数值分析实验三

计 63 陈晟祺 2016010981

2019 年 5 月 12 日

0.1 上机题 6

0.1.1 实验概要

本题要求使用 Cholesky 分解方法求解方程，并计算残差的误差的 ∞-范数，并在施加扰动和矩阵维度变化的情况下重复这一过程。

0.2 实验过程

首先导入常用库

In [1]:
```python
import numpy as np
```

生成 Hilbert 矩阵 $\mathbf{H}_n$，并计算 $\mathbf{b} = \mathbf{H}_n\mathbf{1}$：

In [2]:
```python
n = 10
H = np.fromfunction(lambda i, j: 1. / (i + j + 1), (n,n))
ones = np.ones(n)
b = np.dot(H, ones)
```

按照算法 3.10 的描述，对 H 进行 Cholesky 分解：

In [3]:
```python
def chole(M):
    n = np.shape(M)[0]
    L = np.zeros_like(M) # avoid damaging H

    for j in range(n):
        L[j][j] = M[j][j]
        for k in range(0, j):
            L[j][j] -= L[j][k] ** 2
        L[j][j] = np.sqrt(L[j][j])
```

```
            for i in range(j + 1, n):
                L[i][j] = M[i][j]
                for k in range(0, j):
                    L[i][j] -= L[i][k] * L[j][k]
                L[i][j] /= L[j][j]

        return L
```

```
In [4]: L = chole(H)
```

此时 $\mathbf{H_n} = \mathbf{L}\mathbf{L}^\intercal$，即 $\mathbf{L}\mathbf{L}^\intercal\mathbf{x} = \mathbf{b}$。因此先按照算法 3.7 求解 $\mathbf{L}\mathbf{y} = \mathbf{b}$（注意此时对角线元素并非都为 1)，再使用算法 3.2 求解 $\mathbf{L}^\intercal\mathbf{x} = \mathbf{y}$ 即可：

```
In [5]: def solve_L(L, b):
            n = np.shape(b)[0]
            y = np.zeros_like(b)
            for i in range(n):
                y[i] = b[i]
                for j in range(0, i):
                    y[i] -= L[i][j] * y[j]
                y[i] /= L[i][i]

            x = np.zeros_like(b)
            for i in reversed(range(n)):
                x[i] = y[i]
                for j in reversed(range(i + 1, n)):
                    x[i] -= L[j][i] * x[j] # actually use L^T
                x[i] /= L[i][i]

            return x
```

下面计算残差和误差：

```
In [6]: x = solve_L(L, b)
        r = np.max(np.abs(b - np.dot(H, x)))
        delta = np.max(np.abs(ones - x))
        print("r = {:.20f}, delta = {:.20f}".format(r, delta))
```

```
r = 0.00000000000000022204, delta = 0.00044458507134448322
```

当右端项有扰动（正态噪音）时，重复上述过程：

```
In [7]: x = solve_L(L, b + np.random.normal(0, 1e-7, n))
        r = np.max(np.abs(b - np.dot(H, x)))
        delta = np.max(np.abs(ones - x))
        print("r = {:.20f}, delta = {:.20f}".format(r, delta))
```

```
r = 0.00000017734695356708, delta = 54922.46394105027138721198
```

可见如果 b 发生扰动，将产生极大的误差，而残差依旧很小。也就是说，在扰动意义下的解依旧是正确的，但是与原本的解差别很大。这说明关于 H_n 矩阵的方程问题敏感性很大，这种矩阵是病态的。可以通过计算矩阵的条件数来观察到这一结论：

```
In [8]: np.linalg.cond(H, p=np.inf)
```

```
Out[8]: 35353724553756.422
```

下面对于不同的 n 计算得到的解的残差和误差：

```
In [9]: def chole_solve(n):
            H = np.fromfunction(lambda i, j: 1. / (i + j + 1), (n,n))
            cond = np.linalg.cond(H, p=np.inf)
            print('n = {},\tcond = {}'.format(n, cond))
            ones = np.ones(n)
            b = np.dot(H, ones)
            L = chole(H)
            x = solve_L(L, b)
            r = np.max(np.abs(b - np.dot(H, x)))
            delta = np.max(np.abs(ones - x))
            print("Original:\tr = {:.20f}, delta = {:.20f}".format(r, delta))
            x_dist = solve_L(L, b + np.random.normal(0, 1e-7, n))
            r_dist = np.max(np.abs(b - np.dot(H, x_dist)))
            delta_dist = np.max(np.abs(ones - x_dist))
            print("Disturbed:\tr = {:.20f}, delta = {:.20f}".format(r_dist, delta_dist))

        def f_to_str(f):
            return '{:.4f}'.format(f)

        with open('result_n={}.txt'.format(n), 'w') as f:
```

```
            f.write('Original:\t')
            f.write('\t'.join(map(f_to_str, x)) + '\n')
            f.write('Disturbed:\t')
            f.write('\t'.join(map(f_to_str, x_dist)) + '\n')
```

```
In [10]: chole_solve(8)
         chole_solve(10)
         chole_solve(12)
```

```
n = 8,          cond = 33872790819.49471
Original:          r = 0.00000000000000022204, delta = 0.00000041154382102171
Disturbed:          r = 0.00000020867323935470, delta = 126.71291104727328047375
n = 10,          cond = 35353724553756.42
Original:          r = 0.00000000000000022204, delta = 0.00044458507134448322
Disturbed:          r = 0.00000015520206209096, delta = 381008.03898283827584236860
n = 12,          cond = 3.798320122691213e+16
Original:          r = 0.00000000000000044409, delta = 0.33580581043297352828
Disturbed:          r = 0.00000025662089786493, delta = 14155193.01611004024744033813
```

由上面的结果可知，当 n 越大，H_n 的条件数越大，并且解的残差、误差都越大，并且施以同样的扰动时，带来的误差也越大。这与 3.4 给出的结论是相符的。得到的详细解可见目录下的各 txt 文件。

此外，我使用 numpy 的更高精度浮点数（`float128`）重复了上述过程，可以观察到在无扰动的情况下，残差和误差都有所减小，说明的确截断误差在本实验中对计算精度有显著的影响。而在有扰动的情况下，更高的机器精度并不能缓解如此病态的矩阵带来的巨大误差。

0.2.1 实验结论

通过本次实验，我实现了正定矩阵的 Cholesky 分解并使用结果进行方程求解。同时，通过对带扰动的 Hilbert 矩阵的方程求解，我们能体会到它的强敏感性；并且随矩阵维度增加，病态性会变得更强。因此，受限制于浮点运算的误差，此类矩阵方程问题事实上很难得到可接受的解。

本次实验中，由于原有矩阵尚有用处，因此我实现的 Cholesky 分解并非原地的。在矩阵规模较大时，应当原地存储系数，或者使用稀疏矩阵，从而节省空间。