线性方程的直接解法—CHOLESKY分解

冯卓尔 计86

实验要求

实验题目

编程序生成Hilbert矩阵H_n，以及n维向量$b = H_n x$，其中x为分量全为1的向量。用Cholesky分解算法求解法方程$H_n x = b$，得到近似解$\hat{x}$，计算残差$r = b - H_n \hat{x}$和误差$\Delta x = \hat{x} - x$的$\infty-$范数。

（1）设n=10，计算$||r||_\infty$、$||\Delta x||_\infty$。

（2）在右端项上施加10^{-7}的扰动后解方程组，观察残差和误差的变化情况。

（3）改变n的只为8和12，求解相应的方程，观察$||r||_\infty$、$||\Delta x||_\infty$的变化情况。通过这个实验说明了什么问题。

算法描述

先通过逐行逐列枚举的方法生成Hilbert矩阵$\mathbb{H}_n$。然后根据课本算法3.10对H进行分解得到

$$
\begin{gathered}
H_n = L^T L \\
H_n x = b \\
L^T L x = b \\
L^T y = b \\
L x = y
\end{gathered}
$$

为了简便起见，理论上此处应该使用高斯消元法或者上三角形方程组的回代过程解方程，我调用了numpy的linalg库中的solve函数进行$L^T y = b$、$Lx = y$的求解。

实验数据

输入$n=10$，无扰动时，$||r||_\infty=0$，$||\Delta x||_\infty$ =0.0007446391618334269。

输入$n=8$，无扰动时，$||r||_\infty=2.220446049250313\times10^{-16}$，$||\Delta x||_\infty=1.247862366771102\times10^{-7}$。

输入$n=12$，无扰动时，$||r||_\infty=4.440892098500626\times10^{-16}$，$||\Delta x||_\infty=0.4705651714433937$。

输入$n=8$，有扰动时，$||r||_\infty=2.220446049250313\times10^{-16}$，$||\Delta x||_\infty=0.01681647864482061$。

输入$n=10$，有扰动时，$||r||_\infty=1.1102230246251565\times10^{-16}$，$||\Delta x||_\infty=0.6721108022125473$。

输入$n=12$，有扰动时，$||r||_\infty=4.440892098500626\times10^{-16}$，$||\Delta x||_\infty=24.03847297381097$。

实验结论

1. 实验所用的Hilbert矩阵是一个高度病态的矩阵，随着n的增大，实验中方程解x的误差指数上升。n=12时，x的无穷范数过大，可以认为这个算法已经发散而失效
2. 计算残差$r=b-H_n\hat{x}$受n的大小变化不大，甚至可以认为基本上不受输入波动与阶数的影响，因为计算残差与H_n的病态性无直接关联。
3. 这个实验问题对于输入是高度敏感的，验证了矩阵敏感性对算法准确度的影响。

代码

```
import numpy as np
import numpy.matlib as M
import numpy.matrixlib as m
import numpy as np
import matplotlib.pyplot as plt

def devide():
    print("----------------------------------------")

n = int(input())
H = np.fromfunction(lambda i, j: 1.0 / (i + j + 1), (n, n))
x = np.ones(n)
raodong = np.ones(n)
raodong = raodong * 1e-7
```

```
b = np.dot(H, x)
b = b + raodong
temp = H.copy()
devide()
print(H)
devide()
print(x)
devide()
print(b)
L1 = np.linalg.cholesky(H)

def chol():
    for j in range(n):
        for k in range(j):
            temp[j, j] = temp[j, j] - temp[j, k] * temp[j, k]
        temp[j, j] = np.math.sqrt(temp[j, j])
        for i in range(j + 1, n):
            for k in range(j):
                temp[i, j] = temp[i, j] - temp[i, k]*temp[j, k]
            temp[i, j] = temp[i, j] / temp[j, j]
    for i in range(n):
        for j in range(i + 1, n):
            temp[i, j] = 0

def solve():
    temp1 = temp.copy()
    y = np.linalg.solve(temp1, b)
    temp2 = np.transpose(temp)
    x = np.linalg.solve(temp2, y)
    return x

if __name__ == "__main__":
    chol()
    devide()
    print("temp - L1 = ",temp - L1)
    x_chol = solve()
    print("x_chol = ", x_chol)
    r = b - np.dot(H, x_chol)
    d_x = x - x_chol
    delta_x = 0
    delta_r = 0
    for i in range(n):
        if delta_x < d_x[i]:
            delta_x = d_x[i]
        if delta_r < r[i]:
            delta_r = r[i]
    print("delta_x = ",delta_x)
    print("delta_r = ", delta_r)
```