数值分析上机实验

线性方程组的解法

1　问题描述

本次上机实验需要使用直接法（Gauss 消去法、Cholesky 分解法、Tikhonov 正则化方法）和间接法（共轭梯度法，GMRES 法）分别对一类典型的病态矩阵——Hilbert 矩阵进行求解，并将它们作比较。

设 $H_n = [(h_n)_{ij}] \in \mathbb{R}^{n\times n}$ 是 Hilbert 矩阵，也就是：

$$(h_n)_{ij} = \frac{1}{i+j-1}. \tag{1}$$

Hilbert 矩阵有一些基本的性质。首先，Hilbert 矩阵是一个实对称正定矩阵，所以 $\det H_n > 0$ 而且其特征值都是正实数。其次，Hilbert 矩阵是一种非常病态（ill-conditioned）的矩阵，具体而言，其条件数

$$\mathrm{Cond}_2 H_n = \|H_n\|_2 \left\|H_n^{-1}\right\|_2 = \frac{\lambda_{\max}(H_n)}{\lambda_{\min}(H_n)} \tag{2}$$

随着 n 指数增大，如图1所示。

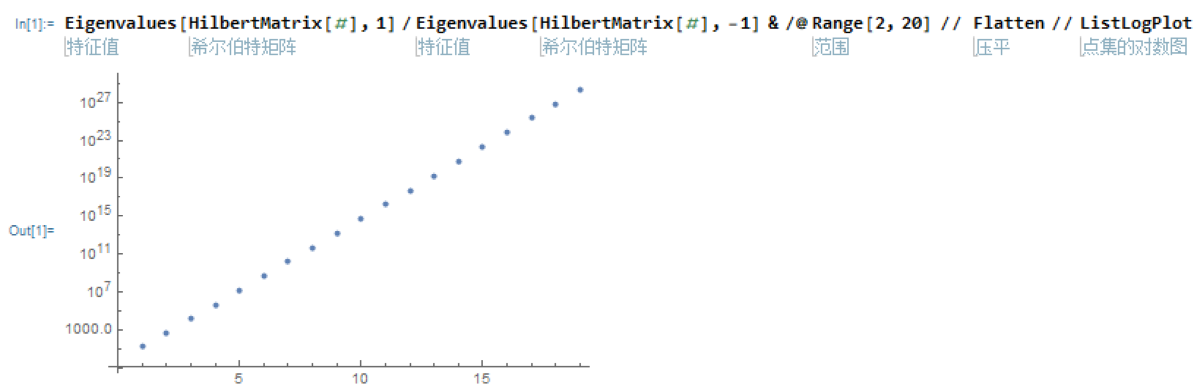

图 1: Hilbert 矩阵的条件数

在本实验中，对 $n = 10, 11, 12, 13, 14, 15$，我们取 $\widetilde{x} = (1, 1, \cdots, 1)^T \in \mathbb{R}^n$，令 $b_n = H_n\widetilde{x}$，再分别用以上方法求解 $H_n x = b_n$，看看误差有多大。

2 实验内容

我们首先用 Gauss 消元法求解。注意到 Hilbert 矩阵的所有顺序主子阵都是较阶的 Hilbert 矩阵，因此其每个顺序主子阵也是实对称正定的，即 $\Delta_i > 0$，$i = 1, \cdots, n-1$。对于这样的矩阵，我们可以做顺序 Gauss 消元法，也就是将矩阵 H_n 做 LU 分解：

$$H_n = LU, \tag{3}$$

其中 L 是单位下三角矩阵，U 是上三角矩阵。我们可以用 `scipy.linalg` 库中的 `lu` 方法直接实现 LU 分解，然后先求解 $L\mathbf{y} = \mathbf{b}$（为了符号上的简洁，我们将 b_n 简记为 $\mathbf{b}$，下同），再求解 $U\mathbf{x} = \mathbf{y}$。写成分量形式则为：

$$\begin{cases} y_1 = b_1, \\ y_i = b_i - \sum_{k=1}^{i-1} l_{ik} y_k, \quad i = 2, 3, \cdots, n \end{cases} \tag{4}$$

以及

$$\begin{cases} x_n = y_n / u_{nn}, \\ x_i = (y_i - \sum_{k=i+1}^{n} u_{ik} x_k) / u_{ii}, \quad i = n-1, n-2, \cdots, 1. \end{cases} \tag{5}$$

这一算法可以用 `numpy.array` 库的相应操作来实现，具体如下：

```
###use gaussian elimination (lu decomposition) to solve the linear system
p,l,u = LA.lu(Hilbert_n)
p_inv_b_n = LA.inv(p) @ b_n

#solve ly = p^{-1} b_n
y = np.array([0.0]*n)
y[0] = p_inv_b_n[0]
for i in range(1,n):
    y[i] = p_inv_b_n[i] - np.dot(l[i],y)

#solve uw = y
w = np.array([0.0]*n)
w[n-1] = y[n-1]/u[n-1,n-1]
for i in range(1,n):
    w[n-1-i] = (y[n-1-i] - np.dot(u[n-1-i],w))/u[n-1-i,n-1-i]
print("Gaussian Elimination:{}".format(w))
```

由于 Hilbert 矩阵实对称正定，我们也可以对该矩阵做 Cholesky 分解，即：

$$H_n = LL^T, \tag{6}$$

其中 L 是下三角矩阵。Cholesky 分解的分量形式也很简洁（称为平方根法），如下：

$$\begin{cases} l_{jj} = (a_{jj} - \sum_{k=1}^{j-1} l_{jk}^2)^{1/2}, \\ l_{ij} = (a_{ij} - \sum_{k=1}^{j-1} l_{ik} l_{jk})/l_{jj}, \quad i = j+1, \cdots, n \end{cases} \tag{7}$$

然后，先后求解 $L\mathbf{y} = \mathbf{b}$ 和 $L^T\mathbf{x} = \mathbf{y}$。写成分量形式为：

$$y_i = (b_i - \sum_{k=1}^{i-1} l_{ik} y_k)/l_{ii}, \quad i = 1, 2, \cdots, n, \tag{8}$$

$$x_i = (y_i - \sum_{k=i+1}^{n} l_{ki} x_k)/l_{ii}, \quad i = n, n-1, \cdots, 1. \tag{9}$$

代码实现为：

```python
###use cholesky decomposition (LL^T decomposition) to solve the linear system (
    Hilbert matrix is real-symmetric and positive-definite)
L = np.zeros((n,n)) + 0.0
for j in range(n):
    L[j,j] = np.sqrt(Hilbert_n[j,j] - np.sum(L[j][:j]**2))
    for i in range(j+1,n):
        L[i,j] = (Hilbert_n[i,j] - np.dot(L[i],L[j]))/L[j,j]

#solve Ly = b
Y = np.array([0.0]*n)
for i in range(n):
    Y[i] = (b_n[i] - np.dot(L[i],Y))/L[i,i]

#solve L^T x = y
W = np.array([0.0]*n)
for i in range(n):
    W[n-1-i] = (Y[n-1-i] - np.dot(np.transpose(L)[n-1-i],W))/L[n-1-i,n-1-i]
print("Cholesky Decomposition:{}".format(W))
```

Tikhonov 正则化方法是一种对病态矩阵使用的近似方法。其基本原理是奇异值分解（SVD）。在本例中，存在酉矩阵 V 和 U，使得 $H_n = VDU^*$，其中 $V = (\mathbf{v}_1, \cdots, \mathbf{v}_n)$，$U = (\mathbf{u}_1, \cdots, \mathbf{u}_n)$，$D = \mathrm{diag}(\sigma_1, \cdots, \sigma_n) > 0$，于是此时的解可以显示地写出：

$$\mathbf{x} = \sum_{j=1}^{n} \frac{1}{\sigma_j} \langle \mathbf{b}, \mathbf{v}_j \rangle \mathbf{u}_j. \tag{10}$$

事实上，从该显示解可以看出，由于 H_n 是病态矩阵，所以其最小特征值非常小。而在 $\mathbf{x}$ 的表达式中，σ_j 在分母上，因此 H_n 或者 $\mathbf{b}$ 的小扰动会引起很大的误差。吉洪诺夫正则化方法则考虑对奇异值 $\sigma_j = \sqrt{\lambda_j(A^*A)}$ 进行“偏移”，从而缓解以上问题，即：

$$A^*A \mapsto \alpha I + A^*A, \tag{11}$$

其中 α 称为正则化参数，此时问题相当于求解

$$(\alpha I + A^* A)\mathbf{x}_\alpha = A^*\mathbf{b} \Leftrightarrow \mathbf{x}_\alpha = \sum_{j=1}^{n} \frac{\mu_j}{\alpha + \mu_j^2} \langle \mathbf{b}, \mathbf{v}_j \rangle \mathbf{u}_j, \tag{12}$$

当 α 充分小时，$\mathbf{x}_\alpha$ 将是 $\mathbf{x}$ 的一个很好的近似。另一方面，为了使奇异值（条件数）真正得到改善，我们要求 α 至少比 $\mu_1\mu_n$ 大至少一个量级。在本例中，$n = 15$ 时最小奇异值在 10^{-18} 量级，因此我们选择 $\alpha = 10^{-13}$ 作为正则化参数。

我们使用 scipy.linalg 库的 svd 方法实现奇异值分解。具体的代码实现为：

```
###use Tikhonov regularization to alleviate this problem

#calculate SVD of Hilbert matrix
unitary_v,sv,unitary_u_h = LA.svd(Hilbert_n)
unitary_u = np.transpose(np.conj(unitary_u_h))

#choose the value of regularization parameter
alpha = 1e-13

#use the formular x_\alpha = \sum_{j=1}^r \frac{\mu_j}{\alpha+\mu_j^2}(b,v_j)u_j to
    solve (\alpha I +A^\ast A) x_\alpha = A^\ast b
x_alpha = np.array([0.0]*n)
for j in range(n):
    x_alpha += (sv[j]/(alpha+sv[j]**2)) * np.dot(b_n,unitary_v[:,j])*unitary_u[:,j]
    #since b_n is real, so we don't need to take conjugate on b_n
print("Tikhonov Regularization with alpha = {}:{}".format(alpha,x_alpha))
```

共轭梯度法和 GMRES 方法是两种典型的适用于实对称正定矩阵的线性方程组迭代解法，可以分别用 scipy.sparse.linalg.cg 和 scipy.sparse.linalg.gmres 来实现。

3 实验结果与讨论

我们对每个 n 输出以上各种方法的计算结果，原始输出如下：

```
=========n=10 output=========
Gaussian Elimination:[1.         0.99999973 1.00000581 0.99994725 1.0002513
    0.99930985
1.00113142 0.99890735 1.0005733  0.99987398]
Cholesky Decomposition:[1.         0.99999979 1.00000435 0.99996074 1.00018631
    0.99949006
1.00083359 0.99919698 1.00042042 0.99990777]
Tikhonov Regularization with alpha = 1e-13:[0.99999895 1.00003359 0.99976634
    1.00051595 0.99984201 0.99954939
0.99990516 1.0003973  1.00042844 0.99956138]
```

```
Conjugate Gradient:[0.99905491 1.01004148 0.98356913 0.99197253 1.00497259 1.01246553
1.01275945 1.00673462 0.99581993 0.98135781]
GMRES:[0.99905712 1.01002942 0.98357323 0.99198118 1.00498007 1.01246976
1.01275994 1.00673155 0.99581368 0.98134884]
=========n=11 output=========
Gaussian Elimination:[1.00000001 0.9999988  1.00003148 0.99964523 1.00212776 0.99247652
1.01646108 0.97746392 1.01878886 0.99127853 1.00172782]
Cholesky Decomposition:[1.         1.0000005  0.99998602 1.00016657 0.99895627 1.00382384
0.99138431 1.01209046 0.98970495 1.00486672 0.99902037]
Tikhonov Regularization with alpha = 1e-13:[0.99999839 1.00004569 0.99972035 1.0005032  1.00001812 0.99957889
0.99968202 1.00012291 1.00047902 1.00036011 0.99948922]
Conjugate Gradient:[0.99871962 1.01270289 0.98173577 0.98845357 1.00232503 1.01209683
1.01526857 1.01225482 1.00423922 0.99244689 0.97792783]
GMRES:[0.99872342 1.01268348 0.98174054 0.98846631 1.0023372  1.01210505
1.01527184 1.01225316 1.00423306 0.99243678 0.97791434]
=========n=12 output=========
Gaussian Elimination:[0.99999995 1.00000625 0.99980314 1.00268238 0.9803419 1.0863264
0.75963619 1.43475914 0.49070736 1.37269185 0.84517164 1.02787383]
Cholesky Decomposition:[0.99999991 1.00001192 0.9996227  1.00516665 0.96197685 1.16757526
0.53195597 1.84888268 0.00320122 1.73099477 0.69574284 1.05486929]
Tikhonov Regularization with alpha = 1e-13:[0.99999843 1.00004074 0.99978005 1.00030313 1.00016239 0.99977866
0.99965303 0.99986204 1.00021238 1.00042902 1.00026026 0.99951766]
Conjugate Gradient:[0.9983419  1.01547889 0.98026301 0.98504319 0.9993116 1.01085576
1.01645098 1.01607344 1.01066316 1.00131947 0.98903473 0.97462709]
GMRES:[0.99834789 1.01544996 0.98026785 0.98506057 0.9993296  1.01086934
1.01645842 1.01607449 1.01065823 1.00130919 0.98901979 0.97460815]
=========n=13 output=========
Gaussian Elimination:[ 1.0000001   0.99998511  1.00055935  0.99087694  1.08055414  0.56914319
2.48542441 -2.40953937  6.26309881 -4.39845851  4.52767107 -0.32898035
1.21966517]
Cholesky Decomposition:[1.00000001 0.99999926 1.00001305 0.9999905  0.99855924 1.01473047
```

```
0.92811585 1.20759823 0.62122357 1.44194842 0.67982367 1.13130516
0.97669256]
Tikhonov Regularization with alpha = 1e-13:[0.99999887 1.00002583 0.99988776 1.00006163 1.00022295 0.99999566
0.99975271 0.9997406  0.99995155 1.00022913 1.00037272 1.00019773
0.99956081]
Conjugate Gradient:[0.99792566 1.01833437 0.97912088 0.98180064 0.99609662 1.00899641
1.0166274  1.01855605 1.0154845  1.00838209 0.99817578 0.98565603
0.97146368]
GMRES:[0.99793452 1.01829367 0.97912497 0.98182299 0.99612144 1.00901666
1.01664042 1.01856121 1.0154821  1.0083728  0.99816039 0.98563533
0.97143841]
=========n=14 output=========
Gaussian Elimination:[ 0.99999979  1.00002968  0.99898174  1.01500502  0.88273988 1.53469178
-0.46182084  3.2476614  -0.16547059 -1.25538075  6.16392178 -3.69339978
3.12931551  0.60372523]
Cholesky Decomposition:[  0.99999998   1.00001108   0.99927741   1.01726472 0.79199032
2.46831515  -5.59195461  20.71454321 -39.19431501  57.09904841
-51.74820055  32.94508998 -10.25306992   2.752     ]
Tikhonov Regularization with alpha = 1e-13:[0.9999994  1.00000976 0.99999186 0.99986201 1.00022604 1.00016224
0.99988533 0.99972024 0.99978283 1.00001161 1.00025903 1.00035937
1.00016538 0.999563  ]
Conjugate Gradient:[0.99747455 1.02124215 0.97827755 0.97875745 0.99279231 1.00670332
1.01603848 1.01998305 1.01900949 1.01395594 1.0056789  0.99493491
0.98235248 0.96843805]
GMRES:[0.99748703 1.02118739 0.97827992 0.97878492 0.9928248  1.00673147
1.01605848 1.01999376 1.01901101 1.01394891 1.00566421 0.99491346
0.98232515 0.96840563]
=========n=15 output=========
Gaussian Elimination:[  0.99999994   1.00001347   0.9993301    1.01376439 0.85155006
1.9486106   -2.83210013  11.07173484 -16.1203244   18.60729986
-6.9830423   -2.65817475   8.21445309  -2.87867698   1.76556238]
Cholesky Decomposition:[   1.00000029    0.99998491    0.99947145    1.02864101 0.53323464
4.97679219  -19.74540975   72.3880854  -167.45645604  278.30849724
-317.35158929  250.96421455 -126.99847801   39.52159387   -4.16858238]
```

```
Tikhonov Regularization with alpha = 1e-13:[0.99999992 0.99999494 1.00008198
    0.99970795 1.00019802 1.00028126
1.00001746 0.99975503 0.9996895  0.99983335 1.00008859 1.00031472
1.00037119 1.00013848 0.99952568]
Conjugate Gradient:[0.99699193 1.02418119 0.97770207 0.97592798 0.98947556
    1.00411194
1.01486688 1.02057197 1.02147958 1.01829748 1.01180867 1.00273126
0.99167669 0.97914703 0.96554649]
GMRES:[0.99700883 1.02411009 0.97770156 0.97596053 0.98951642 1.00414913
1.01489521 1.02058965 1.02148643 1.01829409 1.01179595 1.00271023
0.99164833 0.97911226 0.96550617]
```

为了看每种方法的误差有多大，我们计算 $\varepsilon = \frac{\|x-\widetilde{x}\|_2}{\|x\|_2}$，列表如下：

表 1: 各种方法的相对误差 $\varepsilon = \frac{\|x-\widetilde{x}\|_2}{\|x\|_2}$

	Gauss 消去	Cholesky 分解	Tikhonov 正则化	共轭梯度法	GMRES 法
n =10	0.0006	0.0004	0.0003	0.0109	0.0109
n =11	0.0108	0.0058	0.0003	0.0125	0.0125
n =12	0.2376	0.4648	0.0003	0.0141	0.0141
n =13	2.5589	0.1976	0.0002	0.0156	0.0156
n =14	2.1928	25.5365	0.0002	0.0170	0.0170
n =15	7.5937	139.6530	0.0002	0.0184	0.0184

从表中可以看出，如果不做正则化，则 Gauss 消去法和 Cholesky 分解法只在 $n = 10$ 和 $n = 11$ 时能够得到较好结果，当 $n \geq 12$ 时，结果立刻变得不可信，到了 $n = 15$，误差已经非常大。而做了 Tikhonov 正则化之后，则对于 $n = 10, 11, \cdots, 15$ 都能算出精度非常高的结果（从输出结果也能看出，基本有三位以上的有效数字）。对于共轭梯度法和 GMRES 法，两种方法的表现几乎完全一致，精度也能稳定在 1%-2% 左右（按相对误差 ε 记），从输出结果上看也有至少一位的有效数字。这说明，迭代法比直接法（不做正则化）对病态矩阵更能得到精确的结果。

4 小结

通过本次上机实验，我们体验了 LU 分解、Cholesky 分解和 Tikhonov 正则化方法的 `Python` 代码实现，对这些方法的原理和适用范围有了更深刻的理解。同时，通过对典型的病态矩阵——Hilbert 矩阵的研究，我们对条件数等概念有了更直观的认识，了解到了求解线性方程组的直接和迭代解法之间的区别，也掌握了改善病态矩阵求解问题的方法。