计算物理作业 3

杨远青 22300190015

2024 年 9 月 24 日

远方来朋，喜；假期俱至，悦。

1 题目 1：高斯消元法的时间复杂度分析

1.1 题目描述

Prove that the time complexity of Gaussian elimination algorithm is $\mathcal{O}(n^3)$.

1.2 证明

Gaussian 消元法，此处特指 *Forward Elimination* & *Backward Substitution* 法，而不是最古老的 Gaussian-Jordan 消元法（用于求逆的某浪漫主义教学算法），在大多数情况下的表现，并不如兼具精确度与效率的 $\boldsymbol{LU}$ 分解法，但一些思想被嵌入后者与适用于更大规模矩阵求解的各类迭代算法中，因此仍有必要对其进行分析。

先考虑 *Forward Elimination* 的时间复杂度，即通过初等行变换将原本的增广矩阵 $(\boldsymbol{A}\,|\,\boldsymbol{b})$

$$\left[\begin{array}{cccc|c} a_{11} & a_{12} & \cdots & a_{1n} & b_1 \\ a_{21} & a_{22} & \cdots & a_{2n} & b_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} & b_n \end{array}\right]$$

上三角化为 $\boldsymbol{U}$。暂不考虑 *Pivot* 步骤可能带来的交换操作，尽管这对于提升数值稳定性非常重要。考虑第 1 列的第 2 至 n 行，每一行需要先计算系数 a_{i1}/a_{11}，再进行 n 次乘法与 n 次减法（各行首元素直接设为 0，不计入乘减法操作，但要考虑最右侧 b 的元素），故第 1 列的消元操作数为 $(n-1)(2n+1)$，递推可知，第 i 步便是对 $(n-i+1)\times(n-i+1)$ 子矩阵的消元，迭代操作数为 $(n-i)(2n-2i+3)$，总操作数为

$$T_F(n)=\sum_{i=1}^{n-1}(2n-2i+3)(n-i)=2\sum_{i=1}^{n-1}(n-i)(n-i)+3\sum_{i=1}^{n-1}(n-i)=\frac{4n^3+3n^2-7n}{6}.$$

再考虑 *Backward Substitution* 的时间复杂度，当我们消元得到一个 $n\times n$ 的上三角矩阵 $\boldsymbol{U}$

$$\left[\begin{array}{cccc|c} a'_{11} & a'_{12} & \cdots & a'_{1n} & b'_1 \\ 0 & a'_{22} & \cdots & a'_{2n} & b'_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & a'_{nn} & b'_n \end{array}\right]$$

之后，需要从最后一行开始，逐行求解

$$x_i = \frac{1}{a'_{ii}} \left(b'_i - \sum_{j=i+1}^{k} a'_{ij} x_j \right).$$

每一行涉及的四则运算（我们非常流氓地忽视除法的独特地位，理论上这需要基于牛顿迭代的现代方法进行特殊处理）为 $(n-i)$ 次乘法与 $(n-i)$ 次减法，再进行 1 次除法，故每行的操作数为 $[2(n-i)+1]$，总操作数为

$$T_B(n) = \sum_{i=1}^{n} [2(n-i)+1] = 2\sum_{i=1}^{n}(n-i) + n = n^2.$$

故 Gaussian 消元法的总操作数为

$$T(n) = T_F(n) + T_B(n) = \frac{4n^3 + 3n^2 - 7n}{6} + n^2 = \frac{4n^3 + 9n^2 - 7n}{6}.$$

其中有除法 $n(n+1)/2$ 次，乘法与减法各 $n(n-1)(2n+5)/6$ 次，故

$$\boxed{T(n) = \mathcal{O}(n^3)}$$

伙计，这听起来一点也不酷，怎么到头来还是和求逆矩阵一样是 $\mathcal{O}(n^3)$？但如果我们将 *Substitution* 的思想嵌入到 $\boldsymbol{LU}$ 分解法[1]，对一些特定情形，譬如三对角矩阵的回代操作可以从 $\mathcal{O}(n^2)$ 优化到 $\mathcal{O}(n)$，且对于不同的待解向量 $\boldsymbol{b}$，我们的圣遗物 $\boldsymbol{L}$ 和 $\boldsymbol{U}$ 可以被重复利用，这听上去还是不错的！

如果想和理论计算机科学家一样，执着于对 $\mathcal{O}(n^3)$ 的优化：Strassen 的构造可以帮你将指数因子优化到 $\mathcal{O}(n^{\log_2 7})$，即 $\omega = \log_2 7 \approx 2.8074$[2]，采用 Coppersmith–Winograd 矩阵乘法可以优化到 $\omega \leq 2.3755$[3]. 但这类小数点后的“用力过度”不是我们的菜，有时候反倒是滥用主定理，即它们所需的天文数字规模 $N \times N$ 的矩阵来临时，我们早该另觅出路，比如考虑使用 Jacobi 等迭代法。

公元二〇二四年九月二十四日，午时三刻，于 *HGX106* 室，惊闻徐夫子欲改弦更张，悲哉！

1 题目 1：$\boldsymbol{LU}$ 分解法的时间复杂度分析

1.1 题目描述

Prove that the time complexity of $\boldsymbol{LU}$ decomposition algorithm is $\mathcal{O}(n^3)$.

1.2 证明

$\boldsymbol{LU}$ 分解法的第一步是将系数矩阵 $\boldsymbol{A}$ 分解为一个下三角矩阵 $\boldsymbol{L}$ 和一个上三角矩阵 $\boldsymbol{U}$：

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix} = \begin{bmatrix} l_{11} & 0 & \cdots & 0 \\ l_{21} & l_{22} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ l_{n1} & l_{n2} & \cdots & l_{nn} \end{bmatrix} \begin{bmatrix} 1 & u_{12} & \cdots & u_{1n} \\ 0 & 1 & \cdots & u_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{bmatrix}.$$

[1] 详见 *Numerical Recipes* §2.4

[2] 有个直观而有趣的讨论，详见 *Numerical Recipes* §2.11

[3] $\omega < 2.404$ 的一种证明，参见 *MIT6.890* §23

这一步常采用 Crout 方法实现，即在每一轮中，我们先计算 $\boldsymbol{L}$ 的第 k 列元素 l_{ik}，

$$l_{ik} = a_{ik} - \sum_{s=1}^{k-1} l_{is}u_{sk}, \quad i = k, k+1, \ldots, n.$$

每一个 l_{ik} 的计算涉及 $k-1$ 次乘法和 $k-1$ 次减法，共有 $(n-k+1)$ 个 l_{ik} 需要计算；再计算 $\boldsymbol{U}$ 的第 k 行元素 u_{kj}，

$$u_{kj} = \frac{1}{l_{kk}}\left(a_{kj} - \sum_{s=1}^{k-1} l_{ks}u_{sj}\right), \quad j = k+1, k+2, \ldots, n.$$

相比 l_{ik} 的计算多了一次除法，共有 $(n-k)$ 个 u_{kj} 需要计算，故第 k 轮的操作数为

$$(n-k+1)\cdot(2k-2) + (n-k)\cdot(2k-1) = -4k^2 + (4n+5)k - 3n - 2.$$

因此，分解步骤的总操作数为

$$T_c(n) = \sum_{k=1}^{n}\left[-4k^2 + (4n+5)k - 3n - 2\right] = -4\cdot\frac{n(n+1)(2n+1)}{6} + (4n+5)\cdot\frac{n(n+1)}{2} - (3n+2)\cdot n = \frac{4n^3 - 3n^2 - n}{6}.$$

再考虑回代步骤的操作数，即用分解得到的 $\boldsymbol{L}$ 和 $\boldsymbol{U}$ 求解方程组 $\boldsymbol{Ax} = \boldsymbol{b}$。首先求解 $\boldsymbol{Ly} = \boldsymbol{b}$，即

$$\begin{bmatrix} l_{11} & 0 & \cdots & 0 \\ l_{21} & l_{22} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ l_{n1} & l_{n2} & \cdots & l_{nn} \end{bmatrix} \cdot \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}.$$

这实质上是从第一行开始的 *Forward Substitution*，即

$$y_i = \frac{1}{l_{ii}}\left(b_i - \sum_{j=1}^{i-1} l_{ij}y_j\right).$$

每一步有 1 次除法，$(i-1)$ 次乘法与 $(i-1)$ 次减法；再求解 $\boldsymbol{Ux} = \boldsymbol{y}$，即

$$\begin{bmatrix} 1 & u_{12} & \cdots & u_{1n} \\ 0 & 1 & \cdots & u_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix},$$

这实质上是从最后一行开始的 *Backward Substitution*，即

$$x_i = \left(y_i - \sum_{j=i+1}^{n} u_{ij}x_j\right).$$

每一步有 $(n-i)$ 次乘法与 $(n-i)$ 次减法，故回代步骤操作数为

$$T_s(n) = \sum_{i=1}^{n}\left[(2i-1) + (2n-2i)\right] = \sum_{i=1}^{n}(2n-1) = n(2n-1) = 2n^2 - n.$$

因此，$\boldsymbol{LU}$ 分解法的总操作数为

$$T(n) = T_c(n) + T_s(n) = \frac{4n^3 - 3n^2 - n}{6} + 2n^2 - n = \frac{4n^3 + 9n^2 - 7n}{6}.$$

其中有除法 $n(n+1)/2$ 次，乘法与减法各 $n(n-1)(2n+5)/6$ 次，故

$$\boxed{T(n) = \mathcal{O}(n^3)}$$

Amazing，居然与 *Gaussian* 消元法的各种操作数都相同！

2 题目 2：结合部分主元应用高斯消元法

2.1 题目描述

Using **partial pivoting** Gaussian elimination to solve the system of equations:

$$\begin{cases} 2x_1 + 3x_2 + 5x_3 = 5 \\ 3x_1 + 4x_2 + 8x_3 = 6 \\ x_1 + 3x_2 + 3x_3 = 5 \end{cases}$$

2.2 程序描述

2.3 伪代码

2.4 结果示例

3 题目 3：变分法求解一维薛定谔方程

3.1 题目描述

Solve the 1D Schrödinger equation with the potential (i) $V(x) = x^2$; (ii) $V(x) = x^4 - x^2$ with the variational approach using a **Gaussian basis** (either fixed widths or fixed centers)

$$\phi_i(x) = (\frac{\nu_i}{\pi})^{1/2} \mathrm{e}^{-\nu_i (x - s_i)^2}.$$

Consider the three lowest energy eigenstates.

3.2 程序描述

3.3 伪代码

3.4 结果示例