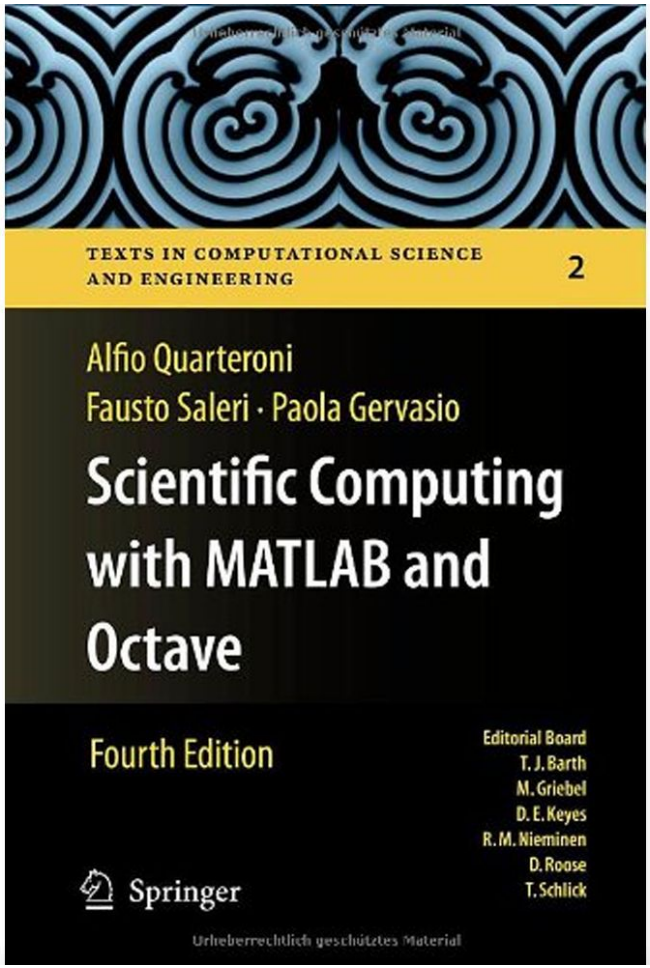
数值计算方法：原理、算法和应用
Numerical Methods: Principles, Algorithms and Applications

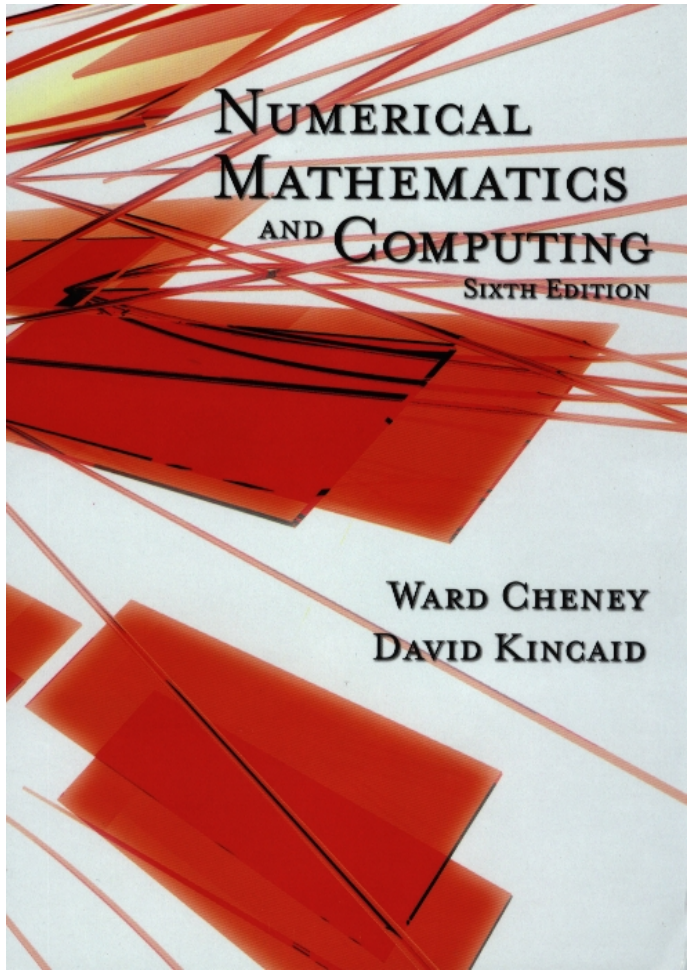
授课教师：周铁

北京大学数学科学学院

2021 年 9 月 14 日

主要参考书


TEXTS IN COMPUTATIONAL SCIENCE AND ENGINEERING
2
Alfio Quarteroni
Fausto Saleri · Paola Gervasio
Scientific Computing with MATLAB and Octave
Fourth Edition
Editorial Board
T. J. Barth
M. Griebel
D. E. Keyes
R. M. Nieminen
D. Roose
T. Schlick
Springer
Urheberrechtlich geschütztes Material


NUMERICAL MATHEMATICS AND COMPUTING
SIXTH EDITION
WARD CHENEY
DAVID KINCAID

数学软件

- MATLAB (MATrix LABoratory)
- GAUSS
- Mathematica
- Maple
- GNU Octave
- Scilab
- Python

https://en.wikipedia.org/wiki/List_of_numerical-analysis_software

https://en.wikipedia.org/wiki/Comparison_of_numerical-analysis_software

先修课，学习环节

先修课程

- 高等数学 (微积分，常微分方程)
- 线性代数
- 计算概论
- 概率统计，微分方程

学习环节

- 教材阅读
- 数值试验
- 课堂听讲
- 课堂讨论
- 课堂报告

引论

数值计算 (Numerical Computation)

- 是专门研究如何利用计算机解决各类科学和工程问题的学科.
- 是伴随着数字式电子计算机的发明而形成的学科，是数学与计算机科学交叉的产物.
- 非数值计算 (Non-Numeric Computation)，符号计算 (Symbolic Computation)

同义词：**数值模拟 (Numerical Simulation)**
科学计算 (Scientific Computation)
计算科学 (Computational Science)

- **实验分析/理论分析/科学计算**是当今科学技术工作的三种方式.

https://en.wikipedia.org/wiki/Computational_science

数值计算的主要内容

- 非线性方程数值解
- 函数逼近
- 数值微分与积分
- 线性方程组数值解
- 特征值问题数值解
- 数值最优化
- 常微分方程数解值
- 偏微分方程数解值

数值计算的过程

数值计算工作包含如下环节：

1. 对实际问题建立数学模型；
2. 数学模型的理论分析；
3. 设计算法并进行理论分析；
4. 编制程序，上机运行，展示计算结果；
5. 将计算结果与理论分析和实验的结果相结合给出实际问题的解答，或提出对数学模型的修正方案.

上述3.和4.两个环节正是“**数值计算方法**”这门课程涉及的主要内容，其核心是算法的原理.
实际问题多种多样，数学模型就那几种.
典型数学问题的数值求解方法及理论分析为大多数实际问题的数值模拟奠定了基础.
数值计算方法课程将着重介绍几类典型数学问题的数值解法.

计算中的误差

数值模拟的过程牵扯到三种意义不完全相同的“解”.

- 实际问题的解.
- 数学模型的解.
- 数值计算的解.
- 实际问题的解和其数学模型的解一般都是未知的.
- 数值模拟的解称为**数值解** (计算解)，它是数学模型解的一个近似.
- 实际问题的解与其数学模型的数值计算得到的数值解之间的差别就是误差.

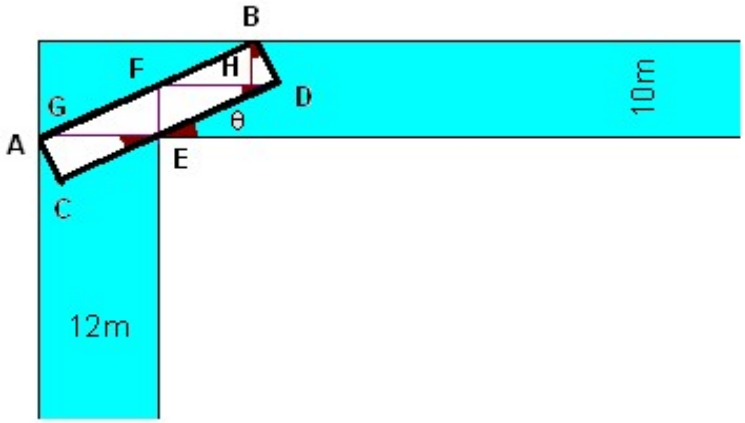
数值计算永远伴随着误差！误差的来源有如下四个方面.

四种误差

- **模型误差**　实际问题抽象成数学模型时往往忽略了许多次要因素. 即使数学模型能找到准确解，它也与实际问题的真解不同，它们的差别称为模型误差. 很多时候模型误差是四种误差中最大的. 模型的建立往往不是数学家的工作.
- **观测误差**　原始数据是由仪器观测和记录而获得的. 由于仪器的精密性有限、周围环境的变化以及人的工作态度和能力等因素，而使数据必然带有误差，这种误差称做观测误差.
- **截断误差**　数学模型的精确解往往要用无限次的运算才能获得，而实际计算时只能用有限次运算的结果来近似，这样引起的误差称做截断误差.
- **舍入误差**　计算机的字长有限，超出计算机字长的数据都要经过舍入或截断处理，这样引起的误差就称做舍入误差.

下面举个例子：

以一个比较简单的实际问题为例. 一艘宽为 5 米的船要通过某河道的直角弯，河道的宽为 10 米和 12 米，试问要驶过该直角弯，船的长度不能超过多少米?



解：如图做一个宽 5 米，长 $f(\theta)$ 米的矩形，不难得到

$$f(\theta) = CE + ED = \frac{12 - 5\sin\theta}{\cos\theta} + \frac{10 - 5\cos\theta}{\sin\theta}, \quad \theta \in (0, \pi/2).$$

船要想通过弯道，其长度不能超过函数 $f(\theta)$ 的最小值. 这个实际问题的数学模型就是一元函数的最小化问题：

$$\min_{\theta\in(0,\pi/2)} f(\theta)$$

要想找到函数 $f(\theta)$ 的最小值，只要解方程：

$$f'(\theta) = \frac{12\sin\theta - 5}{\cos^2\theta} + \frac{5 - 10\cos\theta}{\sin^2\theta} = 0, \quad \text{for } \theta \in (0, \pi/2).$$

容易证明

$$\lim_{\theta\to 0+} f'(\theta) = -\infty, \quad \lim_{\theta\to \pi/2-} f'(\theta) = +\infty, \quad f''(\theta) > 0,\ \theta \in (0, \pi/2)$$

所以数学模型有惟一解. 但是解 $\theta^* = arg\min_{0<\theta<\pi/2} f(\theta)$ 的数值只能用数值方法计算得到近似值

$$\theta^* \approx 0.73, \quad f(\theta^*) \approx 21.$$

模型误差：河道和船的形状规则，弯道是直角，紧贴河岸行驶.
观测误差：各个已知数据.
截断误差：求解非线性方程的方法.
舍入误差：在计算机上进行数值运算.

绝对误差与相对误差

设 x 为精确值，$\tilde{x}$ 为近似值，称 $e(x) = |x - \tilde{x}|$ 为**绝对误差**.
精确值未知时，只能设法估计绝对误差 $e(x)$ 的上界 $e(x) \leqslant \varepsilon$.
这个上界 ε 称为近似值 $\tilde{x}$ 的**绝对误差限**.
精确值非零时，称绝对误差与精确值之比

$$e_r(x) = \frac{e(x)}{|x|} = \frac{|x - \tilde{x}|}{|x|} \quad (x \neq 0)$$

为近似值 $\tilde{x}$ 的**相对误差**，而称

$$\varepsilon_r = \frac{\varepsilon}{|x|} \quad (x \neq 0)$$

为**相对误差限**. 由于 x 一般未知，常常用

$$e_r(x) = \frac{|x - \tilde{x}|}{|\tilde{x}|}, \quad \varepsilon_r = \frac{\varepsilon}{|\tilde{x}|} \quad (|\tilde{x}| \neq 0)$$

表示相对误差和相对误差限.

实数的规范形式 (normalized notation)

一个非零实数 x 总可以写成 10 进制规范小数形式：

$$x = \pm 0.d_1 d_2 \cdots d_i \cdots \times 10^e$$

其中的 $d_i \in \{0, 1, 2, \ldots, 9\}, d_1 \neq 0$， $e \in \mathbb{Z}$.

$$37.21829 = 0.3721829 \times 10^2$$
$$0.002271828 = 0.2271828 \times 10^{-2}$$

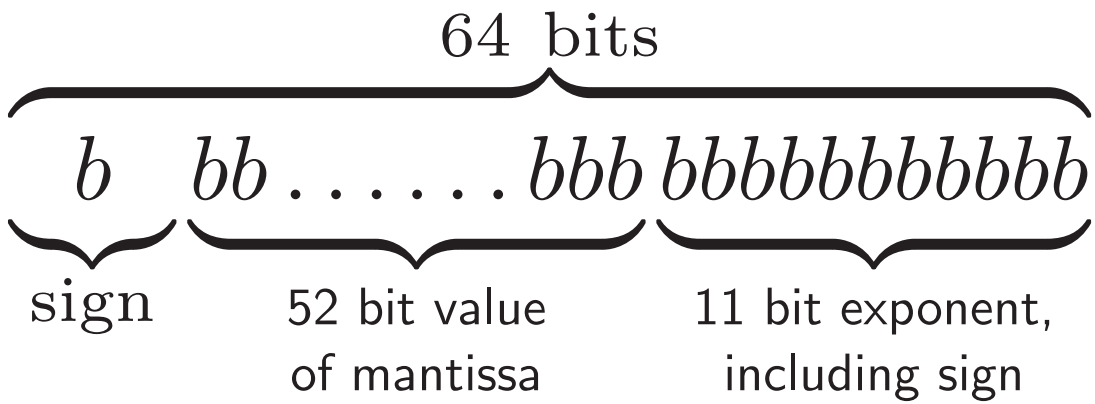
在电子计算机中更方便的是采用 2 进制

$$x = (-1)^s \times 0.d_1 d_2 \cdots d_i \cdots \times 2^e$$

其中的 $d_i \in \{0, 1\}, d_1 = 1$， $s \in \{0, 1\}$， e 为 2 进制整数.

浮点数系统 (floating-point number system)

$$x = (-1)^s \times 0.d_1 d_2 \cdots d_t \times \beta^e = (-1)^s \times d_1 . d_2 \cdots d_t \times \beta^{e-e_0}, \quad d_1 \neq 0.$$



在大部分计算机上，实数系 $\mathbb{R}$ 是用二进制浮点数系统 $\mathbb{F}(2, t, e_{min}, e_{max})$ 表示的，

$$\mathbb{F}(2, t, e_{min}, e_{max}) = \{(-1)^s \times 1 . d_2 \cdots d_t \times 2^{e-(-e_{min}+1)}\} \cup \{0\},$$

其中 $1 \leq e \leq e_{max} - e_{min} + 1$, $d_j \in \{0, 1\}$, $t \in \mathbb{N}_+$ 是小数部分占据的字长.

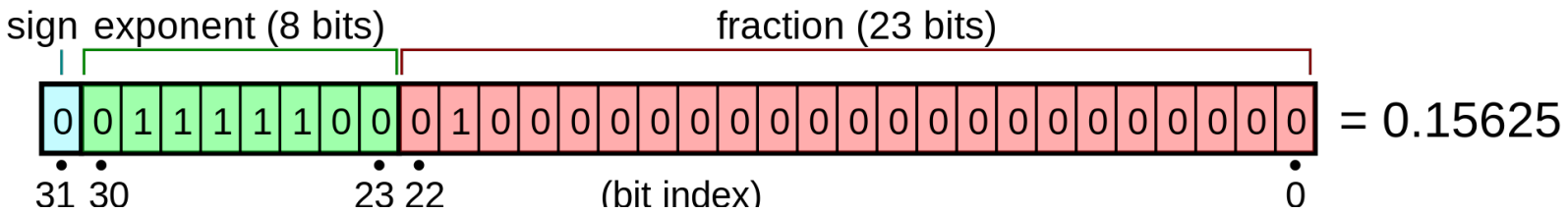
历史上产生过很多种浮点数系统.
其中 IEEE standard floating-point(IEEE 754) 规定的浮点数系统最常用.
IEEE 754 是 1985 年以后的计算机生产标准，最新修改于 2019 年 6 月.

系统	进制	t	e_{min}	e_{max}
IEEE 754 single-precision	2	24	-126	127
IEEE 754 double-precision	2	53	-1022	1023

Format	Precision	Exponent	Approx range	Error
single	24 bits	8 bits	$10^{\pm 38}$	10^{-8}
double	53 bits	11 bits	$10^{\pm 308}$	10^{-16}
extended	$\geq$ 64 bits	$\geq$ 15 bits	$10^{\pm 4932}$	10^{-20}

`https://en.wikipedia.org/wiki/IEEE_754`
William Kahan (UC Berkeley), Turing Award 1989.

https://en.wikipedia.org/wiki/Single-precision_floating-point_format 上给了一个单精度浮点数的例子：



$$(-1)^{b_{31}} \times 2^{(b_{30}b_{29}\cdots b_{23})_2 - 127} \times (1.b_{22}b_{21}\cdots b_0)_2$$

$$\text{sign} = b_{31} = 0 \Rightarrow (-1)^{\text{sign}} = (-1)^0 = +1$$

$$e = b_{30}b_{29}\cdots b_{23} = \sum_{i=0}^{7} b_{23+i} \cdot 2^i = 124 \Rightarrow 2^{124-127} = 2^{-3}$$

$$1.b_{22}b_{21}\cdots b_0 = 1 + \sum_{i=1}^{23} b_{23-i} \cdot 2^{-i} = 1 + 1 \cdot 2^{-2} = 1.25$$

$$\text{value} = (+1) \times 2^{-3} \times 1.25 = +0.15625$$

浮点数系统的特点：

1. $\mathbb{F}$ 是有理数集合 $\mathbb{Q}$ 的有限子集，一定会有“溢出”现象.
2. 绝大部分实数都不能在浮点数系统中精确表示.

任取一个在 IEEE 754 单精度系统中不溢出的正实数 x, 它总可以表示成

$$x = 0.1d_2d_3\cdots d_{24}d_{25}d_{26}\cdots \times 2^e \quad (-126 \le e \le 127)$$

由于 IEEE 754-SP 中尾数只有 24 位，所以在 $\mathbb{F}(2, 24, -126, 127)$ 中存在

$$x_- = 0.1d_2d_3\cdots d_{24} \times 2^e \in \mathbb{F}(2, 24, -126, 127)$$

和

$$x_+ = [0.1d_2d_3\cdots d_{24} + 2^{-24}] \times 2^e \in \mathbb{F}(2, 24, -126, 127)$$

使得 x 属于闭区间 $[x_-, x_+]$，而且 $(x_-, x_+) \cap \mathbb{F}(2, 24, -126, 127) = \emptyset$.

用映射：

$$x \longmapsto fl(x)$$

把实数 x 对应的浮点数记为 $fl(x)$，显然应该取上述 x_-, x_+ 两数中距离 x 比较近的那个为 $fl(x)$. 如果 $fl(x) = x_-$，则有

$$|x - fl(x)| = |x - x_-| \leq \frac{1}{2}|x_+ - x_-| = 2^{-25} \times 2^e.$$

如果 $fl(x) = x_+$，不难验证这个不等式也成立.
所以 $fl(x)$ 的相对误差限为：

$$\left|\frac{fl(x) - x}{x}\right| \leq \frac{2^{-25} \times 2^e}{0.1d_2d_3\cdots \times 2^e} \leq \frac{2^{-25}}{2^{-1}} = 2^{-24} =: \frac{1}{2}\epsilon_{mach}.$$

$\epsilon_{mach} = 2^{-23}$ 称为 IEEE 754 单精度系统的**机器精度**.

IEEE 754 双精度浮点数系统 $\mathbb{F}(2, 53, -1021, 1024)$ 的机器精度为：

$$\epsilon_{mach} = 2^{-52} \approx 2.2204 \times 10^{-16}$$

MATLAB 缺省使用 IEEE 754 双精度浮点数系统.

```
>> eps
ans = 2.2204e-16
>> realmax
ans = 1.797693134862316e+308
>> realmin
ans = 2.225073858507201e-308
>> eps*realmin
ans = 4.940656458412465e-324
>> 0.5*eps*realmin
ans =  0
```

```
x = 1; while 1+x>1; x1 = x; x = x/2; end; x1 %获得eps
```

事实上，$fl(x)$ 与 x 的关系可以写成等式

$$fl(x) = x\,(1 + r)$$

其中 r 依赖于 x 且 $|r\,| \leq \epsilon_{mach}/2.$

这是因为，对 $x \neq 0$，只要记

$$r = \frac{fl(x) - x}{x}$$

就有 $fl(x) = x\,(1 + r)$，而且有

$$|r\,| = \frac{|fl(x) - x\,|}{|x\,|} \leq \frac{1}{2}\epsilon_{mach}.$$

机器精度决定了 $fl(x)$ 的相对误差限.

浮点数的运算

取 $a, b \in \mathbb{F}(2, t, e_{min}, e_{max})$，它们在计算机中做“浮点四则运算”与标准四则运算并不完全相同，有些运算规律 (比如结合律和分配律) 不再成立，零元素也不再惟一.

```
1 >> a=1;b=1;  while a+b not= a;  b=b/2; end; b
2  b = 1.1102e-16
```

用符号 $\odot$ 表示四则运算 $+, -, \times, \div$ 的任一种，如果 $a, b \in \mathbb{F}$，根据前面的讨论，有

$$fl(a \odot b) = (a \odot b)(1 + r), \quad |r| \leq \epsilon_{mach}/2.$$

这说明每次浮点计算都会产生误差.

```
>> s = 0;
>> for i = 1:10000 s = s + 0.0001; end
>> s
 s = 0.999999999999906

>> s = [0:0.0001:1];
>> s(10001)
 ans =
 1
```

有效数字

的概念：以实数 $x = \pi/10$ 为例，

$$x = \frac{\pi}{10} = 0.31415926\cdots\cdots$$

如果用只有 5 位小数的

$$\tilde{x} = 0.31416$$

作为 $x = \pi/10$ 的近似值，则 $\tilde{x}$ 的每一位数字 $3, 1, 4, 1, 6$ 都是有效数字. 如果用

$$\hat{x} = 0.31415$$

作为 $x = \pi/10$ 的近似值，则 $\hat{x}$ 的最后一位数字 5 就不是有效数字. 这是因为它们的绝对误差分别满足

$$|x - \tilde{x}| \le \frac{1}{2} \times 10^{-5}, \quad |x - \hat{x}| > \frac{1}{2} \times 10^{-5}$$

如果实数 x 的一个近似值 $\tilde{x}$ 有规范形式

$$\tilde{x} = \pm 0.a_1 a_2 \cdots a_n a_{n+1} \cdots \times 10^m$$

并且绝对误差 $|x - \tilde{x}|$ 满足不等式

$$|x - \tilde{x}| \leqslant \frac{1}{2} \times 10^{-n} \times 10^m = \frac{1}{2} \times 10^{m-n}$$

则称 $a_1, a_2, \cdots, a_n$ 为 $\tilde{x}$ 中的有效数字，a_{n+1} 开始往后的数字就可能没有保留意义. 当然，如果

$$|x - \tilde{x}| \leqslant \frac{1}{2} \times 10^{-(n+1)} \times 10^m = \frac{1}{2} \times 10^{m-(n+1)}$$

则 a_{n+1} 也是有效数字.

有效数字位数与相对误差限密切相关，具有 n 位有效数字与相对误差限的量级为 10^{-n} 是等价的.

取两个数 $x = 0.37214\,48693,\ y = 0.37202\,14371$, 在一个 5 位精度的 10 进制计算机上计算 $x - y$.
首先，这两个数往此计算机里一存，就产生两个有 5 位有效数字的近似值

$$\tilde{x} = 0.37214, \qquad \tilde{y} = 0.37202$$

分别做减法得到

$$\tilde{x} - \tilde{y} = 0.00012$$
$$x - y = 0.0001234322$$

浮点减法的相对误差达到

$$\frac{|(\tilde{x} - \tilde{y}) - (x - y)|}{|x - y|} = \frac{0.0000034322}{0.0001234322} \approx 2.8\%$$

而且 $\tilde{x} - \tilde{y} = 0.12000 \times 10^{-3}$ 只有两位有效数字!
有效数字消失 (loss of significance, catastrophic cancellation).

浮点运算的稳定性

加法：设 $a, b \in \mathbb{R}$，在 $\mathbb{F}$ 中不溢出，非零且同号. 由于

$$fl(a) = a(1+r_1), fl(b) = b(1+r_2), fl(fl(a)+fl(b)) = (fl(a)+fl(b))(1+r_3)$$

所以有：

$$\begin{aligned} & fl(fl(a)+fl(b)) - (a+b) \\ &= fl(a)+fl(b)+(fl(a)+fl(b))r_3 - a - b \\ &= a(1+r_1)+b(1+r_2)+a(1+r_1)r_3+b(1+r_2)r_3 - a - b \\ &= ar_1+br_2+a(1+r_1)r_3+b(1+r_2)r_3 \end{aligned}$$

$$\frac{|fl[fl(a)+fl(b)]-(a+b)|}{|a+b|} \leq |r_1|+|r_2|+|(1+r_1)r_3|+|(1+r_2)r_3|$$

这说明浮点加法是稳定的.

减法：设 $a, b \in \mathbb{R}$，互异，在 $\mathbb{F}$ 中不溢出，同号，非零. 由于

$$fl[fl(a) - fl(b)] = [fl(a) - fl(b)](1 + r_5)$$

所以有：

$$\begin{aligned} & fl[fl(a) - fl(b)] - (a - b) \\ &= fl(a) - fl(b) + [fl(a) - fl(b)]r_5 - a + b \\ &= ar_1 - br_2 + a(1 + r_1)r_5 - b(1 + r_2)r_5 \\ &= (ar_1 - br_2)(1 + r_5) + (a - b)r_5 \end{aligned}$$

$$\begin{aligned} \frac{|fl[fl(a) - fl(b)] - (a - b)|}{|a - b|} &\leq |r_5| + \frac{|ar_1| + |br_2|}{|a - b|}|1 + r_5| \\ &\leq \frac{1}{2}\epsilon_{mach} + \frac{|a| + |b|}{1.9|a - b|}\epsilon_{mach} \end{aligned}$$

这表明两个相近数做浮点减法是不稳定的!

乘法：设 $a, b \in \mathbb{R}$，在 $\mathbb{F}$ 中不溢出，非零. 由于

$$fl[fl(a)fl(b)] = fl(a)fl(b)(1 + r_4)$$

所以

$$\begin{aligned} & fl[fl(a)fl(b)] - ab \\ &= fl(a)fl(b) + fl(a)fl(b)r_4 - ab \\ &= ab(r_1 + r_2 + r_1 r_2) + ab(1 + r_1)(1 + r_2)r_4 \end{aligned}$$

所以

$$\frac{|fl[fl(a)fl(b)] - ab|}{|ab|} \leq |r_1| + |r_2| + |r_1 r_2| + |(1 + r_1)(1 + r_2)r_4|$$

这说明浮点乘法是稳定的.

除法：设 $a, b \in \mathbb{R}$，在 $\mathbb{F}$ 中不溢出，非零. 由于

$$fl\left[\frac{fl(a)}{fl(b)}\right] = \frac{fl(a)}{fl(b)}(1 + r_6)$$

所以

$$\begin{aligned} & fl\left[\frac{fl(a)}{fl(b)}\right] - \frac{a}{b} \\ &= \frac{fl(a)}{fl(b)} + \frac{fl(a)}{fl(b)} r_6 - \frac{a}{b} \\ &= \frac{a(1+r_1)}{b(1+r_2)} + \frac{a(1+r_1)}{b(1+r_2)} r_6 - \frac{a}{b} \\ &= \frac{a}{b}\left(\frac{1+r_1}{1+r_2} - 1\right) + \frac{a(1+r_1)}{b(1+r_2)} r_6 \end{aligned}$$

所以

$$\frac{|fl[fl(a)/fl(b)] - a/b|}{|a/b|} \leq \frac{|r_1| + |r_2|}{|1+r_2|} + \frac{|1+r_1|}{|1+r_2|}|r_4|$$

这说明浮点除法是稳定的.

```
>> x = 1.0e-15;
>> ((1+x)-1)/x
 ans = 1.11022302462515 7
```

相对误差高达 11%!

量级相差很大的数做加 (减法)

$$
\begin{aligned}
a = 10^9 &= 0.1000000000 \times 10^{10} \\
b = 2 &= 0.0000000002 \times 10^{10} \\
a + b &= 0.1000000002 \times 10^{10}
\end{aligned}
$$

如果在 $\mathbb{F}(10, 9, e_{min}, e_{max})$ 中，结果就是

$$fl[fl(a) + fl(b)] = fl(a)$$

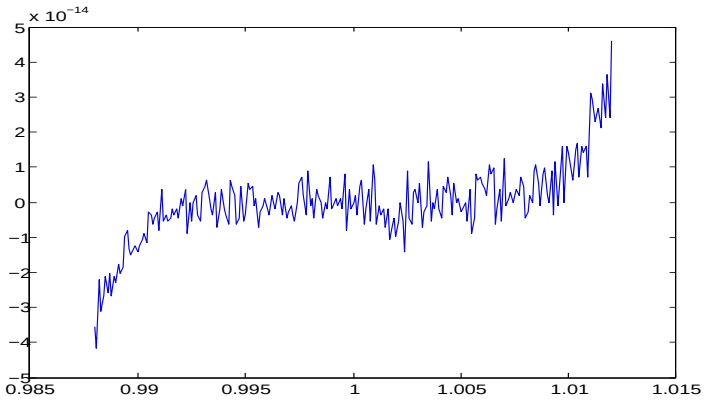
浮点数运算中的“坑”

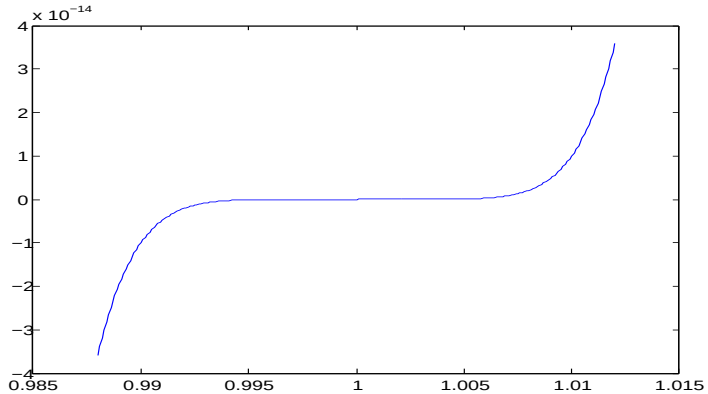
1. 相近数相减 – 误差会强烈放大.
2. 量级相差很大的两个数做除法 –容易溢出 (overflow, underflow).
3. 量级相差很大的两个数做加减法.
4. 不必要的运算步骤 – 运算次数越多，积累误差就越大.
   多项式求值的“嵌套形式 (Horner's nested form)”:

$$p_n(x) = a_n x^n + a_{n-1}x^{n-1} + \cdots + a_1 x + a_0$$
$$p_n(x) = x(x \cdots (x(a_n x + a_{n-1}) + \cdots + a_1) + a_0$$

前者：$2n-1$ 次乘法，n 次加法；
后者：n 次乘法，n 次加法.

```
x = 0.988:0.0001:1.012;
y1 = x.^7-7*x.^6+21*x.^5-35*x.^4+35*x.^3-21*x.^2+7*x-1;
y2 = (x-1).^7;
y3=x.*(x.*(x.*(x.*(x.*(x.*(x-7)+21)-35)+35)-21)+7)-1;
figure
plot(x,y1)
figure
plot(x,y2)
figure
plot(x,y3)
```

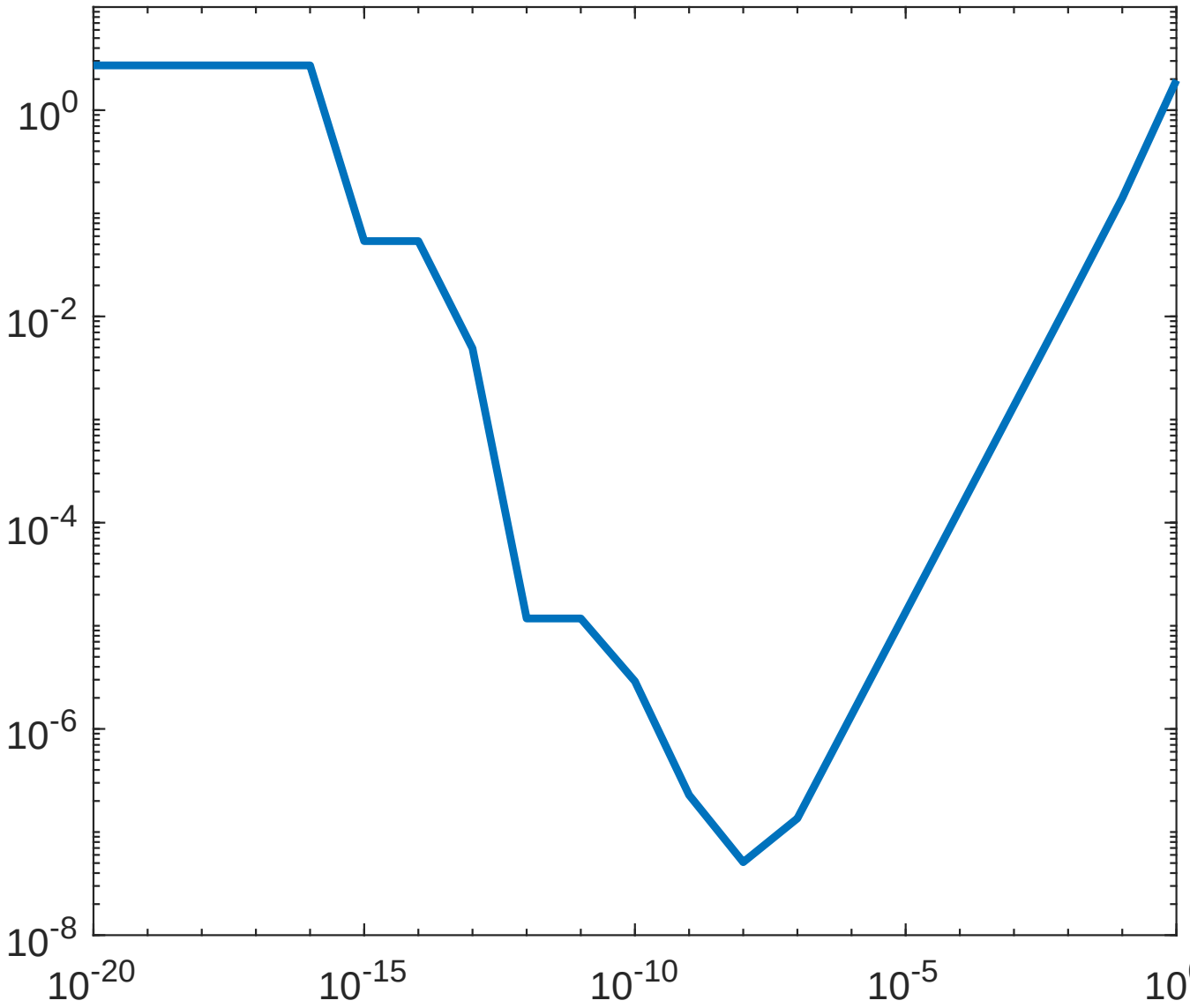




用差商近似导数

$$f(x) = e^x,\ x_0 = 1,\ f'(x_0) \approx \frac{f(x_0+h) - f(x_0)}{h}$$

```
h=10.^(-20:0);
f=@(x) exp(x);
x0=1;
fp=(f(x0+h)-f(x0))./h;
figure
loglog(h,abs(fp-exp(x0)));
```


10^{0}
10^{-2}
10^{-4}
10^{-6}
10^{-8}
10^{-20}
10^{-15}
10^{-10}
10^{-5}
10^{0}

记 $\tilde{f}(x)$ 为 $f(x)$ 的近似，误差上界为 $|\tilde{f}(x) - f(x)| \leq \varepsilon$.
由 Taylor 展开公式：

$$f(x_0 + h) = f(x_0) + hf'(x_0) + \frac{f''(\xi)}{2}h^2,$$

再假设 $\max_x |f''(x)| \leq M$，可得

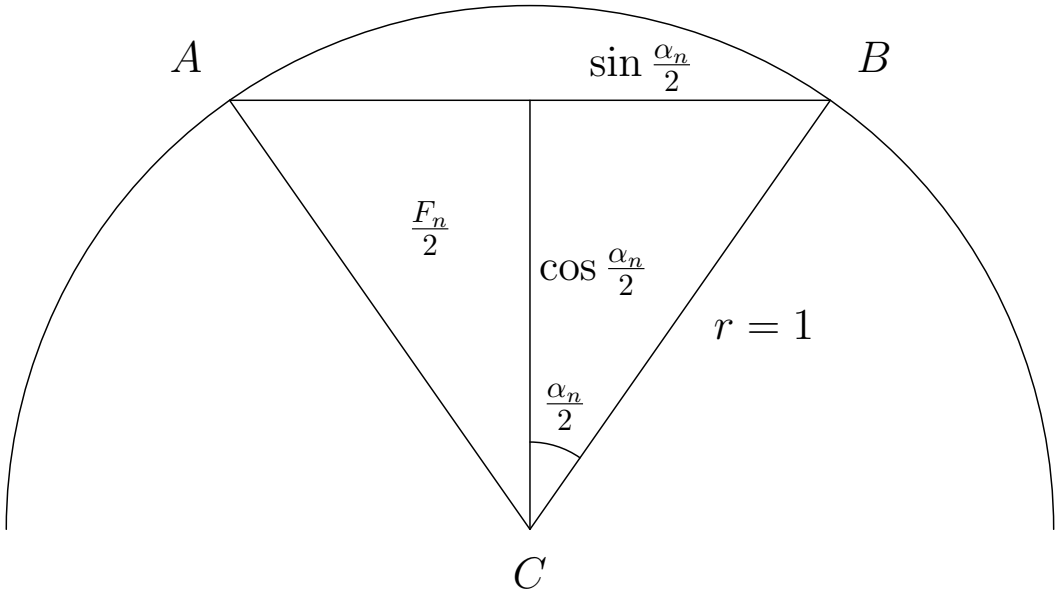
$$\left| \frac{f(x_0 + h) - f(x_0)}{h} - f'(x_0) \right| \leq \frac{Mh}{2}.$$

于是有

$$\left| \frac{\tilde{f}(x_0 + h) - \tilde{f}(x_0)}{h} - f'(x_0) \right| \leq \frac{2\varepsilon}{h} + \frac{Mh}{2}$$

要使误差最小，$h \approx \sqrt{\epsilon_{mach}}$

圆周率的计算–正多边形逼近圆



$$F_n = \cos\frac{\alpha_n}{2}\sin\frac{\alpha_n}{2}$$

$$A_n = nF_n = \frac{n}{2}(2\cos\frac{\alpha_n}{2}\sin\frac{\alpha_n}{2}) = \frac{n}{2}\sin\alpha_n = \frac{n}{2}\sin\frac{2\pi}{n}$$

这个公式不能直接用，因为里面有 π. 从 $n=6$, $\sin\frac{\pi}{3}=\sqrt{3}/2$ 开始，每次把 n 翻倍，角度减半，利用半角公式进行迭代.

圆周率的计算–上述公式的直接实现

已知 $\sin\alpha_n$ 计算 $\sin\frac{\alpha_n}{2}$ 的公式为：

$$\sin\frac{\alpha_n}{2}=\sqrt{\frac{1-\cos\alpha_n}{2}}=\sqrt{\frac{1-\sqrt{1-\sin^2\alpha_n}}{2}}$$

```
1  s=sqrt(3)/2; A=3*s; n=6; % initialization
2  z=[A-pi n A s]; % store the results
3  while s>1e-10 % termination if s=sin(alpha) small
4  s=sqrt((1-sqrt(1-s*s))/2); % new sin(alpha/2) value
5  n=2*n; A=n/2*s; % A=new polygon area
6  z=[z; A-pi n A s];
7  end
8  m=length(z);
9  for i=1 : m
10 fprintf('%10d %20.15f %20.15f\n',z(i,2),z(i,3),z(i,1))
11 end
```

圆周率的计算–结果显然不对!

n	A_n	$A_n - \pi$	$\sin(\alpha_n)$
6	2.59807621135331 6	−0.543516442236477	0.866025403784439
12	3.000000000000000	−0.141592653589794	0.500000000000000
24	3.105828541230250	−0.035764112359543	0.258819045102521
48	3.132628613281237	−0.008964040308556	0.130526192220052
96	3.139350203046872	−0.002242450542921	0.065403129230143
192	3.141031950890530	−0.000560702699263	0.032719082821776
384	3.141452472285344	−0.000140181304449	0.016361731626486
768	3.141557607911622	−0.000035045678171	0.008181139603937
1536	3.141583892148936	−0.000008761440857	0.004090604026236
3072	3.141590463236762	−0.000002190353031	0.002045306291170
6144	3.141592106043048	−0.000000547546745	0.001022653680353
12288	3.141592516588155	−0.000000137001638	0.000511326906997
24576	3.141592618640789	−0.000000034949004	0.000255663461803
49152	3.141592645321216	−0.000000008268577	0.000127831731987
98304	3.141592645321216	−0.000000008268577	0.000063915865994
196608	3.141592645321216	−0.000000008268577	0.000031957932997
393216	3.141592645321216	−0.000000008268577	0.000015978966498
786432	3.141592303811738	−0.000000349778055	0.000007989482381
1572864	3.141592303811738	−0.000000349778055	0.000003994741190
3145728	3.141586839655041	−0.000005813934752	0.000001997367121
6291456	3.141586839655041	−0.000005813934752	0.000000998683561
12582912	3.141674265021758	0.000081611431964	0.000000499355676
25165824	3.141674265021758	0.000081611431964	0.000000249677838
50331648	3.143072740170040	0.001480086580246	0.000000124894489
100663296	3.137475099502783	−0.004117554087010	0.000000062336030
201326592	3.181980515339464	0.040387861749671	0.000000031610136
402653184	3.000000000000000	−0.141592653589793	0.000000014901161
805306368	3.000000000000000	−0.141592653589793	0.000000007450581
1610612736	0.000000000000000	−3.141592653589793	0.000000000000000

圆周率的计算–避免相近数相减

$$
\begin{aligned}
\sin\frac{\alpha_n}{2} &= \sqrt{\frac{1-\sqrt{1-\sin^2\alpha_n}}{2}} \\
&= \sqrt{\frac{1-\sqrt{1-\sin^2\alpha_n}}{2}\frac{1+\sqrt{1-\sin^2\alpha_n}}{1+\sqrt{1-\sin^2\alpha_n}}} \\
&= \sqrt{\frac{1-(1-\sin^2\alpha_n)}{2(1+\sqrt{1-\sin^2\alpha_n})}} \\
&= \frac{\sin\alpha_n}{\sqrt{2(1+\sqrt{1-\sin^2\alpha_n})}}
\end{aligned}
$$

已知 $\sin\alpha_n$ 计算 $\sin\frac{\alpha_n}{2}$ 用这个公式就避免了相近数相减!

圆周率的计算–避免相近数相减

```
oldA=0;s=sqrt(3)/2; newA=3*s; n=6; % initialization
z=[newA-pi n newA s]; % store the results
while newA>oldA % quit if area does not increase
oldA=newA;
s=s/sqrt(2*(1+sqrt((1+s)*(1-s)))); % new sine value
n=2*n; newA=n/2*s;
z=[z; newA-pi n newA s];
end
m=length(z);
for i=1 : m
fprintf('%10d %20.15f %20.15f\n',z(i,2),z(i,3),z(i,1))
end
```

圆周率的计算–这次结果全对了

n	A_n	$A_n - \pi$
6	2.598076211353316	−0.543516442236477
12	3.000000000000000	−0.141592653589793
24	3.105828541230249	−0.035764112359544
48	3.132628613281238	−0.008964040308555
96	3.139350203046867	−0.002242450542926
192	3.141031950890509	−0.000560702699284
384	3.141452472285462	−0.000140181304332
768	3.141557607911857	−0.000035045677936
1536	3.141583892148318	−0.000008761441475
3072	3.141590463228050	−0.000002190361744
6144	3.141592105999271	−0.000000547590522
12288	3.141592516692156	−0.000000136897637
24576	3.141592619365383	−0.000000034224410
49152	3.141592645033690	−0.000000008556103
98304	3.141592651450766	−0.000000002139027
196608	3.141592653055036	−0.000000000534757
393216	3.141592653456104	−0.000000000133690
786432	3.141592653556371	−0.000000000033422
1572864	3.141592653581438	−0.000000000008355
3145728	3.141592653587705	−0.000000000002089
6291456	3.141592653589271	−0.000000000000522
12582912	3.141592653589663	−0.000000000000130
25165824	3.141592653589761	−0.000000000000032
50331648	3.141592653589786	−0.000000000000008
100663296	3.141592653589791	−0.000000000000002
201326592	3.141592653589794	0.000000000000000
402653184	3.141592653589794	0.000000000000001
805306368	3.141592653589794	0.000000000000001

舍入误差积累的灾难后果

1. 1991/02/25，海湾战争，MIM-104 Patriot 导弹防御系统，由于计算机舍入误差积累导致弹道计算错误，漏掉对一个飞毛腿导弹的拦截，致使 28 名美军士兵丧生，100 多名受伤.
2. 1996/06/04，Arianne 5 号火箭由于机载计算机系统浮点数运算发生上溢 (overflow)，致使发射 40 秒后启动自毁程序爆炸，损失 $500 多万.
3. 1982 年，温哥华证券交易所推出一个股票指数，初值设为 1000. 在成分股票并无普遍下跌的 22 个月后，该指数跌到 520，而正确的指数应该为 1098.892，后来发现是由于计算机系统在计算时只保留三位小数，造成舍入误差积累.
4. 1987 年，英国政府发现其使用的软件由于舍入误差积累对过去 21 个月的通胀率低估了 0.1%，这使得与通胀率挂钩的养老金测算系统计算有误，需要给 9 百多万客户做补偿，总金额超过 1 亿英镑.
5. 1991 年，挪威海上油气平台 Sleipner 在建设的最后阶段沉没. 原因是由于在使用软件 NASTRAN 计算时的舍入误差积累导致平台的压力被低估了 47%，损失达 7 亿美元.

参考文献

1. Alfio Quarteroni • Fausto Saleri • Paola Gervasio,
   Scientific Computing with MATLAB and Octave(4th Edition),
   Springer-Verlag Berlin Heidelberg, 2014.
2. Ward Cheney • David Kincaid,
   Numerical Mathematics and Computing (6th Edition),
   Thomson Brooks/Cole, 2008.
3. Michael Heath,
   Scientific Computing: An Introductory Survey (Revised 2nd Edition),
   SIAM, 2018.
4. Alfio Quarteroni • Riccardo Sacco • Fausto Saleri,
   Numerical Mathematics (2nd Edition),
   Springer, 2007.
5. Walter Gander • Martin J. Gander • Felix Kwok,
   Scientific Computing: an introduction using Maple and MATLAB,
   Springer, 2014.