Motivation
Representing Numbers
Exotic Representation
Error
Practical Aspects

Mathematics Methods for Computer Science

Instructor: Xubo Yang

SJTU-SE DALAB

Reference book

Motivation
Representing Numbers
Exotic Representation
Error
Practical Aspects

Reference book: Solomon, Justin. Numerical Algorithms. Published by AK Peters/CRC Press, 2015.

Two Roles

Motivation
Representing Numbers
Exotic Representation
Error
Practical Aspects

From discrete mathematics to continuous mathematics.
From exact solutions to numerical approximations.
Focus on numerical analysis and processing of real-valued data.

Two Roles:

- Client of numerical methods
- Designer of numerical methods

Applications:

- computer graphics,
- computer vision,
- big data,
- machine learning,
- ...

Typical Linear Algebra

$$
\begin{aligned}
\|A\vec{x}-\vec{b}\|_2^2 &= (A\vec{x}-\vec{b})\cdot(A\vec{x}-\vec{b}) \\
&= (A\vec{x}-\vec{b})^\top(A\vec{x}-\vec{b}) \\
&= \left(\vec{x}^\top A^\top - \vec{b}^\top\right)(A\vec{x}-\vec{b}) \\
&= \vec{x}^\top A^\top A\vec{x} - \vec{x}^\top A^\top\vec{b} - \vec{b}^\top A\vec{x} + \vec{b}^\top\vec{b} \\
&= \|A\vec{x}\|_2^2 - 2\left(A^\top\vec{b}\right)\cdot\vec{x} + \|\vec{b}\|_2^2
\end{aligned}
$$

因为Ax与b均为列向量，所以其二范数(模)就可以转化为x^Tx的形式来计算。至于中间两项为何合并的问题，因为都是列向量，并且结果相同，故可以直接合并。

Example: Matrix Vector Multiplication

Motivation
Representing Numbers
Exotic Representation
Error
Practical Aspects

function MULTIPLY$(A, \vec{x})$
  ▷ Returns $\vec{b} = A\vec{x}$, where
  ▷ $A \in \mathbb{R}^{m\times n}$ and $\vec{x} \in \mathbb{R}^n$
  $\vec{b} \leftarrow \vec{0}$
  for $i \leftarrow 1, 2, \ldots, m$
    for $j \leftarrow 1, 2, \ldots, n$
      $b_i \leftarrow b_i + a_{ij}x_j$
  return $\vec{b}$

(a)

function MULTIPLY$(A, \vec{x})$
  ▷ Returns $\vec{b} = A\vec{x}$, where
  ▷ $A \in \mathbb{R}^{m\times n}$ and $\vec{x} \in \mathbb{R}^n$
  $\vec{b} \leftarrow \vec{0}$
  for $j \leftarrow 1, 2, \ldots, n$
    for $i \leftarrow 1, 2, \ldots, m$
      $b_i \leftarrow b_i + a_{ij}x_j$
  return $\vec{b}$

(b)

cache的问题会影响效率，但结果一样

Example: Matrix Vector Multiplication

Motivation
Representing Numbers
Exotic Representation
Error
Practical Aspects

$$A = \begin{pmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{pmatrix}$$

(a)

1	2	3	4	5	6

(b) Row-major

1	3	5	2	4	6

(c) Column-major

Topics I

1. Numeric
   - Stability and error analysis
   - Floating-point representation
2. Linear algebra
   - Guassian elemination and LU
   - Column space and QR
   - Eigenproblems
   - Applications
3. Root-finding and optimization
   - Single variable
   - Multivariable
   - Constrained optimization
   - Iterative linear solvers; Conjugate gradients

Topics II

Motivation
Representing Numbers
Exotic Representation
Error
Practical Aspects

4. Interpolation and quadrature
   - Interpolation
   - Approximating integrals (optional)
   - Approximating derivatives (optional)
5. Differential equations (optional)
   - ODEs: time-stepping, discretization
   - PDEs: Poisson equation, heat equation, waves
   - Techniques: Differencing, finite elements (time-permitting)

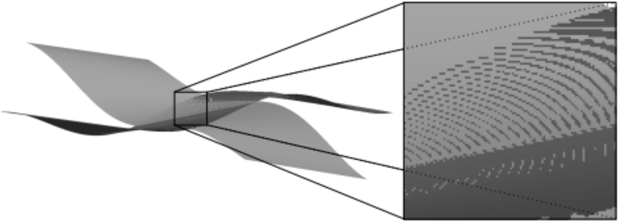
Motivation
Representing Numbers
Exotic Representation
Error
Practical Aspects

Lecture

Numerics And Error Analysis

Motivation
Representing Numbers
Exotic Representation
Error
Practical Aspects

Example: Z-fighting



老师在这里拓展了z-buffer的相关知识：
分析一个计算机图形时，会按照与人眼视线平行的方向设置为z轴，并按照z轴上数值的大小来区分不同的点(具体怎样区分，应该就是用像素的不同表示吧)，而z-fighting说的是，在两个点的z值离得很近，并且其差距小于精确度要求时，就会出现两个不同的点竞争同一个像素位置的情况，本质原因浮点数计算的精确度问题。

Motivation
Representing Numbers
Exotic Representation
Error
Practical Aspects

Prototypical Example

```
double x = 1.0;
double y = x / 3.0;
if (x == y*3.0) cout << "They are equal!";
else cout << "They are NOT equal.";
```

这里应该是不等，回忆ICS的IEEE-754标准可以知道，1/3不能被二进制精确表示，所以是循环的并且不精确的

Using Tolerances

```
double x = 1.0;
double y = x / 3.0;
if (fabs(x-y*3.0) <
    numeric_limits<double>::epsilon)
    cout << "They are equal!";
else cout << "They are NOT equal.";
```

解决方式1：给定一个可以接受的较小的误差范围，若结果位于此误差范围内，则认为两者相同，这种方式应该在之前的物理实验中使用过的，很重要的数据处理方式。

A Crucial Point

Mathematically correct

$$\neq$$

Numerically sound

Rarely if ever should the operator $==$ and its equivalents be used on fractional values. Instead, some tolerance should be used to check if they are equal.

Counting in Binary: Integer

Motivation
Representing Numbers
Exotic Representation
Error
Practical Aspects

$$
\begin{aligned}
463 &= 256 + 128 + 64 + 8 + 4 + 2 + 1 \\
&= 2^8 + 2^7 + 2^6 + 2^3 + 2^2 + 2^1 + 2^0
\end{aligned}
$$

$$\downarrow$$

1	1	1	0	0	1	1	1	1
2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0

Counting in Binary: Fractional

$$
\begin{aligned}
463.25 &= 256 + 128 + 64 + 8 + 4 + 2 + 1 + 1/4 \\
&= 2^8 + 2^7 + 2^6 + 2^3 + 2^2 + 2^1 + 2^0 + 2^{-2}
\end{aligned}
$$

$$\downarrow$$

1	1	1	0	0	1	1	1	1	0	1
2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	2^{-1}	2^{-2}

Motivation
Representing Numbers
Exotic Representation
Error
Practical Aspects

Familiar Problem

$$\frac{1}{3} = 0.0101010101\ldots_2$$

Finite number of bits

Fixed-Point Arithmetic

Motivation
Representing Numbers
Exotic Representation
Error
Practical Aspects

1	1	...	0	0	...	1	1
2^{ℓ}	$2^{\ell-1}$	...	2^0	2^{-1}	...	2^{-k+1}	2^{-k}

- Parameters: $k, \ell \in Z$
- $k + \ell + 1$ digits total
- Can reuse integer arithmetic (fast; GPU possibility):

$$a + b = (a \cdot 2^k + b \cdot 2^k) \cdot 2^{-k}$$

Motivation
Representing Numbers
Exotic Representation
Error
Practical Aspects

Two-Digit Example

$$0.1_2 \times 0.1_2 = 0.01_2 \cong 0.0_2$$

这里的精确度只给到了M=1位。

Multiplication and division easily change order of magnitude!

Demand of Scientific Applications

$$9.11 \times 10^{-31} \to 6.022 \times 10^{23}$$

Desired: graceful transition

Motivation
Representing Numbers
Exotic Representation
Error
Practical Aspects

Observations

- Compactness matters:

$$6.022 \times 10^{23} =$$

602,200,000,000,000,000,000,000

Observations

- Compactness matters:

$$6.022 \times 10^{23} =$$

602,200,000,000,000,000,000,000

- Some operations are unlikely:

$$6.022 \times 10^{23} + 9.11 \times 10^{-31}$$

这个add是没有效果的。

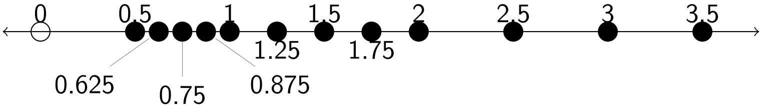
Scientific Notations

Store Significant digits

$$\underbrace{\pm}_{\text{sign}} \underbrace{(d_0 + d_1 \cdot b^{-1} + d_2 \cdot b^{-2} + \cdots + d_{p-1} \cdot b^{1-p}))}_{\text{significand}} \times \underbrace{b^e}_{\text{exponent}}$$

- Base: $b \in N$
- Precision: $p \in N$
- Range of exponents: $e \in [L, U]$

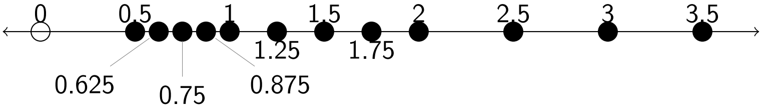
Properties of Floating Point



分布不均匀。

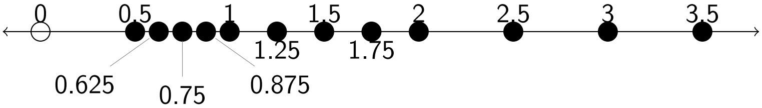
- Unevenly spaced
  - Machine precision ϵ_m: smallest ϵ_m with $1 + \epsilon_m \not\approx 1$

Properties of Floating Point



- Unevenly spaced
  - Machine precision ϵ_m: smallest ϵ_m with $1 + \epsilon_m \ncong 1$
- Needs rounding rule (e.g. "round to nearest, ties to even")

Properties of Floating Point



- Unevenly spaced
  - Machine precision ϵ_m: smallest ϵ_m with $1 + \epsilon_m \ncong 1$
- Needs rounding rule (e.g. "round to nearest, ties to even")
- Can remove leading 1 (normalized的情况下)

Infinite Precision

对浮点数精确表示的方式2：使用两个整数的分式形式来表示这个小数

$$Q = \{a/b : a, b \in Z\}$$

- Simple rules: $a/b + c/d = (ad + cb)/bd$
- Redundant: $1/2 = 2/4$
- Blowup:

$$\frac{1}{100} + \frac{1}{101} + \frac{1}{102} + \frac{1}{103} + \frac{1}{104} + \frac{1}{105} = \frac{188463347}{3218688200}$$

- Restricted operations: $2 \mapsto \sqrt{2}$

Bracketing

Motivation
Representing Numbers
Exotic Representation
Error
Practical Aspects

Store range $a \pm \epsilon$

- Keeps track of certainty and rounding decisions
- Easy bounds:

$$(x \pm \epsilon_1) + (y \pm \epsilon_2) = (x + y) \pm (\epsilon_1 + \epsilon_2 + error(x + y))$$

- Implementation via operator overloading

Sources of Error

- Rounding (or truncation) error (e.g. PI) (估计误差(一般是计算机精度不足引起的))
- Discretization error (e.g. derivative: divided differences) (离散型误差，一般是由于对于连续问题的求解(如微分方程等)无法精确求解而采用离散的方式进行估算)
- Modeling error (e.g. butterfly for weather, g)
- Input error (e.g. approximated parameters, typos)

Example

What sources of error might affect planets simulation?

Motivation
Representing Numbers
Exotic Representation
Error
Practical Aspects

Absolute vs. Relative Error

(绝对误差)

Absolute Error

The difference between the approximate value and the underlying true value.

Absolute vs. Relative Error

Absolute Error

The difference between the approximate value and the underlying true value.

(相对误差)

Relative Error

Absolute error divided by the true value.

Absolute vs. Relative Error

Absolute Error

The difference between the approximate value and the underlying true value.

Relative Error

Absolute error divided by the true value.

$$2\ cm \pm 0.02\ cm$$

$$2\ cm \pm 1\%$$

Example: Catastrophic cancellation

$d \equiv 1 - 0.99 = 0.01$
± 0.004
$d = 0.01 \pm 0.008$

Absolute error $= 0.008$

Relative error $=$? 80%

由上面例子可知，相对误差相比于绝对误差更有意义，因为我们应更关注估计值相对于理论值的偏离程度(%)，而不是偏离量的大小，因为基数可能很小(如上)

Relative Error: Difficulty

Motivation
Representing Numbers
Exotic Representation
Error
Practical Aspects

Problem: Generally not computable

Relative Error: Difficulty

Problem: Generally not computable

Common fix: Be conservative 保守处理

Computable Measures of Success

Motivation
Representing Numbers
Exotic Representation
Error
Practical Aspects

Root-finding problem

For $f : \mathbb{R} \to \mathbb{R}$, find x^* such that $f(x^*) = 0$

Actual output: x_{est} with $|f(x_{est})| \ll 1$

May not be able to evaluate $|x_{est} - x_0|$

Can compute $|f(x_{est}) - f(x_0)| \equiv f(x_{est})$ (a calculable proxy)

Forward Error

(正向误差)

Forward Error

The difference between the approximated and actual solution.

(理论值->实际值)

Backward Error

Motivation
Representing Numbers
Exotic Representation
Error
Practical Aspects

(反向误差)

Backward Error

The amount the problem statement would have to change to make the approximate solution exact

(实际估计解->(修正得出实际解的条件)理论真实解)

Backward Error

Backward Error

The amount the problem statement would have to change to make the approximate solution exact

Example 1: $\sqrt{x}$ (e.g. x=2)

Backward Error

Motivation
Representing Numbers
Exotic Representation
Error
Practical Aspects

Backward Error

The amount the problem statement would have to change to make the approximate solution exact

Example 1: $\sqrt{x}$ (e.g. x=2)
Example 2: $A\vec{x} = \vec{b}$

Conditioning

Motivation
Representing Numbers
Exotic Representation
Error
Practical Aspects

What if backward error is small but nonzero?
Does this condition necessarily imply small forward error?

Conditioning

Motivation
Representing Numbers
Exotic Representation
Error
Practical Aspects

What if backward error is small but nonzero?
Does this condition necessarily imply small forward error?

Well-conditioned (or insensitive):
Small backward error $\Longrightarrow$ small forward error

Motivation
Representing Numbers
Exotic Representation
Error
Practical Aspects

Conditioning

What if backward error is small but nonzero?
Does this condition necessarily imply small forward error?

Well-conditioned (or insensitive):
Small backward error $\Longrightarrow$ small forward error

Poorly conditioned (or sensitive/stiff):
Otherwise

Example: Root-finding: $ax = b \to x_0 \equiv b/a$
Hint: calculate forward and backward errors, check $|a| \ll 1, or|a| \gg 1$

正向误差:x-x0，反向误差：b-ax=a(x-x0)(x0为理论真实解),由此可知，a<<1时小的反向误差不一定对应于小的正向误差，故为poorly conditioned，a>>1时满足well-conditioned的条件

Motivation
Representing Numbers
Exotic Representation
Error
Practical Aspects

Condition Number

Condition number

Ratio of forward to backward error

条件数很大->poorly-conditioned
条件数很小->well-conditioned

Motivation
Representing Numbers
Exotic Representation
Error
Practical Aspects

Condition Number

Condition number

Ratio of forward to backward error

Root-finding example: f(x) = 0

$$c = \frac{1}{|f'(x^*)|}$$

Motivation
Representing Numbers
Exotic Representation
Error
Practical Aspects

Common Cause of Bugs in Numerical Software

Beware of operations that transition between orders of magnitude, like division by small values and subtraction of similar quantities.

E.g. $AX = b$

Theme

Motivation
Representing Numbers
Exotic Representation
Error
Practical Aspects

Extremely careful implementation can be necessary.

Example: Vector Norms $\|\vec{x}\|_2$

Motivation
Representing Numbers
Exotic Representation
Error
Practical Aspects

```
double normSquared = 0;
for (int i = 0; i < n; i++)
normSquared += x[i]*x[i];
return sqrt(normSquared);
```

Overflow issue

这个程序表面上是对的，但是会有可能出现bug：
x[i]过大时会导致乘积overflow或者过小时会underflow，另外，若两次乘积数量级相差较大，还可能出现被直接忽略的情况。

Improved $\|\vec{x}\|_2$

```
double maxElement = epsilon;

for (int i = 0; i < n; i++)
maxElement = max(maxElement, fabs(x[i]));
for (int i = 0; i < n; i++) {
double scaled = x[i] / maxElement;
normSquared += scaled*scaled;
}
return sqrt(normSquared) * maxElement;
```

第一次遍历找出最大值

第二次遍历中，在每次平方之前先除以最大值，来约束每次运算的精度

Motivation
Representing Numbers
Exotic Representation
Error
Practical Aspects

More Involved Example: Large Scale Summation $\Sigma_i x_i$

```
double sum = 0;
for (int i = 0; i < n; i++)
sum += x[i];
```

Motivation
Representing Numbers
Exotic Representation
Error
Practical Aspects

Simple Sum and Kahan Sum

function SIMPLE-SUM($\vec{x}$)
    $s \leftarrow 0$ ▷ Current total
    for $i \leftarrow 1, 2, \ldots, n : s \leftarrow s + x_i$
    return s

(a)

Motivation
Representing Numbers
Exotic Representation
Error
Practical Aspects

Simple Sum and Kahan Sum

```
function SIMPLE-SUM(x⃗)
    s ← 0                                ▷ Current total
    for i ← 1, 2, ..., n : s ← s + x_i
    return s
```

(a)

$$((a+b)-a)-b \stackrel{?}{=} 0$$

不一定为0，因为存在越界或者数量级相差大等情况。

Store compensation value !

Simple Sum and Kahan Sum

```
function SIMPLE-SUM(x⃗)
    s ← 0                                ▷ Current total
    for i ← 1, 2, ..., n : s ← s + x_i
    return s
```

(a)

kahan算法的关键就在于：
c<-v - (s_next - s);
其主要目的就是在计算
当次计算一起的偏差值，
但是实际上若未发生flow的话
为0，并且如果偏差是由于数量
级相差过大而引起的话并不能
很好的解决。

$$((a+b)-a)-b \stackrel{?}{=} 0$$

Store compensation value !

```
function KAHAN-SUM(x⃗)
    s, c ← 0                                  ▷ Current total and compensation
    for i ← 1, 2, ..., n
        v ← x_i + c                           ▷ Try to add x_i and compensation c to the sum
        s_next ← s + v                        ▷ Compute the summation result of this iteration
        c ← v − (s_next − s)                  ▷ Compute compensation using the Kahan error estimate
        s ← s_next                            ▷ Update sum
    return s
```

(b)