计算物理学作业 1

朱寅杰 1600017721

2017 年 10 月 11 日

1 数值误差的避免

1.1

我们用 $\sum$ 来指代通常数学的求和，用 $\oplus$ 来指代在机器中会产生舍入误差的求和。在计算平均值时，每一步的求和中都会产生一定的舍入误差：

$$\bigoplus_{i=1}^{N} x_i = (...((x_1 + x_2)(1+\epsilon_2) + x_3)(1+\epsilon_3) + ...) + x_N)(1+\epsilon_N) \tag{1}$$

其中 $\epsilon_i (i = 2, 3, ..., N)$ 为每一步求和中的相对舍入误差，它的绝对值小于 $\epsilon_M/2$。由于各个 ϵ_i 均为小量，我们对 (1) 式只保留其一阶量：

$$\bigoplus_{i=1}^{N} x_i \doteq \sum_{i=1}^{N} x_i + (x_1 + x_2)\epsilon_2 + (x_1 + x_2 + x_3)\epsilon_3 + ... + (x_1 + x_2 + ... + x_N)\epsilon_N \tag{2}$$

要估计最大可能的误差，我们首先把这个各个 ϵ_i 放大到上限 $\epsilon_M/2$，然后再把每一个 x_i 的部分和求和放大成对所有 x_i 的求和，这样放大的结果是

$$|\bigoplus_{i=1}^{N} x_i - \sum_{i=1}^{N} x_i| \leq (N-1)(x_1 + x_2 + ... + x_N)\epsilon_M/2 \tag{3}$$

于是我们估计出了 $\sum_{i=1}^{N} x_i$ 的相对舍入误差的上限为 $(N-1)\epsilon_M/2$。要求平均值还需要再除以一个常数 N，最多再产生一个 $\epsilon_M/2$ 的相对舍入误差。于是最后总的相对舍入误差应该不超过 $N\epsilon_M/2$。当然这个估计是一个对最坏情况上限的估计，实际计算中产生的误差一般远比这个数值小。

1.2

所给的两个计算样本方差的公式为

$$S^2 = \frac{1}{N-1}(\sum_{i=1}^{N} x_i^2 - N\bar{x}^2) \tag{4}$$

与

$$S^2 = \frac{1}{N-1}\sum_{i=1}^{N}(x_i - \bar{x})^2 \tag{5}$$

当 $\bar{x} \to 0$ 时，两种算法显然并没有什么差别。但当 $\bar{x}$ 的绝对值比较大时，两种算法都会产生一定的舍入误差。具体地说，首先在前一种算法中会涉及到 $\sum_{i=1}^{N} x_i^2$ 和 $N\bar{x}^2$ 这两个大数的相减，在后一种算法中会涉及到 $x_i - \bar{x}, i = 1, 2, ..., N$ 的减法。当 $\bar{x}$ 相当大时，这两个减法操作都会造成相当大的有效数字的损失，但经过简单的分析可知这两种操作损失的精度基本相当。（好比 $100001.123456789^2 - 100000.987654321^2$ 与 $(100001.123456789 - 100000.987654321)^2$ 两者损失的精度是差不多的，都丢了五位小数）其次在求和的过程中，由于前者求和时的数字往往大于后者许多，因此在相同的相对舍入误差下会产生较大的绝对误差，因而前者最终得到的结果的精确度不如后者。

1.3

对于积分

$$I_n = \int_0^1 \frac{x^n dx}{x+5} \tag{6}$$

容易有 $I_0 = \int_0^1 \frac{dx}{x+5} = \ln(6/5)$，并且有

$$I_k + 5I_{k-1} = \int_0^1 dx \frac{x^k + 5x^{k-1}}{x+5} = \int_0^1 x^{k-1} dx = \frac{1}{k}, k = 1, 2, 3, ... \tag{7}$$

由此我们获得了一个计算 I_n 的递推式。然而这个递推算法并不稳定。比如当我们在计算 I_0 时有一个误差 ϵ_0，也就是我们算出的是 $I_0' = I_0 + \epsilon_0$，那相应地后面每一项都会产生这样的偏差 $I_n' = I_n + \epsilon_n$。从递推式可以得出偏差 ϵ_n 满足

$$\epsilon_k = -5\epsilon_{k-1}, k = 1, 2, ... \tag{8}$$

这意味着每一项递推里的偏差 ϵ_n 会指数增长，也就是说用这个算法计算 I_n 并不稳定。

2 矩阵的模与条件数

2.1

所给的 A 为上三角矩阵，行列式 $|A|$ 等于所有主对角元的乘积，即 $|A| = 1$。因此 A 是非奇异的。

2.2

所给矩阵 A 是一个上三角矩阵，因此只需将 $Ax=y$ 写成方程组的形式，然后反代用 y 表示 x，得到 $x=A^{-1}y$ 即可写出 A^{-1}。

$$A^{-1}=\begin{bmatrix}1 & 1 & 2 & 4 & ... & 2^{n-3} & 2^{n-2}\\ 0 & 1 & 1 & 2 & ... & 2^{n-4} & 2^{n-3}\\ \vdots & \vdots & \vdots & \vdots & & \vdots & \vdots\\ 0 & 0 & 0 & 0 & ... & 1 & 1\\ 0 & 0 & 0 & 0 & ... & 0 & 1\end{bmatrix} \tag{9}$$

是一个上三角矩阵，主对角元均为 1，其他元素 $a_{ij}=2^{j-i-1}, j>i$ 均为 2 的整数幂。

2.3

对于 n 维矢量的 p 模 $\|x\|_p$，当 $p\to\infty$ 时有 $\|x\|_\infty=\max\limits_{i=1,...,n}|x_i|$，因而有

$$\|A\|_\infty=\sup_{\forall x\neq 0}\frac{\max\limits_{i=1,...,n}|\sum\limits_{j=1}^{n}a_{ij}x_j|}{\max\limits_{i=1,...,n}|x_i|}=\sup_{\forall x\neq 0}\max_{i=1,...,n}|\sum_{j=1}^{n}\frac{a_{ij}x_j}{\max\limits_{k=1,...,n}|x_k|}|=\max_{i=1,...,n}\sum_{j=1}^{n}|a_{ij}| \tag{10}$$

2.4

对于幺正矩阵 U，我们知道对于任意矢量 x 都有

$$\|Ux\|_2^2=(x^\dagger U^\dagger)(Ux)=x^\dagger(U^\dagger U)x=x^\dagger x=\|x\|_2^2 \tag{11}$$

故有 $\|U\|_2=1$，同理 $\|U^\dagger\|_2=1$。

将上一式中的 x 替换成 Ax 即得 $\|UAx\|_2=\|Ax\|_2^2$。因此

$$\|UA\|_2=\sup_{\forall x\neq 0}\frac{\|UAx\|_2}{\|x\|_2}=\sup_{\forall x\neq 0}\frac{\|Ax\|_2}{\|x\|_2}=\|A\|_2 \tag{12}$$

因此用欧式模定义的条件数就满足 $K_2(UA)=\|UA\|_2\|A^{-1}U^\dagger\|_2=K_2(A)$。

2.5

回到题中所给的那个上三角矩阵，我们来计算它的条件数 $K_\infty(A)$。有

$$\|A\|_\infty=\max_{i=1,...,n}|\sum_{j=1}^{n}a_{ij}|=n, n=2,3,... \tag{13}$$

$$\|A^{-1}\|_\infty=\max_{i=1,...,n}|\sum_{j=1}^{n}a_{ij}|=2^{n-1}, n=2,3,... \tag{14}$$

故条件数 $K_\infty(A)=n\times 2^{n-1}$

3 Hilbert 矩阵

3.1

为了确定使积分

$$D=\int_0^1 dx(\sum_{i=1}^n c_i x^{i-1}-f(x))^2 \tag{15}$$

的值最小的系数 c_i，我们有

$$\begin{aligned}0=\frac{\partial D}{\partial c_i}&=\int_0^1 2x^{i-1}dx(\sum_{j=1}^n c_j x^{j-1}-f(x))\\&=\sum_{j=1}^n c_j\int_0^1 x^{i+j-2}dx-\int_0^1 x^{i-1}f(x)dx\\&=\sum_{j=1}^n\frac{c_j}{i+j-1}-\int_0^1 x^{i-1}f(x)dx\end{aligned} \tag{16}$$

即有 $\sum\limits_{j=1}^n (H_n)_{ij}c_j=b_i$，其中 $(H_n)_{ij}=\frac{1}{i+j-1}$，而 $b_i=\int\limits_0^1 x^{i-1}f(x)dx$。

3.2

下面试图证明 H_n 是一个正定矩阵。我们从正定的定义出发，任意取一个 n 维向量 $c=(c_1,c_2,...c_n)^T$，有

$$\begin{aligned}c^THc&=\sum_{1\le i,j\le n}\frac{c_ic_j}{i+j-1}=\sum_{1\le i,j\le n}c_ic_j\int_0^1 x^{i+j-1}dx=\int_0^1 dx\sum_{1\le i,j\le n}c_ic_jx^{i+j-1}\\&=\int_0^1 xdx(c_1+c_2x+c_3x^2+...+c_nx^{n-1})^2\ge 0\end{aligned} \tag{17}$$

等号成立仅当 $c_1=c_2=...=c_n=0$ 时取。由定义知 H_n 是一个正定矩阵，自然地它是一个非奇异矩阵。

3.3

下面来估计 $\det(H_n)$ 的值。由文献所给公式知

$$\ln\det(H_n)=4\sum_{k=1}^n\ln(k-1)!-\sum_{k=1}^{2n}\ln(k-1)!=4\sum_{k=1}^{n-1}(n-k)\ln k-\sum_{k=1}^{2n-1}(2n-k)\ln k \tag{18}$$

使用带有求和功能的计算器计算 $\ln\det(H_n)$ 的值，结果如下：

n	$\ln\det(H_n)$	$\det(H_n)$
2	-2.48491	8.33333×10^{-02}
3	-7.67786	4.62963×10^{-04}
4	-15.61524	1.65344×10^{-07}
5	-26.30945	3.74930×10^{-12}
6	-36.76621	1.07805×10^{-16}
7	-55.98858	4.83580×10^{-25}
8	-74.97843	2.73705×10^{-33}
9	-96.73695	9.72023×10^{-43}
10	-121.26497	2.16418×10^{-53}

可见 H_n 当 n 稍大时就已经非常接近奇异了。

3.4

笔者写了两个 Python 小程序，分别使用高斯消元法和 Cholesky 分解来求解方程组 $H_n x = b$，其中 $b = (1, 1..., 1)^T$。程序文档的文件名分别为 gem.py 和 cholesky.py，调试时使用 64 位 Python 3.60 运行无误。运行时按照提示输入一个正整数 N，程序会分别对系数矩阵为 $H_2, H_3, ...H_N$ 的方程组进行计算，结果会输出到 GEMOutput.txt 和 choleskyOutput.txt 中。

容易证明 $H_n x = b$ 的解 x 的第一个分量满足 $x_1 = (-1)^{n-1}n$，我们将其与计算机求得的结果对比，以作为判断解是否可靠的依据。从运行结果看，由于浮点数 10^{-16} 左右的精度限制，GEM 法在 $n \geq 12$，Cholesky 法在 $n \geq 13$ 开始就完全无法给出有意义的解了（GEM 法偏差超过 20%，而 Cholesky 法在分解时会遇到负数开平方根）。

在题中考察的 $n \leq 10$ 的区域，两种算法都可以说是基本准确。抽取几个知道真值的解的分量（事实上 $n \leq 9$ 的区域里离计算机所解出的数值最接近的整数便是真值）看，两者离真值的偏差在同一个数量级内；硬要比较起来，在 n =5~9 的区域中 GEM 法的偏差大约比 Cholesky 法小两三倍。

造成这并不算大的差别的原因是什么呢？我也不能给出明确的答案。也许对一个很小的数开平方根的操作本身就会丢好多有效数字（$y = x^{0.5}$ 在 x 很小的时候本来斜率就发散，导致开完根之后有效数字会变少），可能因此会比只做除法还能做支点选择的 GEM 法要差一些。当然也有可能是别的和语言本身相关或是其他需要仔细分析算法才能明白的问题。