数值线性代数第二次上机作业

吴秉哲 1200010666

2014 年 11 月 19 日

第二章习题

习题 1

估计 5 到 20 阶 Hilbert 矩阵的 ∞ 范数条件数：

本题使用了书中算法 5.12(估计矩阵的 1 范数)。对于 n 阶的 Hilbert 矩阵 H_n, 在估计 $\|H_n^{-1}\|_\infty$ 的时候需要求解方程 $H_n^T\omega = x$ $H_n z = v$, 我使用了选列主元的 Gauss 消去法求解，得到了 $\|H_n^{-1}\|_\infty$ 的估计值。考虑到 H_n 的高度病态性，求解上面两个方程组可能会产生比较大的误差，故需要使用一种方法来检验。Matlab 自带有矩阵求逆函数 inv() 和求范数函数 norm()，可以求出 H_n^{-1} 和 $\|H_n^{-1}\|_\infty$，经测试发现当 n 比较大的时候误差也很大的。但是目前没有 H_n^{-1} 的精确值，无法判断该估计的误差大小，所以我就把用书中算法估计的范数和用 inv() 函数求逆后再求的范数进行了比较。又 $\|H_n\|_\infty$ 的计算比较简单，误差应该相对较小，故我直接计算了 $\kappa_n = \|H_n^{-1}\|_\infty \|H_n\|_\infty$。计算结果如下表所示：

阶数n	$\left\|H_n^{-1}\right\|_\infty$	自带函数计算$\left\|H_n^{-1}\right\|_\infty$	相对误差	$\left\|H_n\right\|_\infty$	κ_n
1	1	1	0	1	1
2	18	18	0	1.5	27
3	408	408	0	1.833333333	748
4	13620	13620	-2.27E-14	2.083333333	28375
5	413280	413280	-2.50E-13	2.283333333	943656
6	11865420	11865420	6.13E-12	2.45	29070279
7	379964971.3	379964970.4	2.35E-09	2.592857143	9.85E+08
8	12463050464	12463049866	4.80E-08	2.717857143	3.39E+10
9	3.88711E+11	3.8871E+11	1.31E-06	2.828968254	1.1E+12
10	1.20704E+13	1.20702E+13	1.17E-05	2.928968254	3.54E+13
11	4.07506E+14	4.07902E+14	-0.00097	3.019877345	1.23E+15
12	1.23E+16	1.22E+16	0.0096111	3.103210678	3.83E+16
13	1.46E+17	1.98E+17	-0.265843	3.180133755	4.63E+17
14	4.22E+18	2.90E+18	0.4530299	3.251562327	1.37E+19
15	3.39E+17	1.17E+17	1.9029652	3.318228993	1.12E+18
16	3.98E+17	4.09E+19	-0.990285	3.380728993	1.34E+18
17	5.73E+17	2.93E+18	-0.804191	3.439552523	1.97E+18
18	2.61E+19	1.53E+18	16.077667	3.495108078	9.13E+19
19	9.57E+18	4.78E+17	19.020088	3.547739657	3.40E+19
20	8.79E+17	1.67E+18	-0.474402	3.597739657	3.16E+18

从表中可以看出，

1. $\|H_n^{-1}\|_\infty$ 随阶数的增长是近乎是指数型的，但比指数慢，而 $\|H_n^{-1}\|$ 基本是一个常数，所以 $\kappa_n \|H_n^{-1}\|_\infty$ 基本一样也是增长得很快，到 13 阶就已经达到了 10^17 数量级，超出了机器精度。从维基百科上查到，$\kappa_n = O(\frac{(1+\sqrt{2})^{4n}}{\sqrt{n}})$，符合计算结果

2. 而对 $\|H_n^{-1}\|_\infty$ 的估计在 n 比较大的时候也明显产生了误差，无论是算法 5.12 还是 Matlab 的自带函数，它们都不单调。我猜测 $\|H_n^{-1}\|_\infty$ 应该随着 n 的增加单调上升，尚未查找到文献验证。

3. Matlab 的函数计算的结果和误差估计的相对误差随着 n 的增加而增加，整体也是比较小的，说明这种方法应该误差不大。在网上未查找到 $\|H_n^{-1}\|_\infty$ 的准确值来比较

习题 2

估计解的误差与真实相对误差比较。本题中 x 的分量服从 $U(0,1)$ 分布，使用列主元 Gauss 消去法求解得到 $\widehat{\chi}$, 并使用 $\frac{\kappa_\infty(A_n)\|r\|_\infty}{\|b\|_\infty}$ 来估计解的相对误差 $\frac{\|x-\widehat{\chi}\|_\infty}{\|x\|_\infty}$. 具体结果如下表：

阶数	估计误差	实际误差	估计误差/实际误差
1	0	0	NaN
2	0	0	NaN
3	0	0	NaN
4	3.38E-16	7.25E-17	4.657351243
5	1.75E-15	4.87E-16	3.587317547
6	2.21E-15	3.52E-16	6.282187429
7	1.37E-14	1.59E-15	8.616975198
8	2.88E-15	1.05E-15	2.7312107
9	2.29E-14	5.20E-15	4.40080627
10	3.61E-15	4.47E-16	8.078334767
11	9.53E-14	2.70E-14	3.522065939
12	1.19E-13	3.03E-14	3.929671715
13	1.55E-14	6.47E-15	2.396757062
14	2.30E-13	7.24E-14	3.175972916
15	1.82E-13	4.93E-14	3.681405757
16	2.26E-12	4.49E-13	5.023882908
17	3.45E-12	6.66E-13	5.182961942
18	1.53E-12	5.76E-13	2.65850936
19	1.38E-11	2.65E-12	5.202948803
20	2.13E-11	3.97E-12	5.368788445
21	6.35E-11	1.58E-11	4.014230806
22	1.88E-10	8.05E-11	2.338742879
23	2.22E-10	3.68E-11	6.023956928
24	9.35E-10	9.28E-11	10.0722088
25	4.68E-10	1.51E-10	3.094464995
26	1.34E-10	4.97E-11	2.704822986
27	1.86E-09	3.91E-10	4.751784555
28	1.41E-09	2.54E-10	5.576213724
29	1.55E-08	4.88E-09	3.186594838
30	9.44E-09	4.30E-09	2.197146379

从表中可见，

1. 估计的误差和实际的误差随阶数的增加而总体增加，并且总体来说误差非常小
2. 估计误差总是比实际误差大，而且理论推导证明这个估计误差应该大于实际误差，这个结果说明得到的结果没有和理论推导矛盾
3. 估计误差是实际误差的 10 倍以内，大部分在 5 倍以内，并没有随着 n 的增大而增大。说明这个估计是比较好的。

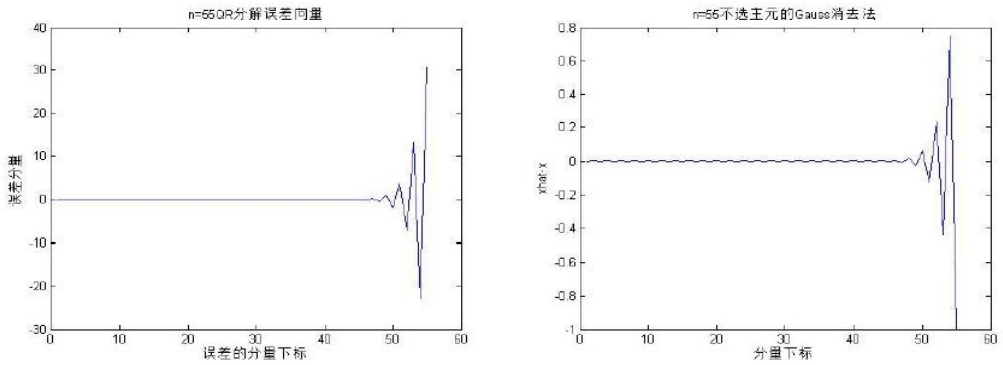
第三章习题

习题 1

用 QR 分解求解第一章的三个方程。

- 方程 1

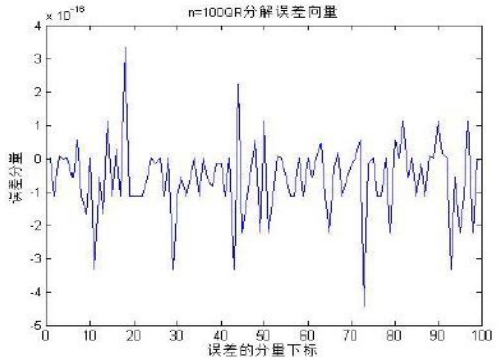
  n=55；计算解的误差分量如下图：

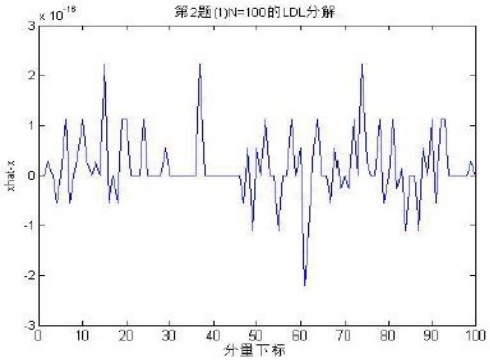


  1. 另人惊奇的是，使用 Householder 变换的 QR 分解最多只能解到 n=55 的情况，阶数再大就会出现 inf 和 NaN。查看运算中间结果发现，$\|A(j:n,j)\| \to 0$, 随着 $j \to n$, 导致解上三角方程时出现问题。
  2. 查看中间结果还发现，$\beta \to 1, v \to (1,-1,0,...,0)'$。
  3. 和不选主元的 Gauss 消去法相比误差大约是其 50 倍，和选主元的更没法比较了，有非常大的误差。但两者误差分量变换模式相似，都是前面的分量误差小，之后越来越大，正负跳跃。

- 方程 2

  随机选择结果如下图：



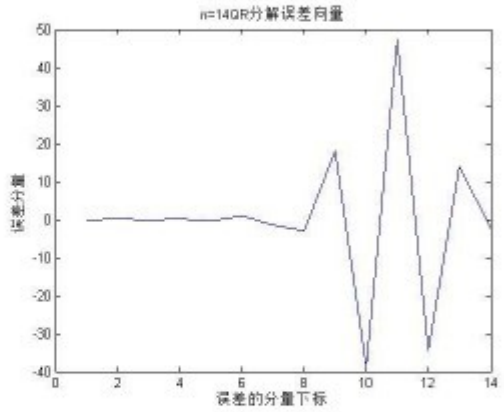


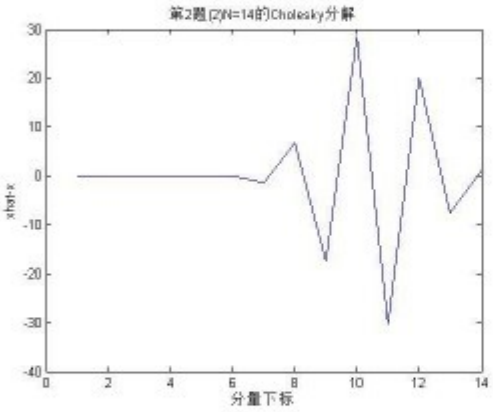
  1. QR 分解求解的误差很小，约为机器精度，和第一次作业中使用的 Gauss 消去法，Cholesky 分解和 LDL^T 表现相似。这是由于矩阵 A 的良态
  2. 解的误差分量变动很随机，没有明显规律，从大小来看应该也近似服从正态分布。

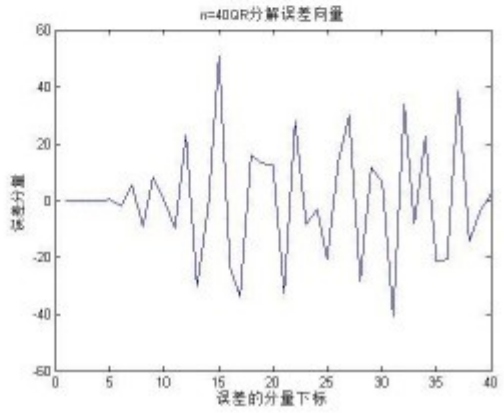
- 方程 3

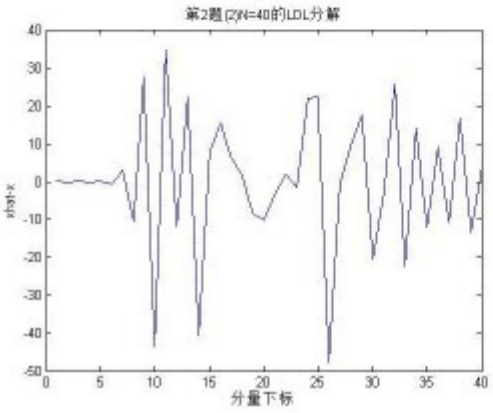
  Hilbert 矩阵：

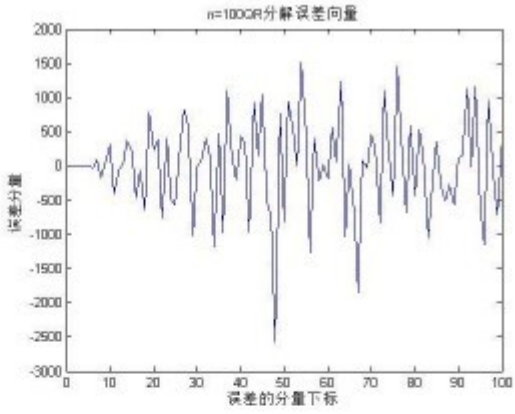
  以下分别对 n=14，40，100 画出误差分量并与 Cholesky，Gauss 消去法作比较：

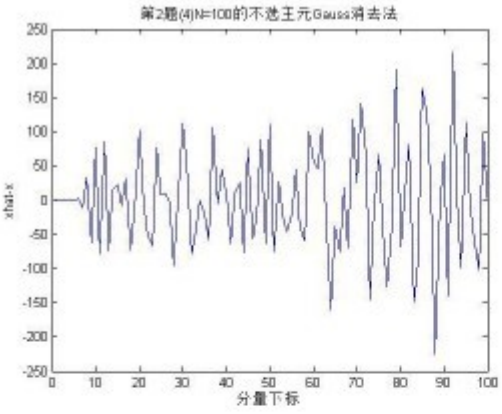

n=14QR分解误差向量
误差分量
误差的分量下标


第2题(2)N=14的Cholesky分解
xhat-x
分量下标


n=40QR分解误差向量
误差分量
误差的分量下标


第2题(2)N=40的LDL分解
xhat-x
分量下标


n=100QR分解误差向量
误差分量
误差的分量下标


第2题(4)N=100的不选主元Gauss消去法
xhat-x
分量下标

从以上图可以看到：

1. QR 分解可以对较大的 n 求解，比 Choslesky 强
2. QR 分解的误差也很大，n=14 时，比 Cholesky 分解误差大，n=40 比 LDL^T 分解误差大，n=100 是不选主元的 Gauss 分解误差的 10 倍
3. 无论哪种方法得到的误差模式都很相似，说明这是方程本身的特性，或者说是矩阵的性质，和采取的计算方法无关。

综上看来：

1. 虽然 QR 分解的运算量要更大，并且上课老师说这个方法要更精确，但是对这三个方程并未表现出明显的差别。相反，QR 方法的误差反而比第一章方法误差要大。
2. 这里的正交分解使用的是 Householder 变换，如果采用 Givens 变换结果或许会有所差别。因为本次作业的时间限制，并未对其进行实验
3. 因为测试的方程组阶数都很小，没有发现明显的运算时间变化。
4. 观察发现对于同一方程组的不同方法计算得到的误差向量可能大小有很大差别，但是把向量归一化后，误差分量的变化模式都极为相像。

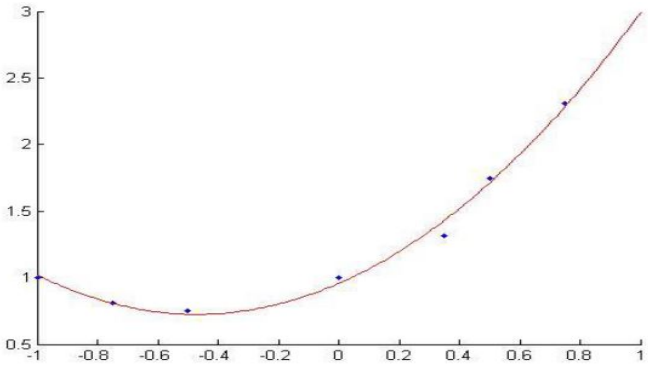
第二题

求解最小 2 乘问题，拟合抛物线

题目模型为 $y = at^2 + bt + c$, 数据略，直接给出利用树上算法得出的 QR 分解的结果：

$$y = 1.0483t^2 + 0.9863t + 0.9569$$

拟合抛物线与原始数据作图如下：



从上图看出，拟合的效果还是不错的，说明方法是没错的

第三题

求解多元回归的实际问题：

先给出书上算法得出的计算结果，在给出用专门的统计软件给出的结果。对两个结果进行对比。共有 n=28 个观测值，k=11 个解释变量，直接给出 QR 分解求解此 OLS 问题的解：

其中 a_0 是截距项

另外使用 stata 对数据进行 OLS 回归可得如下结果：

Source	SS	df	MS
Model	5144.18187	11	467.652898
Residual	267.007044	16	16.6879403
Total	5411.18892	27	200.414404

Number of obs	= 28
F(11, 16)	= 28.02
Prob > F	= 0.0000
R-squared	= 0.9507
Adj R-squared	= 0.9167
Root MSE	= 4.0851

b	Coef.	Std. Err.	t	P>|t|	[95% Conf.	Interval]
var1	.7188873	.8587489	0.84	0.415	-1.101579	2.539354
var2	9.680207	7.469397	1.30	0.213	-6.154207	25.51462
var3	.1535059	.5232132	0.29	0.773	-.9556566	1.262668
var4	13.67959	5.029469	2.72	0.015	3.017593	24.34159
var5	1.986828	1.803573	1.10	0.287	-1.836575	5.810232
var6	-.9582243	2.658803	-0.36	0.723	-6.594634	4.678186
var7	-.4840239	4.197758	-0.12	0.910	-9.382873	8.414825
var8	-.0736469	.0922074	-0.80	0.436	-.2691179	.1218241
var9	1.0187	.7144894	1.43	0.173	-.4959497	2.53335
var10	1.443524	2.775972	0.52	0.610	-4.441274	7.328321
var11	2.90279	2.525782	1.15	0.267	-2.451629	8.257209
_cons	2.077513	8.462866	0.25	0.809	-15.86296	20.01799

由上可以看出，两种方法对系数的估计是相同的。

从数据来看模型可能存在多重共线性的问题。

本次作业的代码全由 Matlab 编写，感觉比较简单，但是在估计矩阵的无穷范数的问题上，以及用多种方法解方程，还是有很多地方很值的思考的。