数值试验报告

袁一杨　141110101

2016-10-01

在本篇实验报告中，

$$T_n = \begin{bmatrix} 2 & -1 & & \\ -1 & \ddots & \ddots & \\ & \ddots & \ddots & -1 \\ & & -1 & 2 \end{bmatrix}$$

$$A_n = \begin{bmatrix} T_n + 2I_n & -I_n & & \\ -I_n & T_n + 2I_n & & \\ & \ddots & \ddots & -I_n \\ & & -I_n & T_n + 2I_n \end{bmatrix}$$

6-6 Jacobi迭代法和Gauss-Seidel方法在不同停机标准下的比较

基本内容

编制程序实现Jacobi迭代方法和Gauss-Seidel方法。对应不同的停机标准（例如残量、相邻差量、后验误差停机标准），比较迭代次数以及算法停止时的真实误差。

Jacobi迭代公式为：

$$x_k = Bx_{k-1} + g$$

其中，

$$B = L + U, g = D^{-1}b$$

GS迭代公式为：

$$x_i^{(k)} = \frac{1}{a_{ii}}(b_i - \sum_{j-1}^{i-1} a_{ij}x_j^{(k)} - \sum_{j=i+1}^{n} a_{ij}x_j^{(k-1)})$$

实验目的

比较Jacobi迭代法与Gauss-Seidel迭代法的性能，以及不同的停机条件对迭代结果的影响。

数据及分析

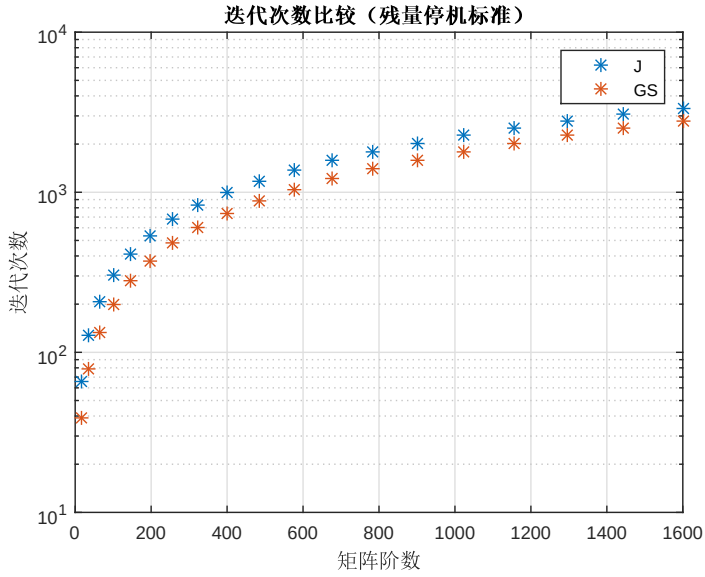
迭代次数的比较:

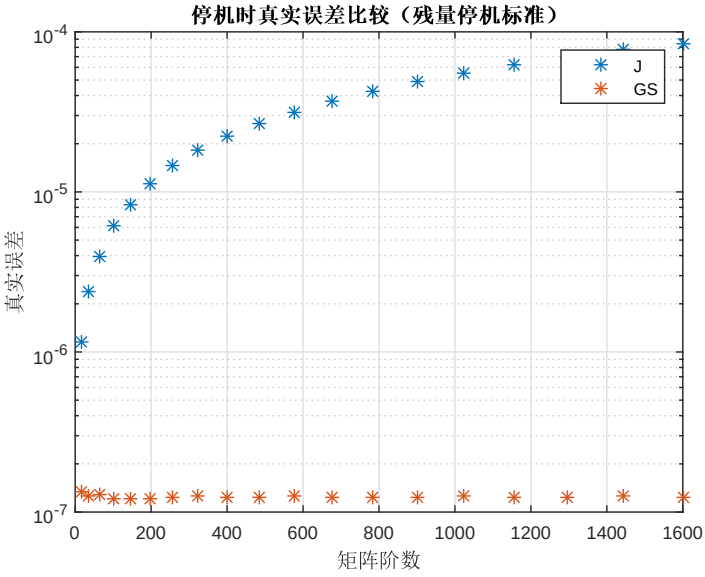
order	Jacobi迭代法			Gauss-Seidel迭代法		
	残差	相邻差量	后验误差	残差	相邻差量	后验误差
3	41	38	38	22	21	21
4	66	60	67	34	33	35
5	95	86	94	49	47	50
6	128	116	137	65	63	70
7	165	150	172	84	80	91
8	207	185	230	105	100	116
9	253	226	270	128	121	143
10	301	269	345	152	144	174

算法停止时真实误差（无穷范数）的对比（数量级为 10^{-6}）:

order	Jacobi迭代法			Gauss-Seidel迭代法		
	残差	相邻差量	后验误差	残差	相邻差量	后验误差
3	0.95367	2.86102	2.86102	0.44703	0.89407	0.89407
4	1.15394	4.11566	0.93356	1.10960	1.69531	0.72623
5	1.78986	6.59963	2.08817	1.32550	2.35644	0.99412
6	2.37610	8.30514	0.92951	2.35667	3.57650	0.83061
7	3.34509	11.00367	1.92786	2.84981	5.36899	0.94065
8	3.94189	15.48875	0.94271	3.70053	6.89296	0.94178
9	4.88160	18.94688	2.08265	4.41979	8.92304	0.98082
10	6.16003	23.13372	0.99867	5.91688	11.46631	0.95925

取一种停止准则可视化为图像（阶数比表中大）





结论

从运行的结果，可以得出以下一些结论：

1. 同样的停机标准下，Gauss-Seidel迭代方法往往比Jacobi 迭代方法迭代速度快，且迭代终止时的精度更高。

2. 对于同一种迭代方法，相邻差量作为停机标准时迭代的次数最少，迭代终止时的精度一般也就最差。当n比较大时，后验误差作为停机标准时迭代次数最多，精度也一般最高。并且随着n的增加，参量和相邻差量将越来越大，这两种停机标准将不宜作为这一类型的方程组的停机标准。

3. 对于Gauss-Seidel迭代方法，真实误差能够被很好地控制在预先给定的后验误差的容限下；而对于Jacobi 迭代方法，可以注意到，只有当n为偶数时，真实误差控制在预先给定的后验误差的容限下。

4. 当阶数足够大，取不同的范数，迭代次数也会相差很多，但相对于总的迭代次数，所占比例不大。

6-7 实现SOR迭代法

基本内容

编写程序实现SOR迭代方法。以真实误差作为停机标准，利用 *rand*随机生成的 ω的值，绘制迭代次数图像，找出最佳迭代因子的取值。在这里，我取200个随机 ω,选出使得迭代次数最少的 ω的值。

实验目的

观测SOR松弛因子ω的值对迭代效果的影响，并找出最佳迭代因子的取值。

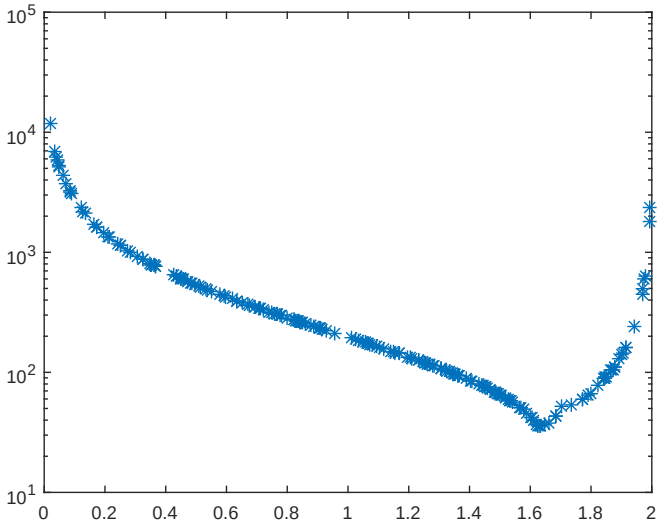
数据及分析

order	ω_{opt}	ω_{opt}(by calculating)
4×4	1.2886	1.2596
6×6	1.4084	1.3948
8×8	1.5279	1.4903
10×10	1.5695	1.5604
12×12	1.6323	1.6138

其中，精确值由

$$\omega_{opt}=\frac{2}{1+(1-(\rho(B))^2)^{1/2}}$$

算出松弛因子与迭代次数的关系如下图（阶数为 12×12）：



结论

1. 通过对图像的观察与分析，这与 $\rho(T_\omega)\omega$之间满足的函数关系式的图像趋势是基本一致的。可以发现随着 ω的增加，迭代次数的整体趋势是先减小后增大。且在靠近 ω_{opt}的左边，函数图像的斜率的变化率交大，而在 ω_{opt}的右侧，图像接近于线性（不考虑接近2的部分）。
2. 在取小数点后三位的精度情况下， ω_{opt}的观测值与理论值偏差不大。
3. 随着n取值的增大， ω_{opt}也在不断扩大。

6-8 J方法、GS方法、SOR方法的比较

基本内容

对于J方法、GS方法和（带有最佳松弛因子的）SOR法，分别绘制误差下降曲线以及残量下降曲线（对数坐标系），绘制（按真实误差）迭代次数与矩阵阶数倒数的关系。

SOR迭代公式：

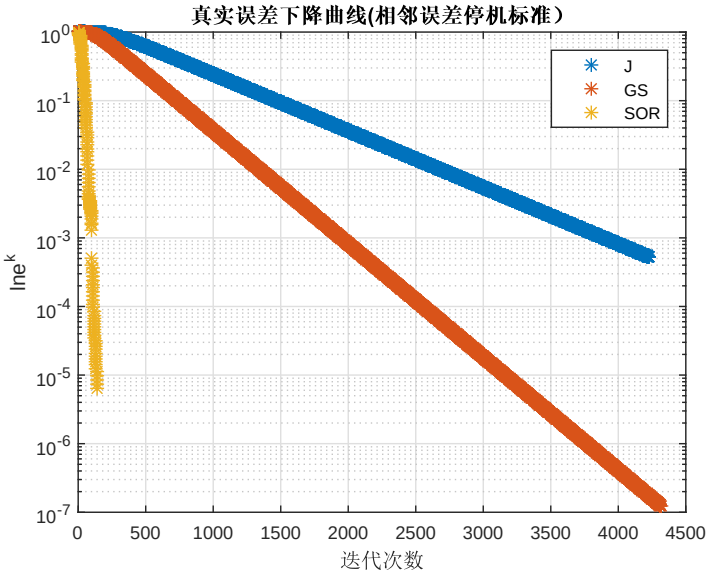
$$x_i^{(k)} = \frac{\omega}{a_{ii}}(b_i - \sum_{j-1}^{i-1} a_{ij}x_j^{(k)} - \sum_{j=i+1}^{n} a_{ij}x_j^{(k-1)}) + (1-\omega)x_i^{(k-1)}$$

实验目的

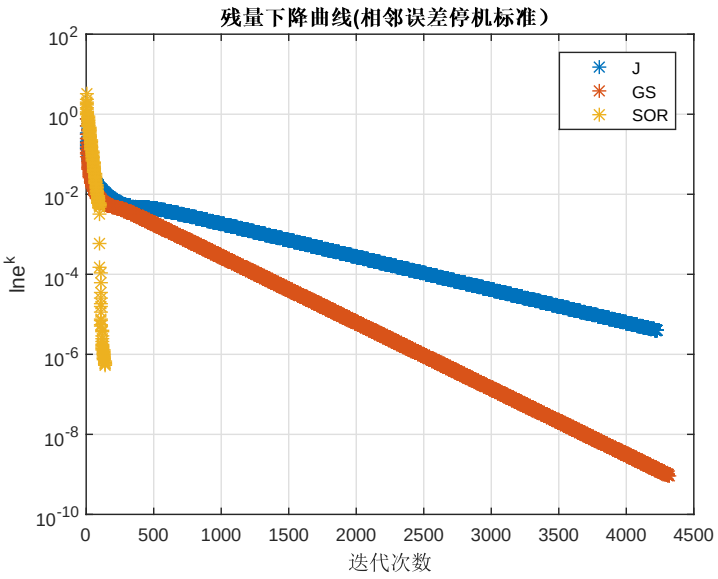
通过将SOR方法与J法、GS法比较，了解SOR法的优越性。同时对三种方法的收敛速度有一个直观的了解。

数据及分析

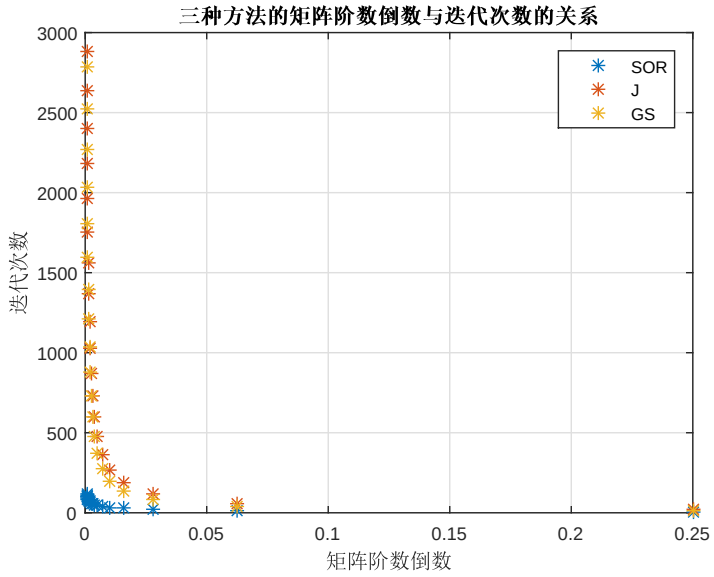
三种方法真实误差下降的比较：



三种方法残差下降的比较：



（按真实误差）迭代次数与矩阵倒数的关系：



结论

1. 随着迭代次数的增加，迭代的精度在不断提高
2. SOR的迭代速度最快，误差最小，其次是GS方法。也验证了理论推导：

带有最佳松弛因子的SOR算法迭代速度J法的平方倍。GS法的迭代速度是J法的两倍。

3. 由第二幅图可以看出，三种方法均是在迭代刚开始的时候收敛速度最快。

4. SOR法的迭代次数随矩阵阶数的变化不大，而J法与GS法均有较为明显的变化。这足以说明SOR算法的优越性：不管矩阵阶数有多大，都可以尽可能地将迭代次数降为最小。

6-9 变系数Richardson迭代法

基本内容

编制变系数Richardson迭代法，绘制误差下降曲线以及残量的下降曲线。观测循环指标m对收敛速度的影响。

迭代公式：

$$x_m = x_{m-1} + \tau_m(b - Ax_{m-1})$$

其中，

$$\frac{1}{\tau_m} = \frac{\lambda_0 - \lambda_1}{2}\cos\theta_m + \frac{\lambda_0 + \lambda_1}{2}$$

有 λ_0为A的最大特征值， λ_1为最小特征值

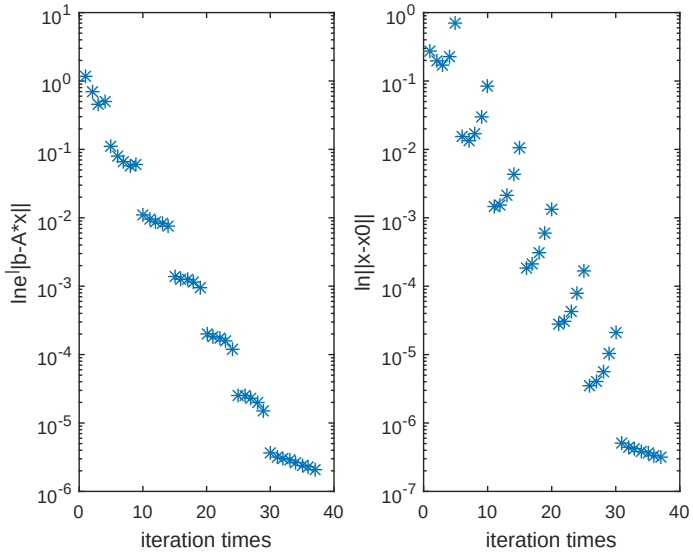
实验目的

了解Richardson变系数迭代法的性能。

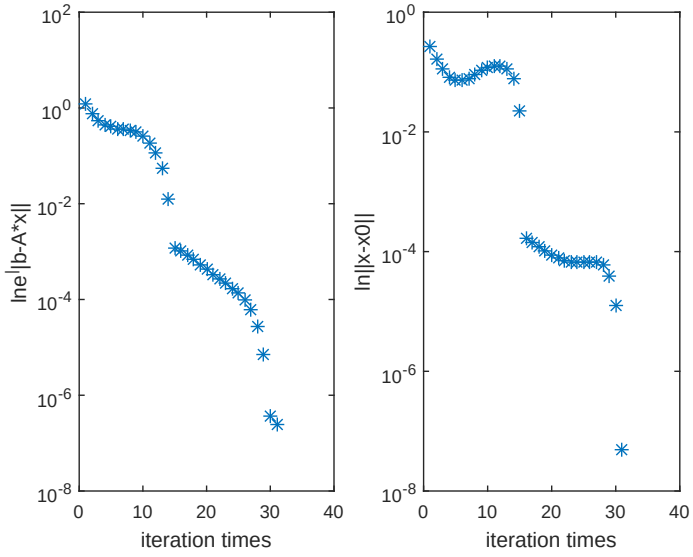
数据及结果

阶数为5时：

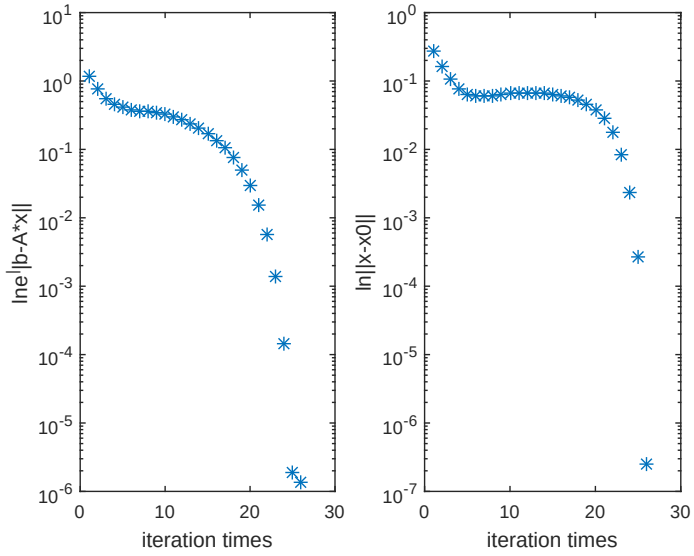
m=5


lne|b-A*x||
ln||x-x0||
iteration times
iteration times

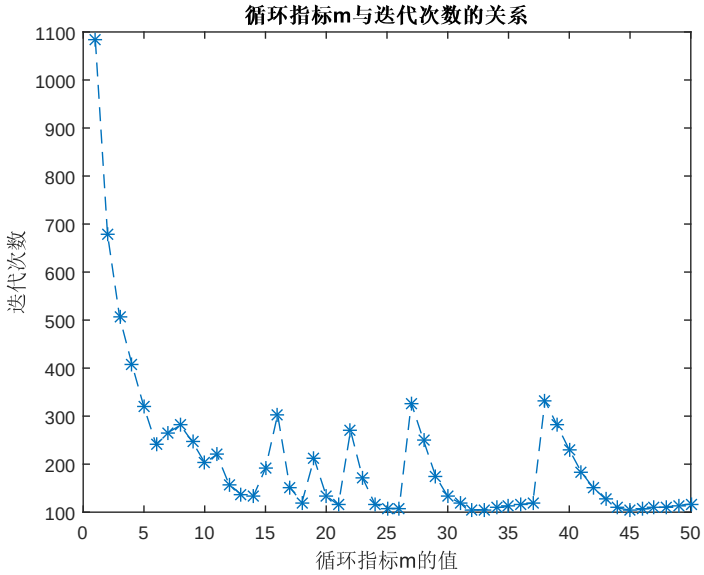
m=15


lne|b-A*x||
ln||x-x0||
iteration times
iteration times

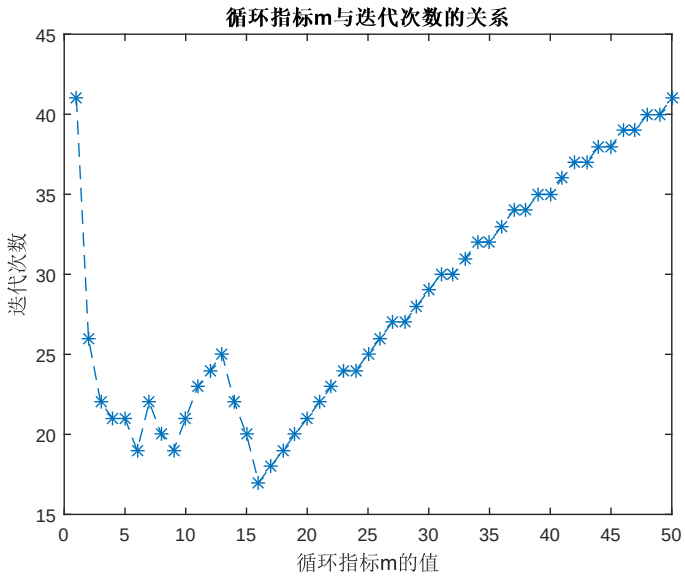
m=25



固定矩阵阶数为（25*25），取m为（1,50）之间的整数：



固定矩阵阶数为（3*3），取m为（1,50）之间的整数：



结论

1. 从矩阵阶数为（5*5）的一组图中，我们可以很清楚的辨别出每一次完整的黑箱迭代。当m从5到15再到25时，每一次黑箱迭代中误差（残量）的反弹幅度在不断地变小。这也说明了合适的m选取的重要性。当m的值偏大或者偏小时，会导致 τ_m 的值出现偏差。好再下一次的黑箱迭代的幅度大于误差偏离的幅度，使得Richardson算法最终是收敛的。

2. 在第二组图中，可以看出，m对于迭代次数的影响具有波动性。当矩阵的阶数比较大时，m的变动对收敛速度展现出波动性，但是当m的取值过大时，就会出现如第二组图中图二所展现出了，迭代次数线性增长，也就是说过大的m所带来的误差（例如舍入误差）过大，已经无法被平衡掉，因此直接引导了迭代次数的走向。

3. 在矩阵阶数较大时，m的取值对迭代的速度影响很大。

6-10 Jacobi半迭代法

基本内容

对Jacobi迭代进行半迭代加速（考虑 $m = 5$ 的循环迭代），绘制误差下降曲线以及残量下降曲线。

迭代公式：

$$x_{k+1} = \rho_{k+1}(\nu(Gx_k + g) + (1-\nu)x_k) + (1-\rho_{k+1})x_{k-1}$$

其中，

$$\nu = \frac{2}{2 - \lambda_{max} - \lambda_{min}} \xi = \frac{2 - \lambda_{max} - \lambda_{min}}{\lambda_{max} - \lambda_{min}}$$

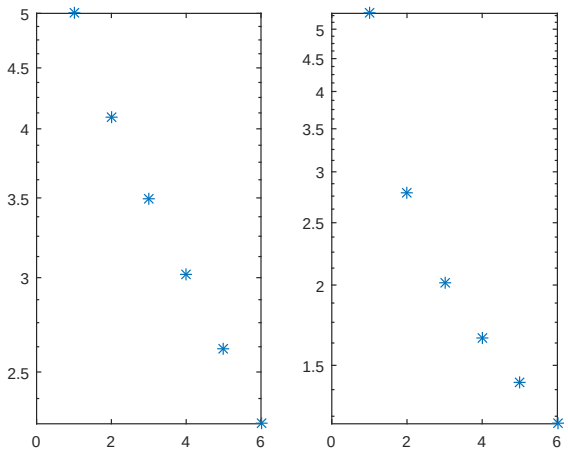
$$\rho_{k+1} = [1 - \frac{1}{4\xi^2}\rho_k]^{-1}, \rho_1 = 2$$

实验目的

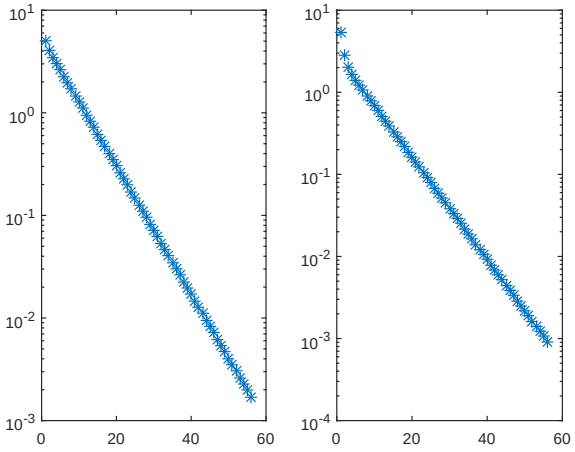
了解半迭代加速的作用。

数据及结果

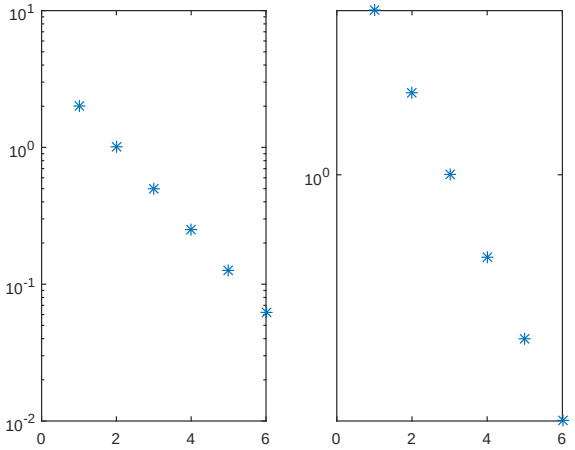
考虑矩阵阶数为（5*5），m=5时



考虑矩阵阶数为（5*5），m=55时



考虑矩阵阶数为（2*2），m=5时



结论

1. 误差是线性下降的
2. 猜测：矩阵的阶数和m的大小需满足一定关系才能保证收敛。矩阵阶数大时，m也要大。有图一与图二比较即可得。

6-11 共轭梯度算法的研究

基本内容

比较 $n = 100, 101$时CG算法与SOR算法的迭代次数；绘制误差下降曲线以及残量下降曲线，绘制迭代次数与矩阵阶数的关系。

迭代方法：

取初始方向为 $p_0 = -r_0 = b - Ax_0$，依次执行

$$x_{k+1} = x_k + \alpha_k p_k, \alpha_k = -\frac{r_k^\top p_k}{p_k^\top A p_k}$$

$$r_{k+1} = r_k + \alpha_k A p_k$$

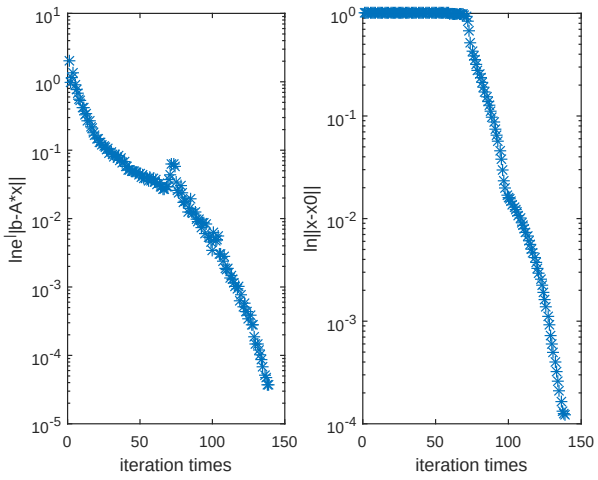
$$p_{k+1} = -r_{k+1} + \beta_k p_k, \beta_k = \frac{r_{k+1}^\top A p_k}{p_k^\top A p_k}$$

数据及结果

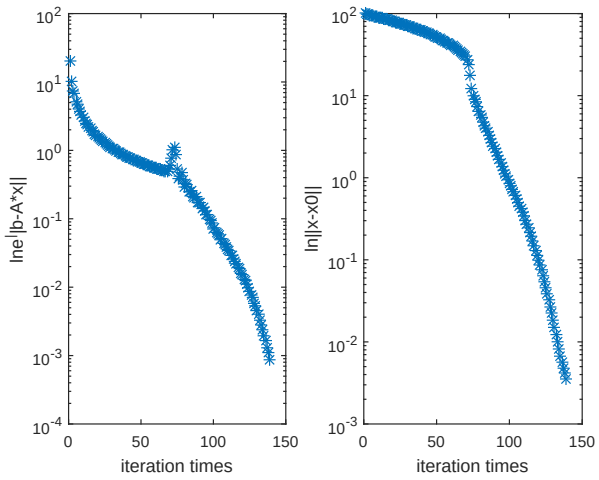
CG算法与SOR算法的比较：

order	CG	SOR
5	6	18
50	71	140
100	136	262
101	140	264

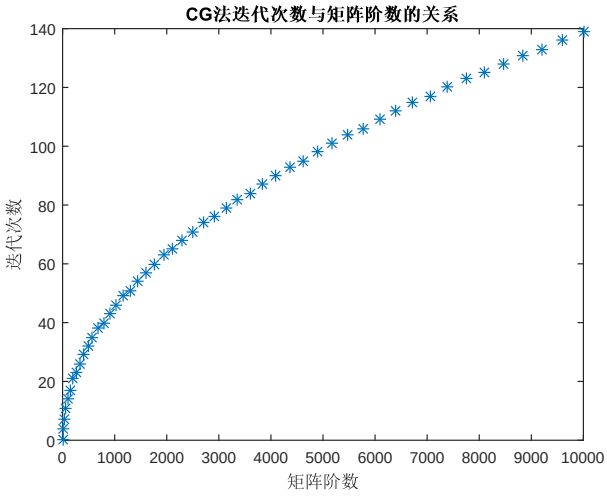
CG算法的误差下降曲线以及残量下降曲线：取范数inf的图像，迭代次数：



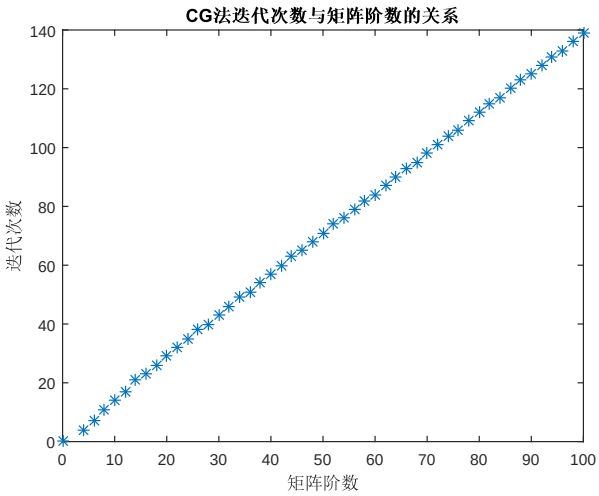
取范数2的图像迭代次数：



CG算法迭代次数与A矩阵阶数（n*n)的关系：



CG算法迭代次数与 T_n 矩阵阶数（n)的关系：



结论

1. CG方法的迭代速度是SOR的两倍，且其矩阵形式的算法公式易于优化。而由6-6知，J法的（8*8）阶的矩阵就需要迭代200次以上，可见CG算法的优越性。

2. 取不同的范数,迭代次数相差不大，但在迭代开始的前部分，无穷范数的误差下降曲线有一个平台阶段。这是因为，在迭代开始的前一段时间，每一次的 x 里都有一个 0存在。而由图二知，取二范数则不会。

3. 可以看出，在迭代开始一段时间后，与Jacobi、GS这些方法不同的是，CG法此时迭代速度会变快。

4. 迭代次数的增加可以看出与n为线性关系。

附录

1. 所有的算法进行都是通过Matlab实现的。虽然有些算法的收敛速度在理论上更快一点，但是在代码实现的过程中，Matlab会对矩阵运算进行优化。也就不难看出通过较多矩阵运算实现的算法在运行时间上较为快一些，因此用时间作为衡量收敛快慢的标准是不妥当的。eg:Jacobi的收敛速度不如GS，但在Matlab实现的过程中，J法要快很多。CG法与SOR法在100阶时迭代次数相差一倍，但运行时间则相差很多。

2. 在矩阵阶数比较大时，范数的选取也会对迭代次数产生影响，在本次报告中，由于考虑到运行速度的问题，如果没有特别说明，均为使用无穷范数。