第三章：线性方程组的直接解法

计 93 王哲凡 2019011200

上机题 6

题目描述

> 6. 编程生成 Hilbert 矩阵 $\mathbf{H}_n$（见例 3. 4），以及 n 维向量 $\mathbf{b}=\mathbf{H}_n\mathbf{x}$，其中 $\mathbf{x}$ 为所有分量都是 1 的向量。
>
>   用 Cholesky 分解算法求解方程 $\mathbf{H}_n\mathbf{x}=\mathbf{b}$，得到近似解 $\hat{\mathbf{x}}$，计算残差 $\mathbf{r}=\mathbf{b}-\mathbf{H}_n\hat{\mathbf{x}}$ 和误差 $\Delta\mathbf{x}=\hat{\mathbf{x}}-\mathbf{x}$ 的 ∞-范数。
>
>   1. 设 $n=10$，计算 $\|\mathbf{r}\|_\infty,\|\Delta\mathbf{x}\|_\infty$。
>   2. 在右端项施加 10^{-7} 的扰动然后解方程组，观察残差和误差的变化情况。
>   3. 改变 n 的值为 8 和 12，求解相应的方程，观察 $\|\mathbf{r}\|_\infty,\|\Delta\mathbf{x}\|_\infty$ 的变换情况。通过这个实验说明了什么问题?

实验过程

基本实现

我首先编写函数 `Hilbert()` 生成了 $\mathbf{H}_n$ 和对应的 $\mathbf{b}$:

```python
def Hilbert(n, noise=False):
    H: np.ndarray = np.fromfunction(lambda i, j: 1 / (i + j + 1), (n, n))
    x = np.ones(n)
    b: np.ndarray = np.dot(H, x)
    if noise:
        b += np.random.normal(0, 1e-7, n)
    return H, b
```

其中使用 `noise` 控制是否对于 $\mathbf{b}$ 加入高斯扰动。

然后实现 Cholesky 分解算法（防止后续使用到原矩阵，因此 $\mathbf{L}$ 是新开的矩阵）:

```python
def Cholesky(M: np.ndarray):
    n, n = M.shape
    L = np.zeros_like(M)

    for j in range(n):
        L[j][j] = M[j][j]
        for k in range(j):
            L[j][j] -= L[j][k] ** 2
        L[j][j] = np.sqrt(L[j][j])
```

```python
        for i in range(j + 1, n):
            L[i][j] = M[i][j]
            for k in range(j):
                L[i][j] -= L[i][k] * L[j][k]
            L[i][j] /= L[j][j]

    return L
```

之后编写 solve() 函数，通过分解后的 $\mathbf{H}_n = \mathbf{L}\mathbf{L}^T$ 求解 $\mathbf{L}\mathbf{y} = \mathbf{b}$ 和 $\mathbf{L}^T\mathbf{x} = \mathbf{y}$，并计算对应的残差和误差：

```python
def solve(n, noise=False):
    H, b = Hilbert(n, noise)
    cond = np.linalg.cond(H, p=np.inf)
    print(f'Cond(H) = {cond}')
    L = Cholesky(H)
    y = np.zeros(n)

    for i in range(n):
        y[i] = b[i]
        for j in range(i):
            y[i] -= L[i][j] * y[j]
        y[i] /= L[i][i]

    x = np.zeros(n)

    for i in reversed(range(n)):
        x[i] = y[i]
        for j in range(i + 1, n):
            x[i] -= L[j][i] * x[j]
        x[i] /= L[i][i]

    x_ori = np.ones(n)
    r = np.max(np.abs(b - np.dot(H, x)))
    delta = np.max(np.abs(x_ori - x))
    print(f'r = {r:.20f}, delta = {delta:.20f}')
    return x
```

work() 函数负责整个几个模块。

第 1 问

运行 work(10) 得到结果：

```
Without noise n = 10
Cond(H) = 35352948668981.49
r = 0.00000000000000011102, delta = 0.00059323244471332082
Final x = [1.         0.99999985 1.00000311 0.99997193 1.00013296 0.99963662
 1.00059323 0.99942917 1.00029855 0.99993456]
```

可以看到残差和误差都较小，其中残差明显数量级更低。

第 2 问

同 1 运行观察剩余结果：

```
1 With noise n = 10
2 Cond(H) = 35352948668981.49
3 r = 0.00000000000252953214, delta = 269275.69607346970587968826
4 Final x = [ 2.31478454e-01  6.70422370e+01 -1.39836506e+03  1.26624650e+04
5  -6.01352032e+04  1.64686052e+05 -2.69274696e+05  2.59425818e+05
6  -1.35817331e+05  2.97944816e+04]
```

可以看到，右端项的扰动，会导致误差的急剧放大，而残差依然维持在一个可以接受的数量级。

这表明，经过扰动后的解，在求解方程意义下确实可认为是正确的，但与期望的结果（原解）会有很大的偏差。

这是因为求解的矩阵本身条件数极大（可以看到上面第二行的输出），即矩阵是病态的。

第 3 问

对于 $n = 8$ 执行 `work(8)`：

```
Without noise n = 8
Cond(H) = 33872789109.75766
r = 0.00000000000000022204, delta = 0.00000032588079057483
Final x = [1.         1.         1.00000001 0.99999992 1.00000023 0.99999967
 1.00000024 0.99999993]
With noise n = 8
Cond(H) = 33872789109.75766
r = 0.00000000000001421085, delta = 614.19532788550236546143
Final x = [   0.957605      3.26723992   -28.6240938   161.58400253 -432.19217283
  615.19532789 -436.94606123  124.7899347 ]
```

对于 $n = 12$ 执行 `work(12)`：

```
Without noise n = 12
Cond(H) = 3.920200975864383e+16
r = 0.00000000000000022204, delta = 0.47735610905467973364
Final x = [0.99999996 1.00000568 0.99981988 1.00246909 0.98181599 1.08018109
 0.77596833 1.40643453 0.52264389 1.35012343 0.85425125 1.02628691]
With noise n = 12
Cond(H) = 3.920200975864383e+16
r = 0.00000000479539297160, delta = 228301035.22306957840919494629
Final x = [-2.21570946e+01  2.9055671e+03 -9.05905241e+04  1.22590532e+06
 -8.93515104e+06  3.90639921e+07 -1.08369064e+08  1.95406659e+08
 -2.28301034e+08  1.66684571e+08 -6.91076918e+07  1.24195467e+07]
```

可以看到依然有与 2 类似的结果，但当 n 增大时，矩阵的条件数也在显著变大，进而导致矩阵的病态程度不同。

当 n 较小时，加入扰动后残差和误差的变化都更小，当 n 较大时，则两者变化均明显加剧，尤其是残差的变化。

实验结论

通过这次实验，我实践了 Cholesky 分解正定矩阵进而求解线性方程的做法，对于算法有了更深的理解。

而对于 Hilbert 矩阵方程的求解，则让我感受到了病态矩阵存在的求解困难，而且这种困难是随着矩阵规模变大而快速加剧的，这种问题的求解可能难有较好的解决方案。