

Report on Chapter 6&7

—Direct Methods for Solving Linear Systems & Iterative Techniques in Matrix Algebra

Abstract

In this report, we give a brief introduction to 2 direct methods and some iterative methods for solving linear systems and give some numerical experiments to compare them with each other.

Contents

1	Introduction to Algorithms	1
2	Codes and Numerical Experiments	2
2.1	Direct Methods—Gauss Elimination & LL^T Method	2
2.1.1	Gauss Elimination	2
2.1.2	Cholesky factorization	3
2.2	Iterative Methods vs Direct Methods	3
2.2.1	Three Iterative Methods	4
2.2.2	Iterative Methods vs Direct method	6
3	Discussions and Conclusions	8
A	Codes for Gauss Elimination	9
B	Codes for Cholesky Factorization and Solving Systems	10
C	Codes for Jacobi Method	12
D	Codes for Gauss-Seidel Method	13
E	Codes for SOR Method	14

1 Introduction to Algorithms

There are two kinds of methods for solving linear systems—direct method and indirect method.

For direct method, when the coefficient matrix is triangular, it becomes much easier to solve the system by backward substitution. Both of the Gauss elimination method and Cholesky factorization method take this idea. Gauss elimination method eliminates the augmented matrix to a matrix formed by an upper-triangular matrix and a column matrix, then solves the system by backward substitution. Cholesky factorization method factorizes the coefficient matrix into

product of a lower-triangular matrix and its transpose, then solves the system by backward substitution.

For indirect method, we use iterative method to solve linear systems. Although iterative techniques are seldom used to solve linear systems, when the coefficient matrix is sparse, the iterative method is a very efficient way. Our three methods—Jacobi method, Gauss-Seidel method and SOR method use different tricks to transform the original system $Ax = b$ into different equations of form $x = Tx + c$ to accelerate the convergence.

2 Codes and Numerical Experiments

2.1 Direct Methods—Gauss Elimination & LL^T Method

In this section, we use Gauss elimination and Cholesky factorization to solve linear systems. We package these two methods as functions in Matlab, the code is in A, B.

2.1.1 Gauss Elimination

We first generate a 1001×1001 strictly diagonal dominant matrix A using the following code:

```

1 function M=get_mat_sdd(n)
2 %get a n*n matrix which is strictly diagonally dominant
3 M0=ones(n);%n*n matrix of ones
4 M1=eye(n);%n*n identity matrix
5 M2=rand(n);%n*n matrix of uniformly distributed
6 %random number in the interval (0,1)
7 M3=M0-M1;
8 M4=M2.*M3;
9 M=M4+n*M1;
10 end

```

Then we generate a 1001×1 matrix x_0 of random number in $[0, 1]$ and let $b = Ax_0$.

Now, we use our Gauss elimination function to solve the linear system:

$$Ax = b$$

Calculating the norm of $x - x_0$ using Matlab, we get $\|x - x_0\| = 3.765210^{-14}$, which implies the method is very accurate.

2.1.2 Cholesky factorization

As before, we first generate a 1001×1001 symmetric positive definite matrix A using the following code:

```
1 function M=get_mat_pd(n)
2 %get a n*n positive definite matrix
3 M0=rand(n)+eye(n);
4 M=M0*M0';
5 end
```

Then we generate a 1001×1 matrix x_0 of random number in $[0, 1]$ and let $b = Ax_0$.

Using our function of Cholesky factorization, we factorize A into $A = LL^T$.

Now, we use our function of solving linear systems by Cholesky factorization to solve systems

$$Ax = b$$

Calculating the norm of $x - x_0$ using Matlab, we get $\|x - x_0\| = 7.5858 \times 10^{-7}$, which implies the method is accurate.

2.2 Iterative Methods vs Direct Methods

In this section, we compare iterative methods with each other and compare iterative methods with direct method—Gauss elimination. To do this, we package Jacobi method, Gauss-Seidel method and SOR method as functions in Matlab. The codes are in C, D and E.

To begin with, we generate a 1001×1001 sparse matrix A , a 1001×1 matrix x_0 of random number in $[0, 1]$ and let $b = Ax_0$. The codes for generating A are as follows:

```
1 function M=get_mat_sparse(n,p)
2 %get a sparse matrix M with density p in (Mij){i~=j}
3 M0=ones(n); %n*n matrix of ones
```

```

4 M1=eye(n);%n*n identity matrix
5 M2=binornd(1,p,n);%n*n matrix of 2-points distributed
6 %random number in the interval with parameter p
7 ind=M0-M1;
8 M=ind.*M2+n*M1;
9 end

```

We now use these methods to solve linear systems:

$$Ax = b$$

2.2.1 Three Iterative Methods

We use Jacobi method,Gauss-Seidel method,and SOR method with

$$\omega = 1 + 0.01k, k = -9, -8, \dots, 8, 9$$

Setting tolerance of error $TOL = 10^{-10}$,the code is as follow:

```

1 clear
2 clc
3 n=1001;%size of matrix
4 p=0.1;
5 A=get_mat_sparse(n,p);%get sparse matrix
6 x0=rand(n,1);%exact solution
7 b=A*x0;
8 XO=zeros(n,1);%initial value
9 TOL=1e-10;%error tolerance
10 N=100;%tolerance of number of iterations
11 %nt and ms is number of iterations of method
12 %nt and ns is the error of method

```

```

13  mt=zeros ( 1 , 2 );
14  ms=zeros ( 1 , 19 );
15  [ ~ , mt ( 1 ) ] = Jac ( A , b , XO , TOL , N );
16  [ ~ , mt ( 2 ) ] = GS ( A , b , XO , TOL , N );
17  T=10;
18  d=0.01;%step of omega
19  %use SOR method with different omega to get solution
20  for j=1:2*T-1
21      XO=zeros ( n , 1 );
22      omega=1-T*d+j*d;
23      n=length ( b );
24      k=1;
25      s=(1:n);
26      x=zeros ( n , 1 );
27      while k<=N
28          for i=1:n
29              indl=(s<i);
30              indu=(s>i);
31              x ( i )=(1-omega)*XO ( i )+omega*((-sum ( A ( i , indl ) ' ...
32                  .* x ( indl ) )-sum ( A ( i , indu ) ' .* XO ( indu ) ) )+b ( i ))/A ( i , i );
33          end
34          if norm ( x-XO )<TOL
35              m=k;
36              break
37          end

```

```

38         k=k+1;
39         XO=x;
40     end
41     if k>N
42         continue
43     else
44         ms(j)=m;
45     end
46 end

```

The number of iterations at accuracy 10^{-10} is as Table 2.2.1.

Method	Jacobi	Gauss-Seidel	SOR $\omega=0.91$	0.92	0.93	0.94	0.95
Number of Iteration	13	8	12	12	11	11	10
Method	0.96	0.97	0.98	0.99	1.00	1.01	1.02
Number of Iteration	10	9	8	8	8	9	10
Method	1.03	1.04	1.05	1.06	1.07	1.08	1.09
Number of Iteration	10	11	11	12	12	13	13

From Table 2.2.1, we can see that at the same accuracy, Jacobi method uses the most iterations, while Gauss-Seidel method and SOR method with best parameter $\omega = 0.99$ use less iterations. This meets our calculation and expectation.

2.2.2 Iterative Methods vs Direct method

Now we use direct methods—Gauss elimination and iterative methods—Jacobi method, Gauss-Seidel method to solve the system $Ax = b$ and compare the time they cost to show that iterative methods behave better than Gauss elimination when solving sparse systems. The code is as follows:

```

1 clear
2 clc

```

```

3  n=1001;%size of matrix
4  p=0.1;
5  A=get_mat_sparse(n,p);%get sparse matrix
6  x0=rand(n,1);%exact solution
7  b=A*x0;
8  XO=zeros(n,1);
9  N=100;
10 tic
11 xG=GaussE(A,b);
12 toc
13 TOL=0.1*norm(xG-x0);
14 tic
15 xJ=Jac(A,b,XO,TOL,N);
16 toc
17 tic
18 xGS=GS(A,b,XO,TOL,N);
19 toc

```

Firstly, we solve the system by Gauss elimination. Using the tic-toc function from Matlab, after 1.461310s, the method solves the system. Then we get the error of Gauss elimination is $\varepsilon = 2.2340^{-14}$.

Secondly, to show iterative methods behave a lot better than direct methods, we set the tolerance of error to be $TOL = \frac{\varepsilon}{10}$. Then we solve the system by Jacobi method and Gauss-Seidel method. Using the tic-toc function from Matlab, we get the times cost by these two methods are 0.149563s and 0.134174s, which are much smaller than 1.461310s—the time cost by Gauss elimination. Hence we've shown that the iterative methods behave much better than direct methods when solving sparse linear systems.

3 Discussions and Conclusions

For strictly diagonal dominant or positive definite matrix, we can see that both of Gauss elimination method and Cholesky factorization method are accurate.

For sparse systems, the iterative method behaves much better than direct method. It is not only accurate, but it also costs much less time than the direct method.

From the experiment and discussion above, for different linear systems, the tolerance of time cost and different needed accuracy, we know that we should choose different method to solve the systems.

A Codes for Gauss Elimination

```
1  function x=GaussE(A,b)
2  n=length(b);
3  m=zeros(n);
4  for k=1:n-1
5      if max(abs(A(k:n,k)))~=0
6          pp=find(A(k:n,k)~=0);
7          p=pp(1)+k-1;
8      else
9          disp('no unique solution exists. ')
10     end
11     if p~=k
12         l=A(p,:);
13         A(p,:)=A(k,:);
14         A(k,:)=l;
15     end
16     for i=k+1:n
17         m(i,k)=A(i,k)/A(k,k);
18         b(i)=b(i)-m(i,k)*b(k);
19         for j=k+1:n
20             A(i,j)=A(i,j)-m(i,k)*A(k,j);
21         end
22     end
23 end
```

```

24  if A(n,n)~=0
25      x(n)=b(n)/A(n,n);
26  else
27      disp('no unique solution exists. ')
28  end
29  for i=1:n-1
30      j=n-i;
31      x(j)=(b(j)-sum(A(j,j+1:n).*x(j+1:n)))/A(j,j);
32  end
33  x=x';
34  end

```

B Codes for Cholesky Factorization and Solving Systems

```

1  function L=Cho(A)
2  %give L of Cholesky factorization of A
3  nn=size(A);
4  n=nn(1);%get size of A
5  L=zeros(n);
6  L(1,1)=sqrt(A(1,1));
7  for j=2:n
8      L(j,1)=A(1,j)/L(1,1);
9  end
10 for i=2:n-1

```

```

11     L(i,i)=sqrt(A(i,i)-norm(L(i,1:i-1))^2);
12     for j=i+1:n
13         L(j,i)=(A(i,j)-sum(L(i,1:i-1).*L(j,1:i-1)))/L(i,i);
14     end
15 end
16 L(n,n)=sqrt(A(n,n)-norm(L(n,1:n-1))^2);
17 end

```

```

1 function x=Solve_Cho(A,b)
2 L=Cho(A);
3 n=length(b);
4 x=zeros(n,1);
5 y=zeros(n,1);
6 y(1)=b(1)/L(1,1);
7 for i=2:n
8     y(i)=(b(i)-sum(L(i,1:i-1)'.*y(1:i-1)))/L(i,i);
9 end
10 x(n)=y(n)/L(n,n);
11 for j=1:n-1
12     i=n-j;
13     x(i)=(y(i)-sum(L(i+1:n,i).*x(i+1:n)))/L(i,i);
14 end
15 end

```

C Codes for Jacobi Method

```
1 function [x,m]=Jac(A,b,XO,TOL,N)
2 %Jacobi Iterative
3 %x is the approximate solution
4 %m is the final number of iterations
5 n=length(b);
6 k=1;
7 s=(1:n);
8 x=zeros(n,1);
9 while k<=N
10     for i=1:n
11         ind=(s~=i);
12         x(i)=(-sum(A(i,ind)'.*XO(ind))+b(i))/A(i,i);
13     end
14     if norm(x-XO)<TOL
15         m=k;
16         break
17     end
18     k=k+1;
19     XO=x;
20 end
21 if k>N
22     disp('Maximum number of iterations exceeded')
23 end
```

D Codes for Gauss-Seidel Method

```
1  function [x,m]=GS(A,b,XO,TOL,N)
2  %Gauss-Seidel Iterative
3  %x is the approximate solution
4  %m is the final number of iterations
5  n=length(b);
6  k=1;
7  s=(1:n);
8  x=zeros(n,1);
9  while k<=N
10     for i=1:n
11         indl=(s<i);
12         indu=(s>i);
13         x(i)=(-sum(A(i,indl)'.*x(indl))...
14             -sum(A(i,indu)'.*XO(indu))+b(i))/A(i,i);
15     end
16     if norm(x-XO)<TOL
17         m=k;
18         break
19     end
20     k=k+1;
```

```

21      XO=x;

22  end

23  if k>N

24      disp('Maximum number of iterations exceeded')

25  end

26  end

```

E Codes for SOR Method

```

1  function [x,m]=SOR(A,b,XO,omega,TOL,N)

2  %SOR

3  %x is the approximate solution

4  %m is the final number of iterations

5  n=length(b);

6  k=1;

7  s=(1:n);

8  x=zeros(n,1);

9  while k<=N

10     for i=1:n

11         indl=(s<i);

12         indu=(s>i);

13         x(i)=(1-omega)*XO(i)+omega*((-sum(A(i,indl)'.*x(indl))...

14             -sum(A(i,indu)'.*XO(indu)))+b(i))/A(i,i);

15     end

```

```

16         if norm(x-XO)<TOL
17             m=k;
18             break
19         end
20         k=k+1;
21         XO=x;
22     end
23     if k>N
24         disp('Maximum number of iterations exceeded')
25     end
26 end

```

References

- [1] Richard L. Burden, J. Douglas Faires, Annette M. Burden, Numerical Analysis (Tenth Edition), Cengage Learning, 2014.