009$, тогда ошибка равна:

$$E_z = |z - \bar{z}| = |0{,}000012 - 0{,}000009| = 0{,}000003$$

и относительная ошибка равна

$$R_z = \frac{|z - \bar{z}|}{|z|} = \frac{|0{,}000012 - 0{,}000009|}{|0{,}000012|} = 0{,}25.$$

В случае x между абсолютной и относительной погрешностями нет большой разницы и любая из них может быть использована для оценки точности $\bar{x}$. Во втором случае значение y — величина размера 10^6, абсолютная ошибка велика, относительная ошибка мала и $\bar{y}$ можно рассматривать как хорошее приближение y. В третьем случае z — величина размера 10^{-6}, абсолютная ошибка является наименьшей из всех трех ошибок, но относительная ошибка — наибольшая. В процентном отношении она составляет 25%, и такое значение $\bar{z}$ z

является плохим приближением z. Заметим, что чем больше $|p|$ удаляется от 1 (в сторону увеличения или уменьшения), тем относительная ошибка становится лучшим индикатором точности приближения, чем абсолютная ошибка. Относительная ошибка предпочтительнее для представления с плавающей точкой, так как она связана непосредственно с мантиссой.

Источники погрешности решения задачи на ЭВМ

(Пример – методы поиска корней нелинейных уравнений по своей природе являются приближенными в отличие от прямых методов, дающих точное решение.)

Однако на самом деле при решении задачи на компьютере ответ все равно будет содержать погрешность.

В качестве основных источников погрешности обычно рассматривают три вида ошибок. Это так называемые ошибки усечения, ошибки округления и ошибки распространения. Рассмотрим их.

Ошибки усечения

Этот вид ошибок связан с погрешностью, заложенной в самой задаче. Он может быть обусловлен неточностью определения исходных данных. Например, если в условии задачи заданы какие-либо размеры, то на практике для реальных объектов эти размеры известны всегда с некоторой точностью. То же самое касается любых других физических параметров. Сюда же можно отнести неточность расчетных формул и входящих в них числовых коэффициентов.

Большое число расчетных формул являются эмпирическими и дают результат с некоторой погрешностью, содержат подгоночные коэффициенты, обеспечивающие приемлемую ошибку в ограниченном диапазоне входных параметров. Поэтому, как правило, если исходные данные известны с некоторой погрешностью, вряд ли стоит пытаться получить результат с меньшей погрешностью.

Понятие ошибки усечения используется, если ошибка возникает из-за того, что более сложное математическое выражение "заменяется" более простой

формулой. Эта терминология возникла из техники замены сложной функции укороченными рядами Тейлора

$$\sum_{k=0}^{\infty} \frac{f^{(k)}(a)}{k!}(x-a)^k .$$

Пример 8. Разложим в ряд Тейлора функцию:

$$e^{x^2} = 1 + x^2 + \frac{x^4}{2!} + \frac{x^6}{3!} + \frac{x^8}{4!} + \ldots + \frac{x^{2n}}{n!} + \ldots$$

Функцию можно представить только лишь пятью членами ряда, которые могут быть использованы при нахождении приближенных значений интеграла.

Пример 9. Вычислить значение определенного интегрела $\int_0^1 e^x dx$. Определить точность приближения, полученную при замене подинтегральной функции $f(x) = e^x$ усеченным рядом Тейлора (взять пять членов ряда).

Выполняя операции в MatLab, получим:

```
>> int(exp(x),0,.5)
ans =
-1+exp(1/2)
>> -1+exp(1/2)
ans =
  0.648721270700128
>> int(1+x+1/2*x^2+1/6*x^3+1/24*x^4,0,0.5)
ans =
  0.648697916666667
```

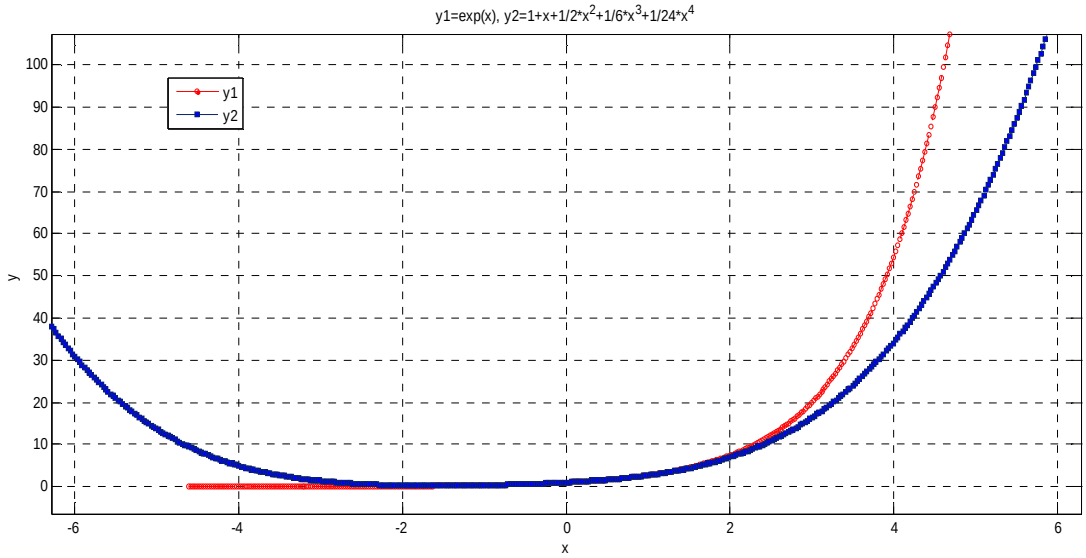
Находим абсолютную погрешность:

```
>> E=0.648721270700128-0.648697916666667
E =
   2.335403346098719e-005
```

Находим относительную погрешность:

```
>> R=E/0.648721270700128
R =
   3.600010438347815e-005
```

Графики функций:



Ошибки распространения

Данный вид ошибок связан с применением того или иного способа решения задачи. В ходе вычислений неизбежно происходит накопление или, иначе говоря, распространение ошибки. Помимо того, что сами исходные данные не являются точными, новая погрешность возникает при их перемножении, сложении и т. п. Накопление ошибки зависит от характера и количества арифметических действий, используемых в расчете.

Обычно для решения одной и той же задачи может быть использован ряд различных методов решения. Например, систему линейных алгебраических уравнений можно решить методом Гаусса или через определители (методом Крамера). Теоретически оба метода позволяют получить точное решение. Однако на практике при решении больших систем уравнений метод Гаусса обеспечивает меньшую погрешность, чем метод Крамера, так как ис-
пользует меньший объем вычислений.

Ошибки округления

Это тип ошибок связан с тем, что истинное значение числа не всегда точно сохраняется компьютером. При сохранении вещественного числа в памяти компьютера оно записывается в виде мантиссы и порядка, примерно так же, как отображается число на дисплее калькулятора (см. рис. 12).

$\pm$	R_1	R_2	R_3	.	.	.	.	R_n	$\pm$	D_1	D_2	.	.	.	D_m
Мантисса									Порядок						

Рис. 1. Структура записи вещественного числа

Здесь R_1, R_2, R_n – разряды мантиссы, D_1, D_2, …, D_m – разряды порядка. На самом деле конечно, в отличие от дисплея калькулятора, мантисса и порядок числа, включая их знаки, в памяти компьютера хранятся в двоичном виде. Но для обсуждения природы ошибок округления это различие не столь принципиально.

Понятно, что иррациональные числа такие, как $\pi = 3.14159...$ и $e = 2{,}712...$ не могут быть представлены в памяти компьютера в принципе. Однако же и рациональные числа, если количество их значащих цифр превышает число отведенных разрядов мантиссы (см. рис. 1), будут представлены не точно. При этом цифра последнего сохраняемого в ЭВМ разряда может быть записана с округлением или без него.

Фактически при заданной структуре хранения числа компьютер может использовать не бесконечное, а конечное число рациональных чисел, которые вписываются в приведенную на рис. 1 схему. Поэтому любой входной параметр решаемой задачи, ее промежуточный результат и окончательной ответ всегда округляются до разрешенных в компьютере чисел.

Следующий важный вывод касается диапазона представления чисел в ЭВМ. Если проводить рассуждения для десятичной системы счисления, то максимальное по модулю число, которое может быть представлено в соответствии со схемой на рис. 1, равно

$$\pm X_\infty = \pm 999...9 \times 10^{+99...9}.$$

Все числа, превышающие по модулю X_∞, не представимы в ЭВМ и рассматриваются как машинная бесконечность (**пример в MatLab − inf**). Если в ходе расчетов будет получен результат, превышающий X_∞, то произойдет аварийное завершение вычислений по переполнению.

Минимальное по модулю число, сохраняемое в памяти компьютера, по схеме на рис. 1 равно

$$\pm X_0 = \pm 000...1 \times 10^{-99...9}$$

Числа, модуль которых меньше X_0, воспринимаются ЭВМ как нуль, точнее как машинный нуль. Если при выполнении расчетов будет получен результат меньше, чем X_0, то это будет воспринято как потеря порядка. Обычно в подобной ситуации результат полагается равным нулю, и вычисления продолжаются.

Пример 10. Минимальное по модулю число в Matlab, сохраняемое в памяти компьютера, равно:

```
>> realmin
ans =
2.225073858507201e-308
```

Максимальное по модулю число равно:

```
>> realmax
ans =
   1.797693134862316e+308
```

Потеря значащих цифр

Рассмотрим два приблизительно равных числа $p = 3{,}1415926536$ $q = 3{,}1415957341$, имеющих точность, равную 11 десятичным цифрам. Предположим, что разность между числами равна $p - q = -0{,}0000030805$. Поскольку первые шесть цифр p и q одинаковы, разность между $p - q$ состоит только из пяти точных десятичных цифр. Этот феномен называется **потерей значащих цифр** или потерями из-за вычитания. Эта потеря точности окончательного ответа может подкрасться, когда о ней не подозреваешь.

Пример 11. Сравним результаты вычисления $f(500)$ и $g(500)$, используя шесть цифр и округление. Функциями являются $f(x) = x(\sqrt{x+1} - \sqrt{x})$ и $g(x) = \dfrac{x}{\sqrt{x+1} + \sqrt{x}}$ соответственно. Для первой функции

$$f(500) = 500(\sqrt{500+1} - \sqrt{500}) = 500(22{,}3830 - 22{,}3607) = 11{,}1500.$$

Для второй функции

$$g(500) = \frac{500}{\sqrt{500+1} + \sqrt{500}} = \frac{500}{22{,}3830 + 22{,}3607} = 11{,}1748.$$

Вторая функция $g(x)$, является алгебраическим єквивалентом $f(x)$:

$$f(x)=\frac{x(\sqrt{x+1}-\sqrt{x})(\sqrt{x+1}+\sqrt{x})}{\sqrt{x+1}+\sqrt{x}}=\frac{x\left((\sqrt{x+1})^2-(\sqrt{x})^2\right)}{\sqrt{x+1}+\sqrt{x}}=\frac{x}{\sqrt{x+1}+\sqrt{x}}.$$

Ответ, $g(500)=11{,}1748$, имеет меньшую ошибку и является таким же, как полученный округлением истинного ответа 11,174755300747198... с шестью знаками.