线性方程组迭代解法部分上机实习报告

姓名:刘群 学号:2014211591 院系: 地学中心
Email: liu-q14@mails.tsinghua.edu.cn
(本文档由LaTeX编写)

November 1, 2014

1 问题的描述

设$H_n = [h_{ij}] \in \mathbb{R}^{n\times n}$是Hilbert矩阵,即$h_{ij} = \frac{1}{i+j-1}$·,取$x = \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix} \in \mathbb{R}^n$, 并令$b_n = H_n x$, 用SOR迭代方法和共轭梯度法求解$H_n x = b_n$, 并与前面的直接方法做比较.

2 SOR方法介绍

2.1 原理介绍

迭代法是一种求解求解线性代数方程组常用的方法, 它是从某些初始向量出发, 然后用设计好的步骤逐次进行迭代求解, 直至逼近真解.具体来说, 我们要求解线性代数方程组$Ax = b$, 我们可以将矩阵A分解成

$$A = M - N \tag{1}$$

其中M是非奇异的,则原方程可以转化为

$$x = M^{-1}Nx + M^{-1}b = Bx + f \tag{2}$$

其中

$$B = M^{-1}N = I - M^{-1}Af = M^{-1}b \tag{3}$$

B称为迭代矩阵. 我们可以通过选择不同的M和N来构造不同的迭代方法.

SOR方法是一种求解线性代数方程组的迭代方法,现简单介绍如下:

我们可以把A分解为

$$A = D - L - U \tag{4}$$

其中,$D = diag(a_{11}, a_{22}, \cdots, a_{nn})$, $-L$和$-U$为A的严格下三角部分和严格上三角部分(不包括对角线),即

$$-L = \begin{pmatrix} 0 & & & \\ a_{21} & 0 & & \\ \vdots & \ddots & \ddots & \\ a_{n1} & \cdots & a_{n,n-1} & 0 \end{pmatrix}$$

$$-U = \begin{pmatrix} 0 & a_{12} & \cdots & a_{1n} \\ & \ddots & \ddots & \vdots \\ & & 0 & a_{n-1,n} \\ & & & 0 \end{pmatrix}$$

对于$A = D - L - U$ 和任意实数ω来说, 我们有

$$\omega A = (D - \omega L) - ((1-\omega)D + \omega U) \tag{5}$$

如果取

$$M = \frac{1}{\omega}(D - \omega L), N = \frac{1}{\omega}[(1-\omega)D + \omega U] \tag{6}$$

就得到了超松弛迭代法, 也称为SOR迭代法.

$$(D-\omega L)x^{(k+1)}=[(1-\omega)D+\omega U]x^{(k)}+\omega b \tag{7}$$

也可以写成

$$x^{(k+1)}=(D-\omega L)^{-1}[(1-\omega)D+\omega U]x^{(k)}+\omega(D-\omega L)^{-1}b \tag{8}$$

即

$$B=(D-\omega L)^{-1}[(1-\omega)D+\omega U]$$
$$f=\omega(D-\omega L)^{-1}b$$

按照公式(7), 我们可以写出分量形式为

$$x_i^{(k+1)}=(1-\omega)x_i^{(k)}+\frac{\omega}{a_{ii}}\left(b_i-\sum_{j=1}^{i-1}a_{ij}x_j^{(k+1)}-\sum_{j=i+1}^{n}a_{ij}x_j^{(k)}\right),i=1,2,\cdots,n \tag{9}$$

其中ω 叫做松弛因子.

2.2 方案设计

在这里我主要编写了函数sor.m来对线性方程组用SOR方法进行求解, 该函数有三个参数, 分别是A,b和ω, 其中A和b分别是方程组$Ax=b$的系数矩阵A和常数向量b, ω是松弛因子. 该函数的返回值是线性方程组的解x. 在求解过程中, 当前后两次x的结果小于10^{-8}时, 我们就停止迭代过程, 直接输出x. 在函数solve_hilbert_equ.m中, 我们在$n=2-15$的循环中对sor.m进行了调用, 并将每次的结果写在了Excel表格中. 在函数solve_hilbert_equ.m中, 我设定了松弛因子$\omega=1.05$. 同时, 我们还分析了结果的误差和残差向量的大小.

2.3 实验结果与分析

Table 1 是采用SOR方法时计算出的结果, 从中可以看出, 即使是当n很大时, 计算出的结果仍然比较准确, 这一点我们也可以通过Table 2 看出. Table 2 表示的是我们计算出的x与理论真解x值之间的差别, 可以看出即使当n很大时, 误差的结果仍然很小. 此时, 通过Table 3 我们可以看出, 残差也非常小, 达到了$10^{-9},10^{-10}$左右的数量级, 从一个侧面反映出了结果的准确性. 从Table 4 中可以看出, 误差向量的二范数在$n=15$时达到了0.011820538, 稍后我们将看到, 这个误差向量的范数比采用直接法进行求解要小得多.

Table 1: n=2-15时SOR方法计算所得的结果

	n=2	n=3	n=4	n=5	n=6	n=7	n=8	n=9	n=10	n=11	n=12	n=13	n=14	n=15
x_1	1.000000015	0.99999991	1.000000371	0.999998168	1.000010152	1.000004249	1.000000245	0.999994427	0.999990389	0.999993995	0.999996798	0.99999955	1.000002637	1.000006301
x_2	0.999999978	1.000000459	0.999996027	1.000033561	0.999716548	0.999905992	1.000036868	1.000219712	1.000276664	1.000148606	1.000051015	0.999957358	0.999855409	0.999738266
x_3		0.999999578	1.000009301	0.999857195	1.001888919	1.00042827	0.999438089	0.998162804	0.99819952	0.999240514	1.000005974	1.000711692	1.001444919	1.002247953
x_4			0.999994062	1.000213616	0.995140899	0.999654016	1.002360583	1.005263689	1.003829156	1.000823955	0.998772502	0.997028077	0.99536663	0.993697275
x_5				0.999896245	1.005318055	0.998822217	0.996567037	0.995518287	0.998701526	1.001343632	1.002735659	1.003585371	1.004102615	1.004368357
x_6					0.997918589	1.002236699	1.000270388	0.997435534	0.9970438	0.998513673	0.999879959	1.00121289	1.002532365	1.003828668
x_7						0.998944875	1.003064003	1.00273845	0.999585205	0.998162775	0.997825882	0.998032997	0.998611761	0.999469359
x_8							0.998258342	1.003713782	1.002498776	1.00010113	0.998526843	0.997490398	0.996890078	0.996672519
x_9								0.996944651	1.002399345	1.002007286	1.000660962	0.999199063	0.997852997	0.996726944
x_{10}									0.997468601	1.001724576	1.00220416	1.001448066	1.000191091	0.998767723
x_{11}										0.997934196	1.00155475	1.002573108	1.002301974	1.00133333
x_{12}											0.997779854	1.001412978	1.002905659	1.003101475
x_{13}												0.997341735	1.00118708	1.003127874
x_{14}													0.996745871	1.000856185
x_{15}														0.996045381

Table 2: n=2-15时SOR方法计算所得的x的误差Δx

	n=2	n=3	n=4	n=5	n=6	n=7	n=8	n=9	n=10	n=11	n=12	n=13	n=14	n=15
Δx_1	1.48392E-08	-9.046E-08	3.70947E-07	-1.83188E-06	1.01525E-05	4.24925E-06	2.44577E-07	-5.57275E-06	-9.61072E-06	-6.00452E-06	-3.20166E-06	-4.49838E-07	2.6366E-06	6.30111E-06
Δx_2	-2.18608E-08	4.58795E-07	-3.97264E-06	3.35615E-05	-0.000283452	-9.40081E-05	3.68676E-05	0.000219712	0.000276664	0.000148606	5.10152E-05	-4.26418E-05	-0.000144591	-0.000261734
Δx_3		-4.22292E-07	9.30101E-06	-0.000142805	0.001888919	0.00042827	-0.000561911	-0.001837196	-0.00180048	-0.000759486	5.97418E-06	0.000711692	0.001444919	0.002247953
Δx_4			-5.93833E-06	0.000213616	-0.004859101	-0.000345984	0.002360583	0.005263689	0.003829156	0.000823955	-0.001227498	-0.002971923	-0.00463337	-0.006302725
Δx_5				-0.000103755	0.005318055	-0.001177783	-0.003432963	-0.004481713	-0.001298474	0.001343632	0.002735659	0.003585371	0.004102615	0.004368357
Δx_6					-0.002081411	0.002236699	0.000270388	-0.002564466	-0.0029562	-0.001486327	-0.000120041	0.00121289	0.002532365	0.003828668
Δx_7						-0.001055125	0.003064003	0.00273845	-0.000414795	-0.001837225	-0.002174118	-0.001967003	-0.001388239	-0.000530641
Δx_8							-0.001741658	0.003713782	0.002498776	0.00010113	-0.001473157	-0.002509602	-0.003109922	-0.003327481
Δx_9								-0.003055349	0.002399345	0.002007286	0.000660962	-0.000800937	-0.002147003	-0.003273056
Δx_{10}									-0.002531399	0.001724576	0.00220416	0.001448066	0.000191091	-0.001232277
Δx_{11}										-0.002065804	0.00155475	0.002573108	0.002301974	0.00133333
Δx_{12}											-0.002220146	0.001412978	0.002905659	0.003101475
Δx_{13}												-0.002658265	0.00118708	0.003127874
Δx_{14}													-0.003254129	0.000856185
Δx_{15}														-0.003954619

Table 3: n=2-15时SOR方法计算所得的x的残差$r = b - Ax$

	n=2	n=3	n=4	n=5	n=6	n=7	n=8	n=9	n=10	n=11	n=12	n=13	n=14	n=15
r_1	-3.9088E-09	1.82634E-09	-3.79215E-10	8.16662E-11	-1.81815E-11	-3.59917E-11	-6.22569E-11	-1.21566E-10	2.01088E-11	2.66671E-11	3.50369E-11	4.68301E-11	6.53038E-11	9.63802E-11
r_2	-1.32666E-10	-2.12878E-09	1.15464E-09	-4.5585E-10	1.5966E-10	2.74189E-10	4.18583E-10	7.16711E-10	-1.87681E-10	-2.30796E-10	-2.82693E-10	-3.52599E-10	-4.5817E-10	-6.29833E-10
r_3		-8.7105E-11	-9.68329E-10	9.09094E-10	-5.49664E-10	-7.7201E-10	-9.78097E-10	-1.33255E-09	6.61095E-10	7.36618E-10	8.20393E-10	9.26417E-10	1.07853E-09	1.31427E-09
r_4			-4.34088E-11	-5.39092E-10	7.52969E-10	6.69282E-10	4.80739E-10	7.30567E-11	-7.71252E-10	-7.02705E-10	-6.26435E-10	-5.29674E-10	-3.9057E-10	-1.74539E-10
r_5				-2.56981E-11	-3.38475E-10	1.98145E-10	5.32321E-10	9.29068E-10	-1.03178E-10	-2.65752E-10	-4.06147E-10	-5.52298E-10	-7.29506E-10	-9.61497E-10
r_6					-1.69431E-11	-3.38612E-10	-5.19926E-11	4.9876E-10	3.74031E-10	2.61896E-10	1.31003E-10	-2.99232E-11	-2.51174E-10	-5.78382E-10
r_7						-1.5972E-11	-3.59798E-10	-2.62173E-10	2.71524E-10	3.42925E-10	3.53897E-10	3.25833E-10	2.52054E-10	1.01861E-10
r_8							-1.66241E-11	-5.54515E-10	-4.89065E-11	1.29985E-10	2.61146E-10	3.66031E-10	4.58946E-10	5.3672E-10
r_9								-2.54676E-11	-2.13524E-10	-1.11098E-10	3.64302E-11	1.95591E-10	3.79505E-10	6.05669E-10
r_{10}									-9.97724E-12	-1.8919E-10	-1.47675E-10	-3.01683E-11	1.46562E-10	4.04078E-10
r_{11}										-8.74523E-12	-1.80203E-10	-1.85297E-10	-9.7775E-11	8.90015E-11
r_{12}											-8.2786E-12	-1.90781E-10	-2.44759E-10	-1.95709E-10
r_{13}												-8.73546E-12	-2.26581E-10	-3.44971E-10
r_{14}													-1.03545E-11	-2.9443E-10
r_{15}														-1.34334E-11

Table 4: n=2-15时SOR方法误差向量的范数

	n=2	n=3	n=4	n=5	n=6	n=7	n=8	n=9	n=10	n=11	n=12	n=13	n=14	n=15
$\|\Delta x\|_2$	2.64216E-08	6.30085E-07	1.17342E-05	0.000279142	0.007737784	0.002795579	0.005492658	0.009403121	0.006854686	0.004466417	0.005342856	0.007228831	0.009423723	0.011820538

3 共轭梯度法(CG)

3.1 CG法基本原理

我们要求线性方程组$Ax = b$的解, 等价于求函数

$$\varphi(x) = \frac{1}{2}(Ax, x) - (b, x) \tag{10}$$

的最小值.

共轭梯度法(Conjugate Gradient)是一种也是采用一维极小搜索的概念,但是不再沿有正交性的$r^{(0)}, r^{(1)}, \cdots$方向进行搜索, 而是要找另一组方向$p^{(0)}, p^{(1)}, \cdots$,这些方向向量是A-共轭的向量组(或者说A-正交向量组), 也就是说它们满足

$$(Ap^{(i)}, p^{(j)}) = 0, i \neq j \tag{11}$$

我们从一个初始点$x^{(0)}$出发, 选取搜索方向$p^{(0)}$和合适的步长α_0就可以得到(1).也就是说,对于$x^{(k+1)}$满足

$$x^{(k+1)} = x^{(k)} + \alpha_k p^{(k)} \tag{12}$$

我们希望α和$p^{(k)}$向量组的选取, 不仅希望使

$$\varphi(x^{(k+1)}) = \min_{\alpha} \varphi(x^{(k)} + \alpha_k p^{(k)}) \tag{13}$$

而且还希望$p^{(k)}$的选取使得

$$\varphi(x^{(k+1)}) = \min_{x \in \text{span}\{p^{(0)}, p^{(1)}, \cdots, p^{(k)}\}} \varphi(x) \tag{14}$$

通过一系列运算, 我们可以得到

$$\alpha_k = \frac{(r^{(k)}, p^{(k)})}{(Ap^{(k)}, p^{(k)})} \tag{15}$$

对于$p^{(k)}$的选取, 我们要让它与$p^{(0)}, p^{(1)}, \cdots, p^{(k-1)}$都共轭, 这种向量的取法不是唯一的, 在这里, 我们取得是$r^{(k)}$与$p^{(k-1)}$的线性组合, 即

$$p^{(k)} = r^{(k)} + \beta_{k-1} p^{(k-1)} \tag{16}$$

通过令$(Ap^{(k)}, p^{(k)}) = 0$, 我们可以得出

$$\beta_{k-1} = -\frac{(r^{(k)}, Ap^{(k-1)})}{(Ap^{(k-1)}, p^{(k-1)})}$$

从而β_k满足

$$\beta_k = -\frac{(r^{(k+1)}, Ap^{(k)})}{(Ap^{(k)}, p^{(k)})} \tag{17}$$

上面的$p^{(k)}$的选取方法可以证明$p^{(k)}$构成了一个A-共轭向量组.

综上所述, CG算法的步骤如下:

(1) 任取$x^{(0)} \in \mathbb{R}^n$;

(2) $r^{(0)} = b - Ax^{(0)}, p^{(0)} = r^{(0)}$;

(3) 对于$k=0,1,\cdots,$

$$\alpha_k=\frac{(r^{(k)},p^{(k)})}{(Ap^{(k)},p^{(k)})}$$
$$x^{(k+1)}=x^{(k)}+\alpha_k p^{(k)}$$
$$r^{(k+1)}=r^{(k)}-\alpha_k Ap^{(k)}$$
$$\beta_k=-\frac{(r^{(k+1)},Ap^{(k)})}{(Ap^{(k)},p^{(k)})}$$
$$p^{(k+1)}=r^{(k+1)}+\beta_k p^{(k)}$$

对于上述(3), 我们可以进行适当的简化, 变成

$$\alpha_k=\frac{(r^{(k)},r^{(k)})}{(Ap^{(k)},p^{(k)})}$$
$$x^{(k+1)}=x^{(k)}+\alpha_k p^{(k)}$$
$$r^{(k+1)}=r^{(k)}-\alpha_k Ap^{(k)}$$
$$\beta_k=-\frac{(r^{(k+1)},r^{(k+1)})}{(r^{(k)},r^{(k)})}$$
$$p^{(k+1)}=r^{(k+1)}+\beta_k p^{(k)}$$

3.2 实验设计

我们编写了函数CG.m采用上述CG算法对线性方程组进行求解, 其中该函数有两个参数, 分别是A和b, 即线性方程组$Ax=b$的系数矩阵和常数向量, 返回值是线性方程组的解. 在求解过程中, 当前后两次x的结果小于10^{-8}时, 我们就停止迭代过程, 直接输出x. 在函数solve_hilbert_equ.m中, 我们在$n=2-15$的循环中对CG.m进行了调用, 得到了方程组的解, 并将每次的结果写在了Excel表格中. 同时, 我们还分析了结果的误差和残差向量的大小.

3.3 实验结果及分析

Table 5 是采用CG方法时计算出的结果, 从中可以看出, 即使是当n很大时, 计算出的结果仍然比较准确, 这一点我们也可以通过Table 6 看出. Table 6 表示的是我们计算出的x与理论真解x值之间的误差, 可以看出即使当n很大时, 误差的结果仍然很小. 此时, 通过Table 7 我们可以看出, 残差也非常小, 达到了$10^{-9},10^{-10}$左右的数量级, 从另一个侧面反映出了结果的准确性. 从Table 8 中可以看出, 误差向量的二范数在$n=15$时达到了0.004318824 .

Table 5: n=2-15时CG方法计算所得的结果

	n=2	n=3	n=4	n=5	n=6	n=7	n=8	n=9	n=10	n=11	n=12	n=13	n=14	n=15
x_1	1	1	0.999999995	1	1.000001161	1.000004967	1.000012767	1.000025608	0.999998591	0.999997335	0.9999955	0.999992987	0.999989716	0.999985621
x_2	1	1	0.999999997	1	0.999966893	0.999876321	0.999715578	0.999480264	1.000042231	1.000074161	1.000117519	1.000173019	1.000241048	1.00032174
x_3		1	0.999999998	1	1.000223778	1.000690539	1.001360589	1.002183407	0.999719051	0.999549659	0.999341633	0.999098319	0.99882348	0.998520949
x_4			0.999999998	1	0.999418469	0.998758943	0.998230288	0.997908104	1.000599223	1.000820994	1.001037884	1.001239441	1.001418425	1.001570103
x_5				1	1.000641352	1.000229423	0.999354479	0.998333188	0.999825585	0.999974797	1.00019086	1.000458276	1.000762499	1.001090819
x_6					0.999747475	1.001211514	1.001369856	1.000788987	0.999494975	0.999348737	0.999285468	0.999306502	0.999405871	0.999574327
x_7						0.99922407	1.001461138	1.00229747	0.999890578	0.999564828	0.999253107	0.998992146	0.998799709	0.998682263
x_8							0.998483224	1.001371353	1.000432121	1.000215896	0.999870712	0.999479015	0.99909189	0.998740105
x_9								0.997585006	1.000466468	1.000686786	1.00058524	1.000292935	0.999898231	0.999460503
x_{10}									0.999529785	1.000480007	1.00088924	1.000935185	1.00074327	1.000402385
x_{11}										0.999283956	1.000428184	1.001024116	1.001231463	1.001167247
x_{12}											0.998999504	1.000315992	1.001088282	1.001458829
x_{13}												0.998683522	1.000150588	1.001084962
x_{14}													0.998342246	0.999939561
x_{15}														0.997980981

Table 6: n=2-15时CG方法计算所得的x的误差Δx

	n=2	n=3	n=4	n=5	n=6	n=7	n=8	n=9	n=10	n=11	n=12	n=13	n=14	n=15
Δx_1	2.88658E-15	1.11022E-14	-5.476E-09	-1.91243E-11	1.16085E-06	4.96688E-06	1.27673E-05	2.56079E-05	-1.40923E-06	-2.6655E-06	-4.50039E-06	-7.01277E-06	-1.02842E-05	-1.43792E-05
Δx_2	1.55431E-15	9.99201E-15	-3.13044E-09	6.69838E-11	-3.31069E-05	-0.000123679	-0.000284422	-0.000519736	4.22309E-05	7.41606E-05	0.000117519	0.000173019	0.000241048	0.00032174
Δx_3		-6.88338E-15	-2.19943E-09	-1.2107E-12	0.000223778	0.000690539	0.001360589	0.002183407	-0.000280949	-0.000450341	-0.000658367	-0.000901681	-0.00117652	-0.001479051
Δx_4			-1.75614E-09	-1.01916E-11	-0.000581531	-0.001241057	-0.001769712	-0.002091896	0.000599223	0.000820994	0.001037884	0.001239441	0.001418425	0.001570103
Δx_5				-5.81637E-11	0.000641352	0.000229423	-0.000645521	-0.001666812	-0.000174415	-2.52035E-05	0.00019086	0.000458276	0.000762499	0.001090819
Δx_6					-0.000252525	0.001211514	0.001369856	0.000788987	-0.000505025	-0.000651263	-0.000714532	-0.000693498	-0.000594129	-0.000425673
Δx_7						-0.00077593	0.001461138	0.00229747	-0.000109422	-0.000435172	-0.000746893	-0.001007854	-0.001200291	-0.001317737
Δx_8							-0.001516776	0.001371353	0.000432121	0.000215896	-0.000129288	-0.000520985	-0.00090811	-0.001259895
Δx_9								-0.002414994	0.000466468	0.000686786	0.00058524	0.000292935	-0.000101769	-0.000539497
Δx_{10}									-0.000470215	0.000480007	0.00088924	0.000935185	0.00074327	0.000402385
Δx_{11}										-0.000716044	0.000428184	0.001024116	0.001231463	0.001167247
Δx_{12}											-0.001000496	0.000315992	0.001088282	0.001458829
Δx_{13}												-0.001316478	0.000150588	0.001084962
Δx_{14}													-0.001657754	-6.0439E-05
Δx_{15}														-0.002019019

Table 7: n=2-15时CG方法计算所得的x的残差$r = b - Ax$

	n=2	n=3	n=4	n=5	n=6	n=7	n=8	n=9	n=10	n=11	n=12	n=13	n=14	n=15
r_1	-3.55271E-15	-1.37668E-14	8.21339E-09	2.16271E-13	-6.15827E-11	4.11347E-10	-5.04263E-11	-1.37405E-10	-1.24802E-10	4.14779E-13	-2.38716E-11	-2.24802E-11	4.94675E-11	6.18807E-11
r_2	-1.9984E-15	-7.10543E-15	4.68256E-09	-7.30971E-13	2.46692E-11	-4.67535E-11	3.7727E-10	1.5005E-09	2.4154E-10	-1.80789E-11	-5.66658E-12	-7.25051E-11	-3.0887E-10	-4.55455E-10
r_3		-4.88498E-15	3.34052E-09	-1.21458E-13	1.23026E-11	-5.58534E-10	-1.85518E-09	-6.23657E-09	1.92464E-10	1.15119E-10	3.25267E-10	6.79465E-10	1.22188E-09	2.28631E-09
r_4			2.61253E-09	3.12417E-13	1.02497E-10	3.98693E-10	2.55719E-09	6.67677E-09	-2.1174E-11	-2.18952E-10	-4.73774E-10	-9.3297E-10	-1.68944E-09	-2.68549E-09
r_5				5.70544E-13	-3.05532E-11	-4.07377E-10	5.6006E-10	3.88814E-09	2.02176E-11	3.32578E-11	-1.76505E-11	-1.93477E-10	-6.03065E-10	-1.32712E-09
r_6					6.44179E-11	-7.64671E-10	-1.83326E-09	-2.80397E-09	1.26565E-13	1.61806E-10	3.23486E-10	5.48714E-10	7.96409E-10	9.72499E-10
r_7						1.40887E-10	-1.55253E-09	-5.63901E-09	-8.98244E-11	8.03863E-11	2.56775E-10	6.07121E-10	1.17122E-09	1.89065E-09
r_8							1.79958E-09	-2.58523E-09	-1.70271E-10	-6.83963E-11	-2.09688E-11	2.02043E-10	6.97863E-10	1.44407E-09
r_9								5.7437E-09	-1.80055E-10	-1.48772E-10	-2.69393E-10	-2.90851E-10	-9.19697E-11	3.20513E-10
r_{10}									-9.24143E-11	-8.4861E-11	-3.31745E-10	-5.92696E-10	-7.53008E-10	-8.28254E-10
r_{11}										1.48029E-10	-1.37795E-10	-5.54208E-10	-1.01568E-09	-1.56779E-09
r_{12}											3.22969E-10	-1.25129E-10	-7.59572E-10	-1.6712E-09
r_{13}												6.83114E-10	3.52884E-11	-1.0617E-09
r_{14}													1.33019E-09	2.4331E-10
r_{15}														2.17424E-09

Table 8: n=2-15时CG方法误差向量的范数

	n=2	n=3	n=4	n=5	n=6	n=7	n=8	n=9	n=10	n=11	n=12	n=13	n=14	n=15
$\|\Delta x\|_2$	3.27845E-15	1.64463E-14	6.90708E-09	9.13286E-11	0.00092976	0.002038347	0.003434076	0.005079961	0.001167327	0.001660695	0.002227918	0.002863483	0.003562104	0.004318824

Table 9: n=2-15时Gauss消去法正则化后计算结果的误差Δx

	n=2	n=3	n=4	n=5	n=6	n=7	n=8	n=9	n=10	n=11	n=12	n=13	n=14	n=15
Δx_1	0.004298821	-0.009991912	-0.019417952	-0.019488126	-0.015355294	-0.010422874	-0.005647367	-0.001217068	0.002834614	0.006488677	0.009722412	0.012515934	0.014859627	0.016757701
Δx_2	-0.008210934	0.066041208	0.08115782	0.058334086	0.029012539	0.003561422	-0.017241849	-0.034167283	-0.047847811	-0.058672041	-0.06689584	-0.072732553	-0.076399644	-0.07813359
Δx_3		-0.068628197	0.004347947	0.03793107	0.047284045	0.04635532	0.041139992	0.033958205	0.025835772	0.01733812	0.008843097	0.00062883	-0.00709854	-0.014194716
Δx_4			-0.090295179	-0.02033764	0.018198443	0.038706139	0.049461902	0.054445166	0.055638232	0.054159257	0.050738743	0.045916888	0.040124269	0.033713715
Δx_5				-0.086575995	-0.029497785	0.006022157	0.029001798	0.04416527	0.053958481	0.059765514	0.062474449	0.062731848	0.061055444	0.057881672
Δx_6					-0.082252121	-0.036714928	-0.004960087	0.018003781	0.034815755	0.04696769	0.055400269	0.06078449	0.063652814	0.06445756
Δx_7						-0.082278974	-0.044375911	-0.015622288	0.006670544	0.024018468	0.037361094	0.047353433	0.054507111	0.059256218
Δx_8							-0.085155772	-0.052176377	-0.025691405	-0.004204446	0.013196704	0.027132485	0.038077667	0.046432213
Δx_9								-0.089229706	-0.059580679	-0.03484679	-0.014155129	0.003072943	0.017269504	0.028794211
Δx_{10}									-0.093488959	-0.066235936	-0.042908224	-0.022970389	-0.006033655	0.008220257
Δx_{11}										-0.097394675	-0.071974731	-0.049827927	-0.030606819	-0.014032963
Δx_{12}											-0.100695522	-0.076762076	-0.055651422	-0.03712218
Δx_{13}												-0.103310933	-0.0806477	-0.060480945
Δx_{14}													-0.105260158	-0.08372986
Δx_{15}														-0.106617096

4 与直接法结果的比较

这里我仅仅列上采用正则化之后的直接法(Gauss消去法和Cholesky分解法)的误差结果, 作为与本实验结果的比较, 如Table 9 和Table 10 所示.

通过Table 9, Table 10与Table 2 和Table 6 对比可以看出, 不论是采用SOR方法还是CG方法, 采用迭代法之后的效果比直接法的结果更加准确. 我们再来看一下误差向量的情况. 如Table 11 和Table 10所示, 是采用正则化后的Gauss消去法和Cholesky分解法时所得的误差向量的二范数(**注意:它们实际上不相等,相差10^{-8}左右, 但这里取的位数比较少, 因此看起来相等, 实际上不相等, 特此说明.**) 可以看出, 采用正则化后的Gauss消去法和Cholesky分解法时, 当$n = 15$时, 误差的向量范数都在0.0324左右. 从Table 4 和Table 8中可以看出, SOR方法和CG方法的误差向量的二范数在$n = 15$分别为0.011820538和0.004318824, 明显小于采用正则化后直接法的结果. 因此可以说采用迭代法的效果较好.

Table 10: n=2-15时Cholesky分解法正则化后计算结果的误差向量Δx

	n=2	n=3	n=4	n=5	n=6	n=7	n=8	n=9	n=10	n=11	n=12	n=13	n=14	n=15
Δx_1	0.004298821	-0.009991912	-0.019417952	-0.019488126	-0.015355294	-0.010422874	-0.005647367	-0.001217068	0.002834614	0.006488677	0.009722412	0.012515934	0.014859627	0.016757701
Δx_2	-0.008210934	0.066041208	0.08115782	0.058334086	0.029012539	0.003561422	-0.017241849	-0.034167283	-0.047847811	-0.058672041	-0.06689584	-0.072732553	-0.076399644	-0.07813359
Δx_3		-0.068628197	0.004347947	0.03793107	0.047284045	0.04635532	0.041139992	0.033958205	0.025835772	0.01733812	0.008843097	0.00062883	-0.00709854	-0.014194716
Δx_4			-0.090295179	-0.02033764	0.018198443	0.038706139	0.049461902	0.054445166	0.055638232	0.054159257	0.050738743	0.045916888	0.040124269	0.033713715
Δx_5				-0.086575995	-0.029497785	0.006022157	0.029001798	0.04416527	0.053958481	0.059765514	0.062474449	0.062731848	0.061055444	0.057881672
Δx_6					-0.082252121	-0.036714928	-0.004960087	0.018003781	0.034815755	0.04696769	0.055400269	0.06078449	0.063652814	0.06445756
Δx_7						-0.082278974	-0.044375911	-0.015622288	0.006670544	0.024018468	0.037361094	0.047353433	0.054507111	0.059256218
Δx_8							-0.085155772	-0.052176377	-0.025691405	-0.004204446	0.013196704	0.027132485	0.038077667	0.046432213
Δx_9								-0.089229706	-0.059580679	-0.03484679	-0.014155129	0.003072943	0.017269504	0.028794211
Δx_{10}									-0.093488959	-0.066235936	-0.042908224	-0.022970389	-0.006033655	0.008220257
Δx_{11}										-0.097394675	-0.071974731	-0.049827927	-0.030606819	-0.014032963
Δx_{12}											-0.100695522	-0.076762076	-0.055651422	-0.03712218
Δx_{13}												-0.103310933	-0.0806477	-0.060480945
Δx_{14}													-0.105260158	-0.08372986
Δx_{15}														-0.106617096

Table 11: n=2-15时Gauss方法正则化后计算结果误差向量的范数

	n=2	n=3	n=4	n=5	n=6	n=7	n=8	n=9	n=10	n=11	n=12	n=13	n=14	n=15
$\|\Delta x\|_2$	9.4824E-06	0.001401083	0.020726106	0.022569853	0.016451714	0.017159102	0.021882421	0.02717609	0.031531454	0.034181662	0.035008877	0.034494486	0.033405417	0.032438342

Table 12: n=2-15时Cholesky法正则化后计算结果误差向量的范数

	n=2	n=3	n=4	n=5	n=6	n=7	n=8	n=9	n=10	n=11	n=12	n=13	n=14	n=15
$\|\Delta x\|_2$	9.4824E-06	0.001401083	0.020726106	0.022569854	0.016451715	0.017159102	0.02188242	0.027176094	0.03153145	0.034181662	0.035008877	0.034494485	0.033405416	0.032438342

5 实验小结

同过这次实验可以看出, 采用迭代法对线性代数方程组求解(特别是大型稀疏矩阵)时效果比直接法要好, 精度更高. 并且, 通过SOR方法与CG方法的对比也可以看出, 采用CG算法的精度要比SOR方法高.

6 Matlab源程序

(1) **sor.m**

```
function x=sor(A, b, w)
% x=sort(A, b, w)
% 超松弛迭代法
% x: return value, the solution to Ax=b
% w should between (0, 2)
% Author: LIU Qun
% Time: 2014-10-24

n=length(b);
x=zeros(n, 1);
err=1;
while(err>1e-8)
    x_init=x;
    for i=1:n
        x(i)=(1-w)*x_init(i)+w/A(i,i)*(b(i)-A(i,1:i-1)*x(1:i-1)-A(i,i+1:
            n)*x(i+1:n));
    end
    err=max(abs(x-x_init));
end
```

(2) **CG.m**

```
function x = CG(A, b)
% x = CG(A, b)
% Calculate Ax=b using Conjugate Gradient method
% Author: LIU Qun
% Email: liu-q14@mails.tsinghua.edu.cn
% Time: 2014-10-25

n=length(b);
x=zeros(n, 1);
rk=b-A*x;
p=rk;
while(max(abs(rk))>1e-8)
    alpha = (rk' * rk)/(p' * A*p);
    x = x + alpha * p;
    rk1 = rk - alpha*A*p;
    beta = (rk1'*rk1)/(rk'*rk);
    p = rk1 + beta * p;
    rk = rk1;
end
```

(3) **solve_hilbert_equ.m**

```
clear;
clc;

% 定义要写入位置
range = {'A','B','C','D','E','F','G','H','I', 'J','K','L','M','N','O'};
% 定义松弛因子w
w=1.05;

% 定义写入文件的名称
file_sor='sor_x.xlsx';
file_sor_err='sor_delta_x.xlsx';
file_sor_r='sor_r.xlsx';
file_sor_norm_dx='sor_norm_dx.xlsx';

file_CG='CG_x.xlsx';
file_CG_err='CG_delta_x.xlsx';
file_CG_r='CG_r.xlsx';
file_CG_norm_dx='CG_norm_dx.xlsx';
N=15;
norm_delta_x_sor = zeros(1,N-1);
norm_delta_x_CG = zeros(1,N-1);
for n=2:N
    A=hilb(n);
    x=ones(n,1);
    b=A*x;
    %%------------------------ SOR Method ------------------------%%
    y_sor=sor(A, b, w);
    delta_x_sor=y_sor-x;      % 解的误差
    r_sor=b-A*y_sor;          % 残差向量
    % 计算误差的向量范数二范数()
    norm_delta_x_sor(n-1) = norm(delta_x_sor);
    % 写入x
    xlswrite(file_sor,y_sor,[range{n-1},'2:', range{n-1}, num2str(n+1)])
    xlswrite(file_sor,{['n=' num2str(n)]}, [range{n-1} '1:' range{n-1} '
        1']);
    % 写入的误差x
    xlswrite(file_sor_err,delta_x_sor,[range{n-1},'2:', range{n-1},
        num2str(n+1)])
    xlswrite(file_sor_err,{['n=' num2str(n)]}, [range{n-1} '1:' range{n
        -1} '1']);
    % 写入残差
    xlswrite(file_sor_r,r_sor,[range{n-1},'2:', range{n-1}, num2str(n+1)
        ])
    xlswrite(file_sor_r,{['n=' num2str(n)]}, [range{n-1} '1:' range{n-1}
         '1']);
    xlswrite(file_sor_norm_dx,{['n=' num2str(n)]},[range{n-1},'1:',
        range{n-1}, '1'])
    %%------------------------ CG Method ------------------------%%
    y_CG=CG(A, b);
    delta_x_CG=y_CG-x;      % 解的误差
    r_CG=b-A*y_CG;          % 残差向量
    % 计算误差的向量范数二范数()
    norm_delta_x_CG(n-1) = norm(delta_x_CG);
    % 写入x
    xlswrite(file_CG,y_CG,[range{n-1},'2:', range{n-1}, num2str(n+1)])
    xlswrite(file_CG,{['n=' num2str(n)]}, [range{n-1} '1:' range{n-1} '1
        ']);
```

```
    % 写入的误差x
    xlswrite(file_CG_err,delta_x_CG,[range{n-1},'2:', range{n-1},
        num2str(n+1)])
    xlswrite(file_CG_err,{['n=' num2str(n)]}, [range{n-1} '1:' range{n
        -1} '1']);
    % 写入残差
    xlswrite(file_CG_r,r_CG,[range{n-1},'2:', range{n-1}, num2str(n+1)])
    xlswrite(file_CG_r,{['n=' num2str(n)]}, [range{n-1} '1:' range{n-1}
        '1']);
    xlswrite(file_CG_norm_dx,{['n=' num2str(n)]},[range{n-1},'1:', range
        {n-1}, '1'])
end
xlswrite(file_sor_norm_dx,norm_delta_x_sor,[range{1},'2:', range{N-1}, '
    2'])
xlswrite(file_CG_norm_dx,norm_delta_x_CG,[range{1},'2:', range{N-1}, '2'
    ])
```