

Погрешности вычислений

Виды погрешностей

- Погрешность модели, задачи
- Погрешность численного метода
- Погрешность вычислений на компьютере

Ошибки

- Точное, истинное значение $p \neq 0$
- Оценка, приближенное значение $\hat{p} \approx p$
 - «крышка»
- Абсолютная ошибка $\Delta p = E_p = |p - \hat{p}|$
 - *Absolute Error*
- Относительная ошибка $\delta p = R_p = \left| \frac{p - \hat{p}}{p} \right|$
 - *Relative Error*

Погрешность операций

- Абсолютная погрешность суммы и разности $\Delta_{x+y} \approx \Delta_{x-y} \approx \Delta_x + \Delta_y$
- Относительная погрешность произведения $\delta_{x \cdot y} \approx \delta_x + \delta_y$

Погрешность среднего

- Вычисление среднего арифметического $\bar{x} = \frac{x_1 + x_2 + \dots + x_n}{n}$
- Погрешность исходных данных $\Delta(x_i)$
- Погрешность результата вычислений
 - Классическая оценка $\Delta(\bar{x}) \approx \Delta(x_i)$
 - Статистическая оценка $\Delta(\bar{x}) \approx \frac{\Delta(x_i)}{\sqrt{n}}$

Вычисления

- Результаты вычислений должны иметь заданную точность
- Расчеты вручную и на калькуляторе
 - Десятичные числа
- Компьютерные вычисления
 - Двоичные числа

Двоичные числа

- Разряды = 0 или 1
- Старший разряд в 2 раза больше

$$101_2 = 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 = 1 \cdot 4 + 1 \cdot 1 = 5_{10}$$

- Примеры

$$11_2 = ?_{10}$$

$$10_2 = ?_{10}$$

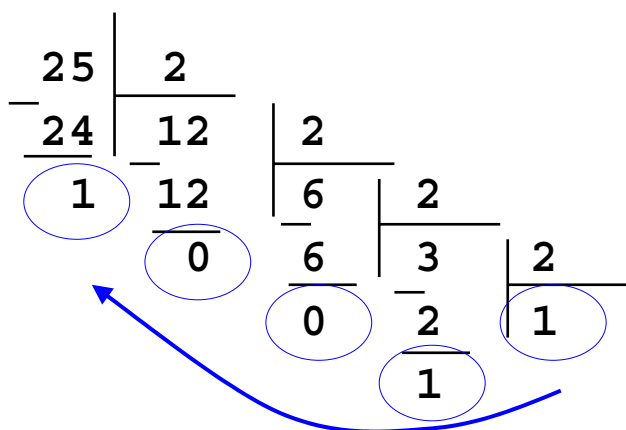
$$110_2 = ?_{10}$$

Перевод в двоичное представление

- Остатки от деления на 2
 - Делим десятичное число на 2 столбиком
 - Повторяем, пока остаток не будет меньше двух
 - Расставляем остатки в обратном порядке

Пример

$$25_{10} = 11001_2$$



Десятичные дроби

- Точка (запятая) - десятичный разделитель
 - Формат ввода-вывода данных
 - Представление в компьютере
- Фиксированная точка
 - 123.17
 - 0.00123
 - 174.9450
- Плавающая точка
 - 1.23*10⁺⁵
 - 1.23*10⁻³
 - 1.23E5
 - 1.23E-3

Задача

- Проведите вычисления над десятичными числами с переводом из представления с плавающей точкой в представление с фиксированной точкой и обратно в представление с плавающей точкой с 4 значащими разрядами и определите погрешность:
- $1,223E-1 + 3,201E-2$

Двоичные дроби

- Машинные числа
- Нормированное двоичное представление с плавающей точкой
- Вместо самого числа в компьютере хранится его двоичное представление

$$x \approx \pm q \times 2^n$$

знак числа

q – мантисса

n – порядок, показатель степени

$$\frac{1}{2} \leq q < 1$$

Мантисса

- (1) Дробная часть логарифма
- (2) Часть числа с плавающей точкой, содержащая значащие цифры
 - лат. *mantissa* – прибавка
- **significantand, coefficient, mantissa**
 - part of a floating-point number, consisting of its significant digits
 - Depending on the interpretation of the exponent, the significantand may represent an integer or a fraction.

Fixed point vs. floating point

- Фиксированная точка (запятая)
 - Заданное число знаков до и после десятичного разделителя (запятой)
- узкий диапазон чисел
 - усечение или округление результата вычислений (rounded or truncated)
 - переполнение (overflow)
 - потеря точности (precision loss)
- Пример: 4-байта, точность в 3 значащих цифры
- Сравнение: Представление с плавающей точкой
 - Floating point number representation

	Fixed-point	Floating-point
диапазон	6 порядков	70 порядков
10^x	10^6	10^7

Разрядная сетка

- Число разрядов ограничено
- Пример:
 - 4 разряда для мантиссы
 - Показатель от -3 до +4

$$x \approx \pm(0, dddd)_2 \times 2^n$$

- Пример
 $(+0,11)_2 * 2^{-1} = (\frac{1}{2} + \frac{1}{4}) * \frac{1}{2} =$
 $= (0,5 + 0,25) / 2 = 0,625 / 2 = 0,375_{10}$

Задача

- 4 двоичных разряда
- Десятичная точность:
 - Сколько десятичных разрядов?

Машинные числа

мантисса	порядок		
	$n = -2$	$n = -1$	$n = 0$
$0,1000_2$	—	—	?
$0,1001_2$	—	—	?
$0,1010_2$	—	?	—
$0,1011_2$	—	?	—
$0,1100_2$	?	—	—
$0,1101_2$	?	—	—

Машинные числа

мантисса	порядок		
	$n = -2$	$n = -1$	$n = 0$
$0,1000_2$	—	—	0,5
$0,1001_2$	—	—	0,5625
$0,1010_2$	—	0,3125	—
$0,1011_2$	—	0,34375	—
$0,1100_2$	0,1875	—	—
$0,1101_2$	0,203125	—	—

Разрядная сетка

- Пример
- Точное и компьютерное вычисление
 $x = 1/3 + 1/5$

Точное решение

$$\frac{1}{3} + \frac{1}{5} =$$

Точное решение

$$\frac{1}{3} + \frac{1}{5} = \frac{5+3}{3 \cdot 5} = \frac{8}{15} = 0,533...$$

$$\frac{1}{3} + \frac{1}{5} = 0,333... + 0,2 = 0,533...$$

Машинное представление

$$1/3 = 0,33333_{10} = ?_2$$

- Сравниваем ошибки двоичного представления

$$0,1010_2 * 2^{-1}: 0,33333 - 0,3125 =$$

$$0,1011_2 * 2^{-1}: 0,33333 - 0,34375 =$$

- Выбираем двоичное представление с меньшей ошибкой

$$1/3 = ?_2$$

Машинное представление

$$1/3 = 0,33333_{10} = ?_2$$

- Сравниваем ошибки двоичного представления

$$0,1010_2 * 2^{-1}: 0,33333 - 0,3125 = 0,02083$$

$$0,1011_2 * 2^{-1}: 0,33333 - 0,34375 = 0,01042$$

- Выбираем двоичное представление с меньшей ошибкой

$$1/3 = 0,1011_2 * 2^{-1}$$

Машинное представление

$$1/5 = 0,2_{10} = ?_2$$

- Сравниваем ошибки двоичного представления

$$?: 0,2 - ? = ?$$

$$?: 0,2 - ? = ?$$

- Выбираем двоичное представление с меньшей ошибкой

$$1/5 = ?_2$$

Машинное представление

$$1/5 = 0,2_{10} = ?_2$$

- Сравниваем ошибки двоичного представления

$$0,1100_2 * 2^{-2}: 0,2 - 0,1875 = 0,0125$$

$$0,1101_2 * 2^{-2}: 0,2 - 0,203125 = 0,003125$$

- Выбираем двоичное представление с меньшей ошибкой

$$1/5 = 0,1101_2 * 2^{-2}$$

Машинное сложение

$$x = 1/3 + 1/5 =$$

$$= 0,1011_2 * 2^{-1} + 0,1101_2 * 2^{-2} =$$

$$= \begin{array}{r} + \quad ?_2 \\ \quad ?_2 \\ \hline \quad ?_2 \end{array}$$

$$= ?_2 * 2^? = ?_{10}$$

- Ошибка вычислений
?%

Машинное сложение

$$\begin{aligned}x &= 1/3 + 1/5 = \\&= 0,1011_2 * 2^{-1} + 0,1101_2 * 2^{-2} = \\&= \begin{array}{r} + \quad 0,010110_2 \\ \quad 0,001101_2 \\ \hline \quad 0,100011_2 \end{array} \\&= 0,1000_2 * 2^0 = 0,5_{10} \\&\bullet \text{ Ошибка вычислений} \\&\quad 0,53333 - 0,5 = 0,03333 = 6,3\%\end{aligned}$$

Вывод

- Машинное двоичное представление чисел может вносить дополнительные погрешности в результаты расчетов
- Нужно правильно задавать разрядность машинного представления

Самостоятельно

- Международный стандарт для арифметики с плавающей точкой
- IEEE Standard for Floating-Point Arithmetic
- IEEE 754

Сумма

$$\sum_{i=1}^{100000} 0,1 = ?$$

Точный ответ

$$\sum_{i=1}^{100000} 0,1 = 0,1 * 100000 = 10000$$

Вычисления в Scilab

```
-->x=0;
-->for i=1:100000
-->x=x+0.1;
-->end
-->printf('x = %10.30f\n', x)
x = 10000.0000000188480000000000000000
-->printf('dx = %10.30f\n', x-10000)
dx = 0.000000018848368199542165000000
```

Вычисления в Visual Studio C++

```
#include <stdio.h>
#include <conio.h>
void main() {
    float x=0; int i;
    for( i=1 ; i<=100000 ; i++ ) {x=x+0.1;}
    printf("x = %10.30f\n", x);
    printf("Delta  = %10.30f\n", x - 10000);
    _getch(); }
```

```
x      = 9998.55664062500000000000000000000000
Delta  = -1.44335937500000000000000000000000
```

Вычисления в Visual Studio C++

```
#include <stdio.h>
#include <conio.h>
void main() {
    double x=0; int i;
    for( i=1 ; i<=100000 ; i++ ) {x=x+0.1;}
    printf("x = %10.30f\n", x);
    printf("Delta  = %10.30f\n", x - 10000);
    _getch(); }
```

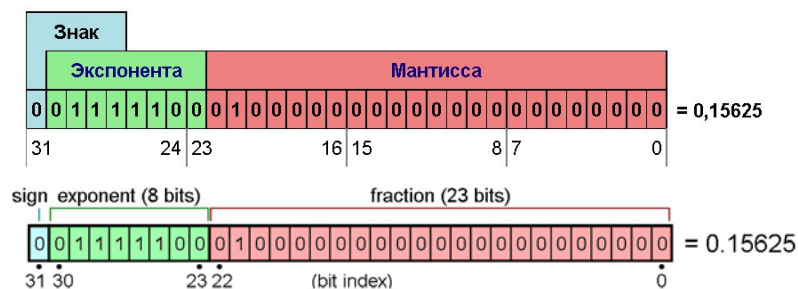
```
x = 10000.00000001884800000000000000000000
Delta  = 0.000000018848368199542165000000
```

Типы данных

Byte	Bit	Precision	Exp. Bin.	Exp. Dec.	Frac.	Dec. digits
4	32	Single	8 −126 +127	$10^{\pm 38}$	23+1	7
8	64	Double	11 −1022 +1023	$10^{\pm 308}$	52+1	16
16	128	Quadruple	15 −16382 +16383	$10^{\pm 4932}$	112+1	34

Binary 32

- Число одинарной точности
– Single precision, Single, Float
- Decimal digits = Binary digits $\times \lg(2)$



Число десятичных разрядов

- Single precision
- Decimal digits = Binary digits $\times \lg(2)$
- $23 \times \lg(2) = 23 \times 0,3 = 6,9$

Значащие цифры

- Разряды, передающие значение, определяющие точность представления числа
 - Нули до и после значащих цифр для указания порядка числа
 - *significant figures (significant digits)*

123000000

123

1,23

0,000123

1,23 $\cdot 10^{-34}$

1,23 $\cdot 10^{+67}$

0,123 $\cdot 10^{-3}$

Округление

- до n знаков после запятой
- до n значащих цифр
- Точность результата вычислений определяется точностью исходных данных

$$\frac{12,3}{0,51898734177215189873417721518987} =$$

Потери значащих цифр

- Потеря точности окончательного результата вычислений
- Потери при вычитании больше, чем при сложении

$$123,234567732 - 123,234567354 =$$

$$123,234567732 + 123,234567354 =$$

Решение квадратного уравнения

Расчеты с учетом погрешностей

Решение квадратного уравнения

$$ax^2 + bx + c = 0$$

$$D = b^2 - 4ac$$

$$x_{1,2} = \frac{-b \pm \sqrt{D}}{2a}$$

Дискриминант

- Дискриминант многочлена
лат. *discriminans* — разделяющий, различающий
dis- (раз-) + *cernere* (делить) = «разделять»
гр. *krinein* - решать
- $P(x) = a_0x^n + a_1x^{n-1} + \dots + a_n$,
- Выражение, в котором произведение распространено на всевозможные разности корней $a_1, a_2, \dots, a_n$ уравнения $P(x) = 0$.
- Дискриминант обращается в нуль, если имеются равные корни многочлена

Потеря точности при вычитании

$$x_{1,2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

$$b^2 - 4ac > 0$$

$$b > 0$$

$$|b| \approx \sqrt{b^2 - 4ac}$$

$$x^2 + 1000,001x + 1 = 0$$

$$D =$$

$$\sqrt{D} =$$

$$-b + \sqrt{D} =$$

$$-b - \sqrt{D} =$$

- Округление: Single 7, Double 16 значащих разрядов

Преобразование x_1

$$x_1 = \frac{-b + \sqrt{D}}{2a}$$

- Умножим числитель и знаменатель на

$$(b + \sqrt{D})$$

$$x_1 =$$

Разность квадратов

$$a^2 - b^2 = (a + b)(a - b)$$

Преобразование формулы

$$x_1 = \frac{-b + \sqrt{D}}{2a}$$

- Умножим числитель и знаменатель на

$$(b + \sqrt{D})$$

$$x_1 = \frac{(-b + \sqrt{D})(b + \sqrt{D})}{2a(b + \sqrt{D})} =$$

Преобразование формулы

$$\begin{aligned} x_1 &= \frac{(-b + \sqrt{D})(b + \sqrt{D})}{2a(b + \sqrt{D})} = \\ &= \frac{b^2 - 4ac - b^2}{2a(b + \sqrt{D})} = \frac{-4ac}{2a(b + \sqrt{D})} = \\ &= \frac{-2c}{b + \sqrt{D}} \end{aligned}$$

Преобразование x_2

$$x_2 = \frac{-b - \sqrt{D}}{2a}$$

- Умножим числитель и знаменатель на

$$(b - \sqrt{D})$$

$$x_2 =$$

Вычисление корней

$$x_1 = \frac{-b + \sqrt{b^2 - 4ac}}{2a} = \frac{-2c}{b + \sqrt{b^2 - 4ac}}$$

$$x_2 = \frac{-b - \sqrt{b^2 - 4ac}}{2a} = \frac{-2c}{b - \sqrt{b^2 - 4ac}}$$

- Какие формулы нужно использовать при положительном и при отрицательном b ?

Второй метод

$$x_{1,2} = \frac{-2c}{b \pm \sqrt{b^2 - 4ac}}$$

$$b^2 - 4ac > 0$$

$$b > 0$$

$$|b| \approx \sqrt{b^2 - 4ac}$$

$$x^2 + 1000,001x + 1 = 0$$

$$D =$$

$$\sqrt{D} =$$

$$-b + \sqrt{D} =$$

$$-b - \sqrt{D} =$$

- Округление: Single 7, Double 16
значащих разрядов

Пример SciLab

```
a=1;
b=-1000.001;
c=1;
m1x1=(-b+sqrt(b^2-4*a*c))/2/a;
m1x2=(-b-sqrt(b^2-4*a*c))/2/a;
m2x1=-2*a/(b+sqrt(b^2-4*a*c));
m2x2=-2*a/(b-sqrt(b^2-4*a*c));
printf('%5.20f\n%5.20f\n%5.20f\n%5.20f\n',m1x1,m2x1,m1x2,m2x2)
```

Результаты SciLab

```
1000.000000000000000000000000
1000.0000000236469000000000
    0.000999999999997635314
    0.001000000000000000000000
```

Float – Одинарная точность

```
#include <stdio.h>
#include <conio.h>
#include <math.h>
void main() {
    float a=1.0, b=-1000.001, c=1.0, m1x1, m1x2,
          m2x1, m2x2, D;
    D = b * b - 4 * a * c;
    m1x1=(-b+sqrt(D))/2/a;
    m1x2=(-b-sqrt(D))/2/a;
    m2x1=-2*a/(b+sqrt(D));
    m2x2=-2*a/(b-sqrt(D));
    printf("%5.20f\n%5.20f\n%5.20f\n%5.20f\n",m1x1
          ,m2x1,m1x2,m2x2);
    _getch(); }
```

Результаты (Float)

1000.0000000000000000000000

996.10870361328125000000

0.00100390648003667590

0.00100000004749745130

Double – Двойная точность

```
#include <stdio.h>
#include <conio.h>
#include <math.h>
void main() {
double a=1.0, b=-1000.001, c=1.0, m1x1,
    m1x2, m2x1, m2x2, D;
D = b * b - 4 * a * c;
m1x1=(-b+sqrt(D))/2/a;
m1x2=(-b-sqrt(D))/2/a;
m2x1=-2*a/(b+sqrt(D));
m2x2=-2*a/(b-sqrt(D));
printf("%5.20f\n%5.20f\n%5.20f\n%5.20f\n",m1
    x1,m2x1,m1x2,m2x2);
_getch(); }
```

Результаты (Double)

1000.0000000000000000000000

1000.00000002364690000000

0.000999999999997635314

0.0010000000000000000000

Сравнение результатов

C Float

1000.0000000000000000000000

996.10870361328125000000

0.00100390648003667590

0.00100000004749745130

C Double, SciLab

1000.0000000000000000000000

1000.00000002364690000000

0.000999999999997635314

0.0010000000000000000000

Выводы

- При составлении алгоритмов и программ нужно учитывать погрешность вычислений
- Погрешность при вычитании больше, чем погрешность при сложении
- Расчетные соотношения желательно преобразовать к формулам, использующим сложение

Свойства операций

Программирование операций

- При вычислениях могут нарушаться очевидные «точные» равенства (свойства операций)
 - Разный порядок суммирования

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

Пример

- Сложить три числа типа `double`
 - плавающая точка с двойной точностью
 - 16 значащих цифр
 - 16 верных десятичных разрядов

`a=1.2345678901234567890123456789*1016`

`b=1.2345678901234567890123456789*107`

`c=1.2345678901234567890123456789*107`

Программа SciLab

```
a=1.2345678901234567890123456789e16;  
b=1.2345678901234567890123456789e7;  
c=1.2345678901234567890123456789e7;  
d1=(a+b)+c;  
d2=(b+c)+a;  
printf('a=%3.20e\n',a)  
printf('b=%3.20e\n',b)  
printf('c=%3.20e\n',c)  
printf('%3.20e\n%3.20e\n',d1,d2)
```

Результаты SciLab

```
a=1.23456789012345680000e+016  
b=1.23456789012345670000e+007  
c=1.23456789012345670000e+007  
  
1.23456789259259240000e+016  
1.23456789259259260000e+016
```

Задача

- Сложить числа столбиком
- Оставлять только 16 разрядов

$$a = 1.2345678901234567890123456789 * 10^{16}$$

$$b = 1.2345678901234567890123456789 * 10^7$$

$$c = 1.2345678901234567890123456789 * 10^7$$

$$d1 = (a+b) + c$$

$$d2 = (b+c) + a$$

Коммутативность

- Переместительность, возможность изменения очередности выполнения операций

– лат. *commutativus* — меняющийся
com (вместе, совместно)
+ mutare (менять)
= «менять местами», «обмениваться»

- Сумма

$$a + b = b + a$$

- Произведение

$$a b = b a$$

Дистрибутивность

- Согласованность операций
 - лат. *distributivus* — распределительный
dis- (отделять, в разные стороны)
+ *tribuere* (давать, присвоить, назначить)
= «раздавать»
- дистрибутивность слева
$$x (y + z) = x y + x z$$
- дистрибутивность справа
$$(y + z) x = y x + z x$$

Ассоциативность

- Отдельные слагаемые можно объединять в группы (ассоциировать); сочетательность
 - лат. *associatio* — соединение
as- (*ad-*) (движение к, в направлении к)
+ *socius* (совместное участие, союз, сообщество)
= объединение, соединение
- Сложение
$$a + (b + c) = (a + b) + c$$
- Умножение
$$a(bc) = (ab)c$$

Пример

- Сумма большого числа одинаковых слагаемых типа `double`
- Для удобства отслеживания верных значащих цифр используем ряд `12...0`

`a=1.2345678901234567890123456789*1016`

- При сложении числа с накопленной суммой ошибка возрастает

Программа SciLab

```
a=1.2345678901234567890123456789e16;  
s=0;  
for i=1:1000000  
    s=s+a;  
end  
printf('  a=%3.20e\n',a)  
printf('  s=%3.20e\n',s)  
printf('n*a=%3.20e\n',a*1000000)  
printf('  d=%3.20e\n',s-a*1000000)
```

Результаты SciLab

```
a=1.23456789012345680000e+016
s=1.23456789011435770000e+022
n*a=1.23456789012345680000e+022
d=-9.09912309760000000000e+010
```

- Сколько верных цифр было в исходных данных и сколько осталось в результате?

Программа SciLab

```
a=1.2345678901234567890123456789e16;
s=0;
for i=1:1000000000
s=s+a;
end
printf('  a=%3.20e\n',a)
printf('  s=%3.20e\n',s)
printf('n*a=%3.20e\n',  a*1000000000)
printf('  d=%3.20e\n',s-a*1000000000)
```

Результаты SciLab

```
a=1.23456789012345680000e+016  
s=1.23456788479204580000e+025  
n*a=1.23456789012345680000e+025  
d=-5.33141097975644160000e+016
```

- Сколько верных цифр было в исходных данных и сколько осталось в результате?

Выводы

- Порядок выполнения операций может влиять на точность вычислений
 - При сложении большого количества чисел ошибка «накапливается» и «усиливается»
 - При большом числе слагаемых точность начинает ухудшаться
- Для сохранения точности вычислений можно изменять порядок выполнения операций в зависимости от соотношения чисел
 - Сначала складывать малые числа, потом большие
 - Выполнять сложение по группам, потом складывать группы