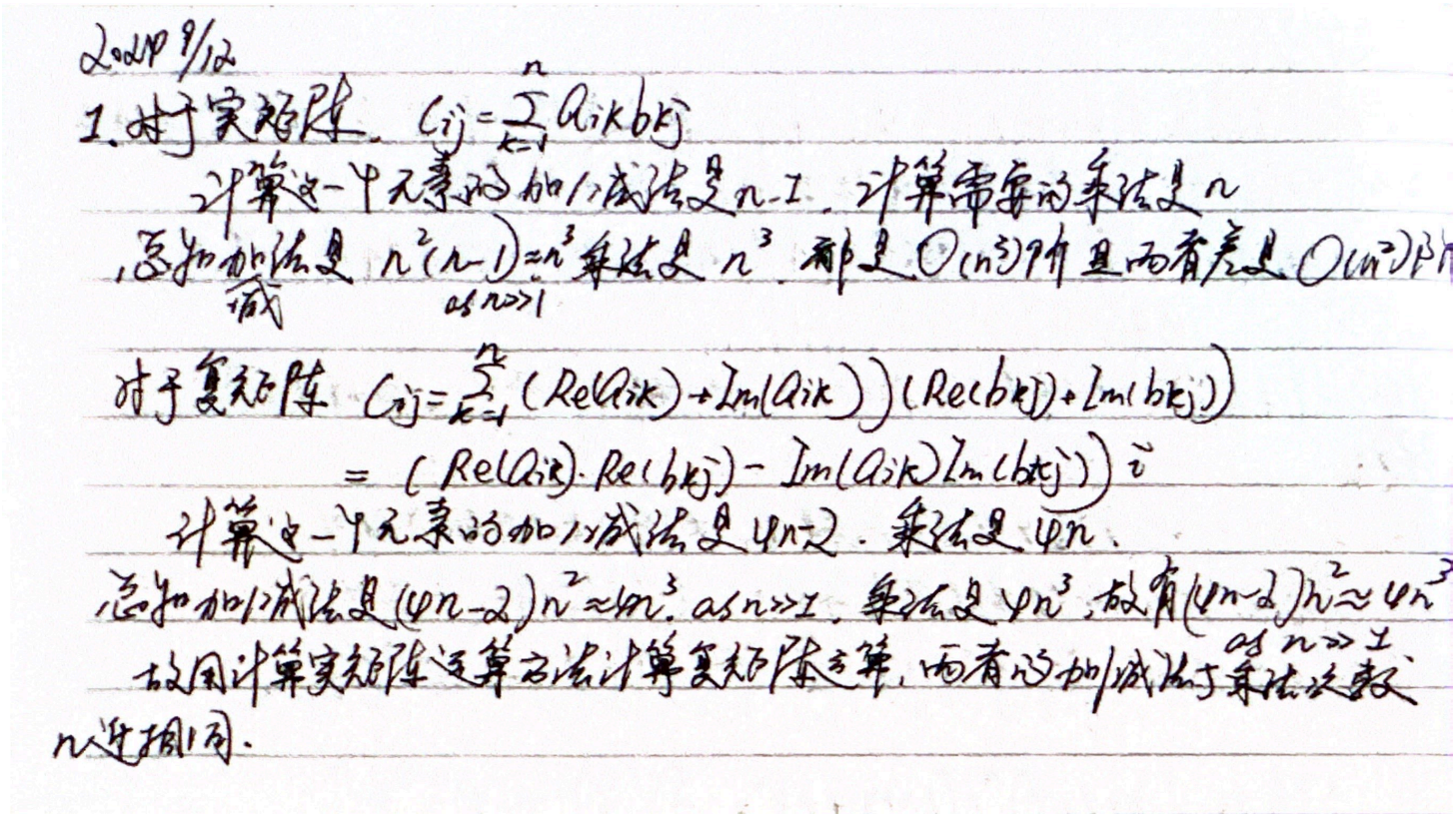
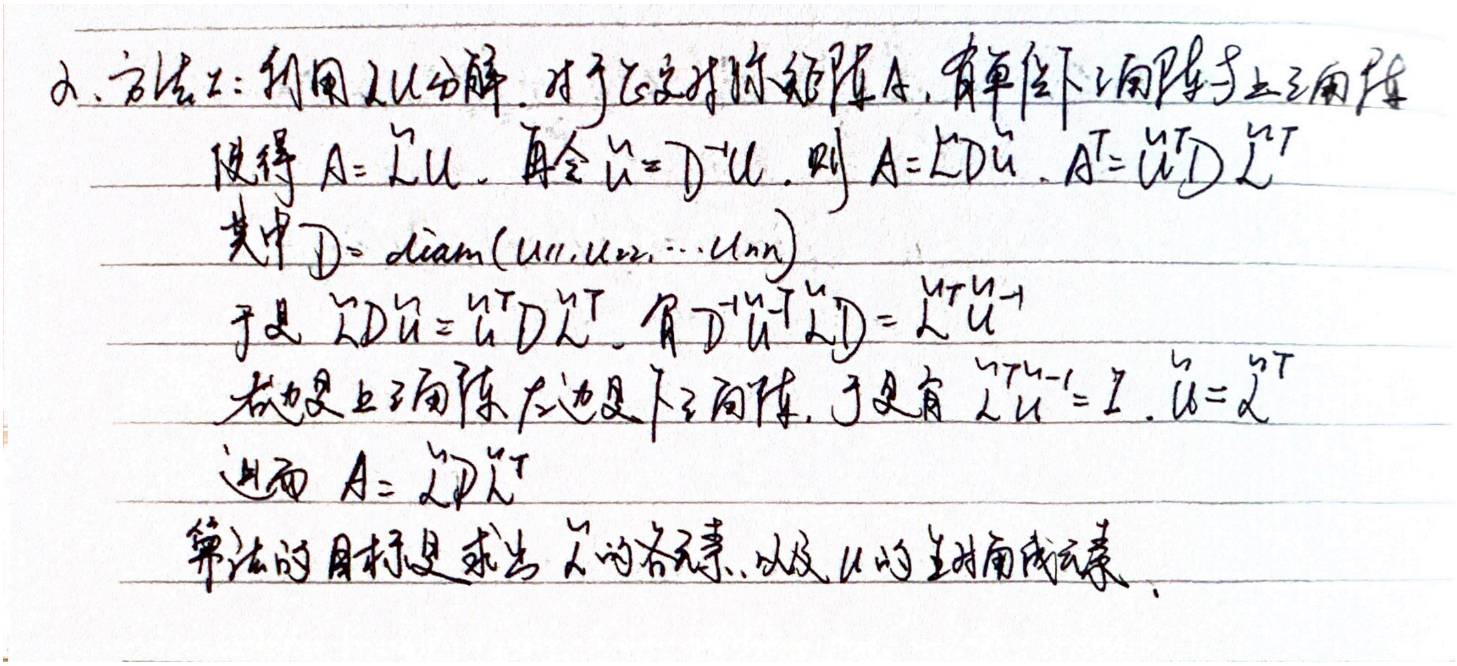
homework_20240910

第一题

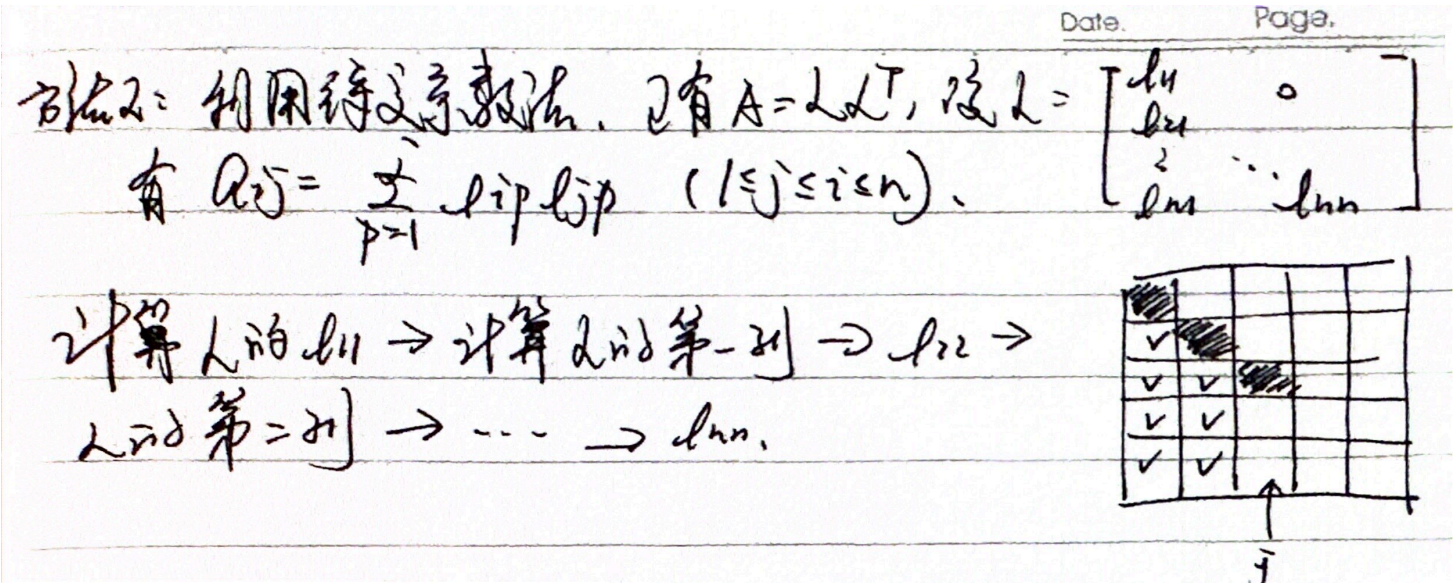


第二题

方法一



方法二



文件 T2.py

```python
import numpy as np
from math import sqrt

# Method One: get Cholesky factorization through LU factorization
def cholesky_1(A):
    n = A.shape[0]
    for i in range(n):
        for j in range(i+1, n):
            factor = A[j, i] / A[i, i]
            A[j, i] = factor
            A[j, i+1:] = A[j, i+1:] - factor * A[i, i+1:]
    diag_elements = np.diag(A)
    diag_matrix = np.diag(np.sqrt(diag_elements))

    L_tilde = np.tril(A, k = -1)
    np.fill_diagonal(L_tilde, 1)
    L = np.dot(L_tilde, diag_matrix)
    return L
```

```
# Method Two: get Cholesky factorization through undetermined coefficient
def cholesky_2(A):
    n = A.shape[0]
    for i in range(n):
        A[i, i] = sqrt(A[i, i])
        A[i+1:, i] = A[i+1:, i] / A[i, i]
        for j in range(i+1 , n):
            A[j:, j] = A[j:, j] - A[j, i] * A[j:, i]
    L = np.tril(A)
    return L

def check_factorization(A, fac_fun):
    L = fac_fun(A)
    A_result = np.dot(L, L.T)
    print(f"{fac_fun.__name__} gets \n{A_result}")
    return A_result


# test
A = np.array([[1,2,3],
              [2,7,6],
              [3,6,10]], dtype = np.float64)

A_copy = np.copy(A)

check_factorization(A, cholesky_1)
check_factorization(A_copy, cholesky_2)
```

第三题

3. 记 $A=[a_{ij}]$ 是主指对角占优阵，经一次高斯消元后为 $\begin{bmatrix} a_{11} & a_1^T \\ 0 & A_2 \end{bmatrix}$ 为方便表示，记 $A_2=[\tilde{a}_{ij}]$。$\tilde{a}_{ij}$ 的角标沿用 A 中角标。

例如 $\tilde{a}_{22}$ 实际上是 A 的 a_{22} 经一次高斯消元的结果，A_2 一行一列的元素是 $\tilde{a}_{2i}$

$$|\tilde{a}_{ii}| = \left|a_{ii} - \frac{a_{i1}}{a_{11}} \cdot a_{1i}\right| \geq |a_{ii}| - \frac{|a_{i1}|}{|a_{11}|} \cdot |a_{1i}| \quad ①$$

$$|\tilde{a}_{ik}| = \left|a_{ik} - \frac{a_{i1}}{a_{11}} \cdot a_{1k}\right| \leq |a_{ik}| + \frac{|a_{i1}|}{|a_{11}|} \cdot |a_{1k}| \quad (2 \leq k \leq n,\ k \neq i) \quad ②$$

$$\text{则 } ① - \sum_{\substack{k=2 \\ k\neq i}}^{n} ②: \left(|a_{ii}| - \sum_{\substack{k=2 \\ k\neq i}}^{n} |a_{ik}|\right) - \frac{|a_{i1}|}{|a_{11}|}\left(\sum_{\substack{k=2 \\ k\neq i}}^{n} |a_{1k}| + |a_{1i}|\right)$$

$$> |a_{ii}| - \frac{|a_{i1}|}{|a_{11}|}\left(\sum_{\substack{k=2 \\ k\neq i}}^{n} |a_{1k}| + |a_{1i}|\right)$$

$$\text{由于 } |a_{11}| > \sum_{\substack{k=2 \\ k\neq i}}^{n} |a_{1k}| + |a_{1i}| - \sum_{\substack{k=1 \\ k\neq i}}^{n} |a_{ik}|$$

$$\text{可有 } ① - \sum_{\substack{k=2 \\ k\neq i}}^{n} ② > 0 \quad \text{于是 } |\tilde{a}_{ii}| > |\tilde{a}_{ik}| \ (i=2,\dots,n,\ k\neq i) \text{ 得证。}$$

第四题

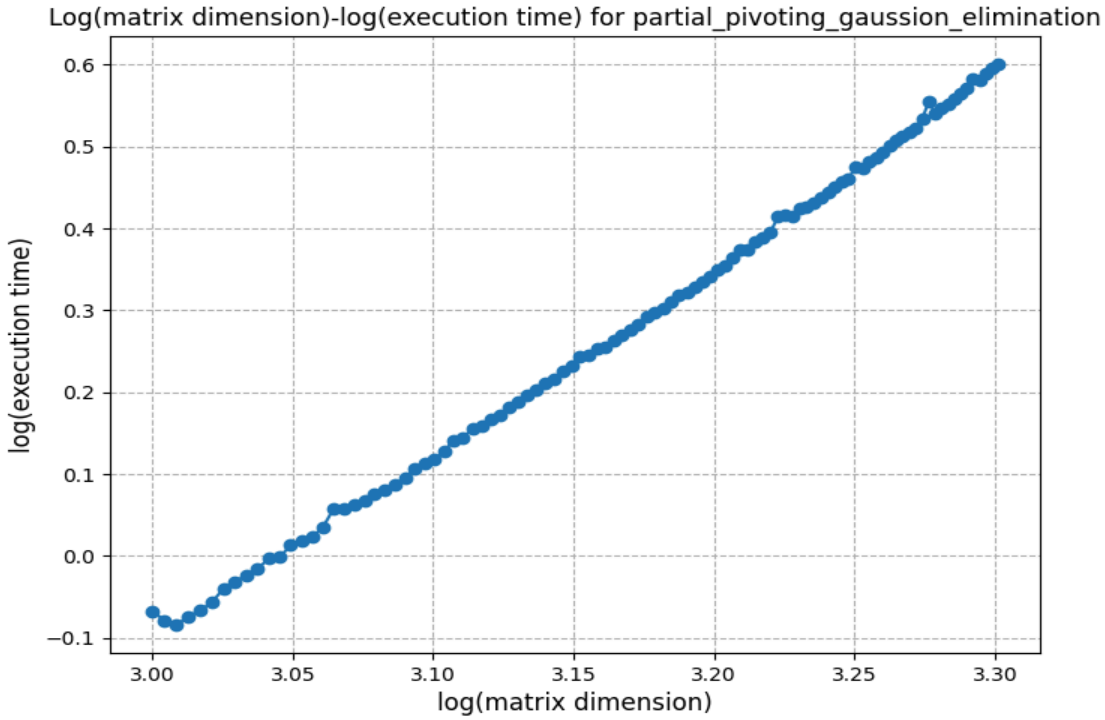
文件 T4.py

```
import numpy as np
import time
import matplotlib.pyplot as plt


def partial_pivoting_gaussion_elimination(A):
    copy_A = np.copy(A)
    n = copy_A.shape[0]

    for i in range(n):
        max_row_i =  np.argmax(np.abs(A[i:, i])) + i
        if max_row_i != i:
            A[[i, max_row_i], :] = A[[max_row_i, i], :]

        for j in range(i+1, n):
            factor = A[j, i] / A[i, i]
            A[j, i:] = A[j, i:] - factor * A[i, i]

    return copy_A


def complete_pivoting_gaussion_elimination(A):
    copy_A = np.copy(A)
    n = copy_A.shape[0]

    for i in range(n):

        max_row_i = np.argmax(np.abs(A[i:, i])) + i
        if max_row_i != i:
            A[[i, max_row_i], :] = A[[max_row_i, i], :]

        max_col_i = np.argmax(np.abs(A[i, i:])) + i
        if max_col_i != i:
            A[:, [max_col_i, i]] = A[:, [i, max_col_i]]

        for j in range(i+1, n):
            factor = A[j, i] / A[i, i]
            A[j, i:] = A[j, i:] - factor * A[i, i:]
```

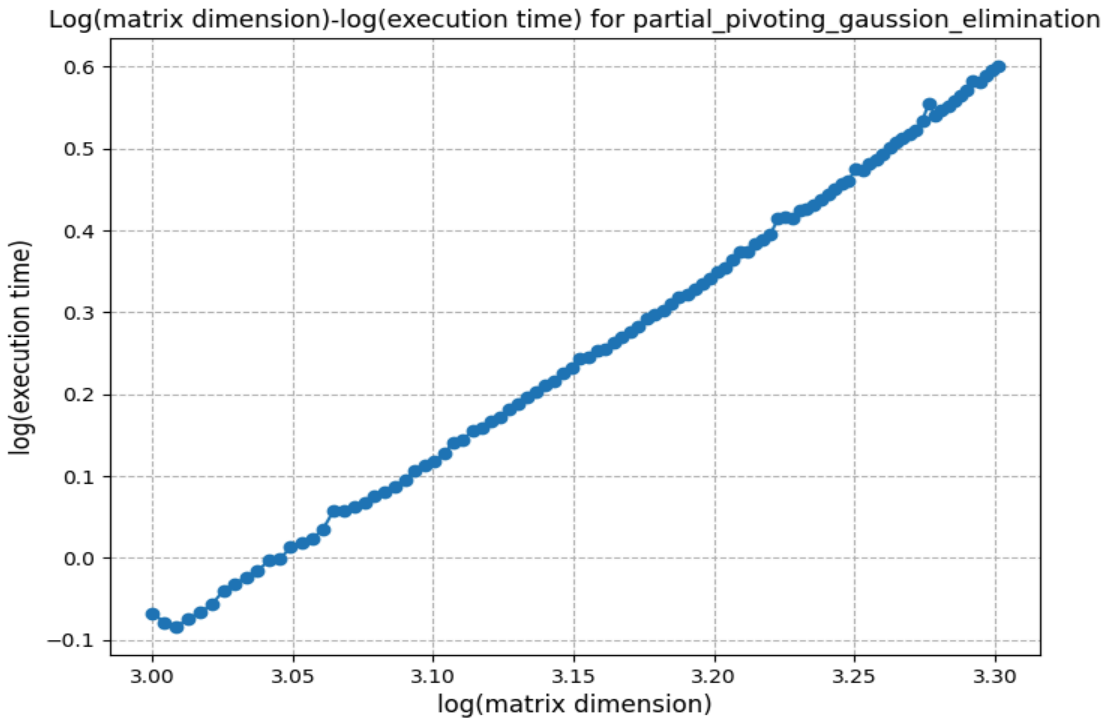
```
    return copy_A


def generate_test_matrix(n):
    A = np.random.randn(n,n)
    return A


def measure_execution_time(n, strategy):
    A = generate_test_matrix(n)
    start_time = time.time()
    strategy(A)
    end_time = time.time()
    return end_time - start_time


def plot_log_log(strategy):
    dimention = np.arange(1000, 2001, 10)
    times = []

    for n in dimention:
        exec_time = measure_execution_time(n, strategy)
        times.append(exec_time)

    log_dimentions = np.log10(dimention)
    log_times = np.log10(times)

    plt.figure(figsize=(8, 6))
    plt.plot(log_dimentions, log_times, marker='o', linestyle='-')
    plt.xlabel('log(matrix dimension)', fontsize=12)
    plt.ylabel('log(execution time)', fontsize=12)
    plt.title(f'Log(matrix dimension)-log(execution time) for {strategy.__name__}')
    plt.grid(True, which='both', ls='--')
    plt.show(block=False)


plot_log_log(partial_pivoting_gaussion_elimination)
plot_log_log(complete_pivoting_gaussion_elimination)
plt.show()
```

绘图结果

部分主元高斯消元


Log(matrix dimension)-log(execution time) for partial_pivoting_gaussion_elimination
log(execution time)
log(matrix dimension)
0.6
0.5
0.4
0.3
0.2
0.1
0.0
−0.1
3.00
3.05
3.10
3.15
3.20
3.25
3.30

全主元高斯消元


Log(matrix dimension)-log(execution time) for partial_pivoting_gaussion_elimination
log(execution time)
log(matrix dimension)
0.6
0.5
0.4
0.3
0.2
0.1
0.0
−0.1
3.00
3.05
3.10
3.15
3.20
3.25
3.30

第五题

Date. Page.

5. 记 $[y_k]=Ax$, $A=[\alpha_1,\alpha_2,\cdots,\alpha_n]$, $x=[x_1,x_2,\cdots,x_n]^T$

$\alpha_i=[a_{i1},a_{i2},\cdots,a_{im}]^T$

则有 $y_k=\sum_{i=1}^{n} x_i a_{ik}$, $k=1,\cdots,m$

$$fl(y_k)=\{[x_1a_{1k}(1+\gamma_{1k})+x_2a_{2k}(1+\gamma_{2k})](1+\delta_{1k})+x_3a_{3k}(1+\gamma_{3k})\}(1+\delta_{2k})+\cdots$$

$$=\sum_{i=1}^{n} x_i a_{ik}(1+\gamma_{ik})\prod_{j=i-1}^{n-1}(1+\delta_{jk}) \quad \delta_{0k}=0.$$

$$=:\sum_{i=1}^{n} x_i a_{ik}(1+\varepsilon_{ik}) \quad \text{其中 } 1+\varepsilon_{ik}=(1+\gamma_{ik})\prod_{j=i-1}^{n-1}(1+\delta_{jk}).$$

目标是估计 ε_{ik} 大小，记 u 是机器精度满足 $fl(x)=x(1+\delta)$, $|\delta|\le u$.

由于 $|\gamma_{ik}|,|\delta_{jk}|\le u$，故 对 $i=1$, $(1-u)^n\le 1+\varepsilon_{ik}\le(1+u)^n$ ①

对 $i=2,\cdots,n$ $(1-u)^{n-i+2}\le 1+\varepsilon_{ik}\le(1+u)^{n-i+2}$ ②.

对 $fl(y_k)=\sum_{i=1}^{n} x_i a_{ik}(1+\varepsilon_{ik})$，$a_{ik}$ 是 A 的作用，$a_{ik}\varepsilon_{ik}$ 是 E 的作用，

记 $E=[\beta_1,\beta_2\cdots,\beta_n]$, $\beta_i=[b_{i1},b_{i2},\cdots b_{im}]$

故 $a_{ik}\varepsilon_{ik}=b_{ik}$ 是 E 中第 k 行第 i 列的元素，由①②获得以下估计

当 $a_{ik}=0$ 时，b_{ik} 满足：对 $i=1$, $[(1-u)^n-1]a_{ik}\le b_{ik}\le[(1+u)^n-1]a_{ik}$

对 $i\ge 2$, $[(1-u)^{n-i+2}-1]a_{ik}\le b_{ik}\le[(1+u)^{n-i+2}-1]a_{ik}$

当 $a_{ik}\le 0$ 时，b_{ik} 满足：对 $i=1$, $[(1+u)^n-1]a_{ik}\le b_{ik}\le[(1-u)^n-1]a_{ik}$

对 $i\ge 2$, $[(1+u)^{n-i+2}-1]a_{ik}\le b_{ik}\le[(1-u)^{n-i+2}-1]a_{ik}$

浮点数系统中：$F=\{0\}\cup\{f: f=\pm 1.d_1d_2\cdots d_t\times 2^J,\ 0\le d_i<2,\ L\le J\le U\}\cup\{\pm\infty\}$

假定 $x>0$，$\exists\alpha$ 满足 $2^\alpha\le x<2^{\alpha+1}$，此时有：

$|fl(x)-x|\le\frac{1}{2}2^{\alpha-t}=\frac{1}{2}2^\alpha\cdot 2^{-t}\le\frac{1}{2}x\cdot 2^{-t}$，$\frac{|fl(x)-x|}{x}\le 2^{-(t+1)}$

此时可以记 $u=2^{-(t+1)}$ 只与浮点数的底数位数有关.