Scientific Computing Homework #1

李阳 11935018

May 1, 2020

Problem 1. *What is the inverse of the following matrix?*

$$A = \begin{bmatrix} 1 & 0 & 0 \\ 1 & -1 & 0 \\ 1 & -2 & 1 \end{bmatrix}$$

Solution.

$$A^{-1} = \begin{bmatrix} 1 & 0 & 0 \\ 1 & -1 & 0 \\ 1 & -2 & 1 \end{bmatrix}$$

□

Problem 2. *Let*

$$A = \begin{bmatrix} 1 & 1+\epsilon \\ 1-\epsilon & 1 \end{bmatrix}.$$

(a) What is the determinant of A?

(b) In floating-point arithmetic, for what range of values of ϵ will the computed value of the determinant be zero?

(c) What is the LU factorization of A?

(d) In floating-point arithmetic, for what range of values of ϵ will the computed value of U be singular?

Solution. (a)

$$\det(A) = 1 - (1+\epsilon)(1-\epsilon) = \epsilon^2.$$

(b) For $\epsilon \in (-\sqrt{\epsilon_{\text{mach}}}, \sqrt{\epsilon_{\text{mach}}})$, the computed value of the determinant will be zero, where ϵ_{mach} denotes the unit roundoff.

(c) The LU factorization of A is as follows

$$L = \begin{bmatrix} 1 & 0 \\ 1-\epsilon & 1 \end{bmatrix}, \quad U = \begin{bmatrix} 1 & 1+\epsilon \\ 0 & \epsilon^2 \end{bmatrix}.$$

(d) For $\epsilon \in (-\sqrt{\epsilon_{\text{mach}}}, \sqrt{\epsilon_{\text{mach}}})$, the computed value of U will be singular, where ϵ_{mach} denotes the unit roundoff.

□

Problem 3. *(a) What is the LU factorization of the following matrix?*

$$\begin{bmatrix} 1 & a \\ c & b \end{bmatrix}$$

(b) Under what condition is this matrix singular?

Solution. (a) The LU factorization is as follows.

$$L = \begin{bmatrix} 1 & 0 \\ c & 1 \end{bmatrix}, \quad U = \begin{bmatrix} 1 & a \\ 0 & b-ca \end{bmatrix}.$$

(b) The matrix is singular if and only if $b - ca = 0$.

□

Problem 4. *If $A, B,$ and C are $n \times n$ matrices, with B and C nonsingular, and $\mathbf{b}$ an n-vector, how would you implement the formula*

$$\mathbf{x} = B^{-1}(2A + I)(C^{-1} + A)\mathbf{b}$$

without computing any matrix inverses?

Solution. The basic idea is that whenever we see a matrix inverse in a formula, we should think "solve a system" rather than "invert a matrix." So we can implement the formula using the following steps.

- solve $C\mathbf{y} = b$ for $\mathbf{y}$;
- $\mathbf{y} \leftarrow \mathbf{y} + A\mathbf{b}$;
- $\mathbf{w} \leftarrow (2A + I)\mathbf{y}$;
- solve $B\mathbf{x} = \mathbf{w}$ for $\mathbf{x}$.

□

Problem 5. *Prove that the Woodbury formula*

$$(A - UV^T)^{-1} = A^{-1} + A^{-1}U(I - V^TA^{-1}U)^{-1}V^TA^{-1}$$

given in Section 2.4.9 is correct. (Hint: Multiply both sides by $A - UV^T$.)

Proof. Following the hint, multiply both sides by $A - UV^T$, and we have

$$\begin{aligned}
&\left(A - UV^T\right)\left(A^{-1} + A^{-1}U(I - V^TA^{-1}U)^{-1}V^TA^{-1}\right)\\
&= I + U(I - V^TA^{-1}U)^{-1}V^TA^{-1} - UV^TA^{-1} - UV^TA^{-1}U(I - V^TA^{-1}U)^{-1}V^TA^{-1}\\
&= I - UV^TA^{-1} + U(I - V^TA^{-1}U)(I - V^TA^{-1}U)^{-1}V^TA^{-1}\\
&= I - UV^TA^{-1} + UV^TA^{-1}\\
&= I,
\end{aligned}$$

which completes the proof. □

Problem 6. *Suppose that the symmetric matrix*

$$B = \begin{bmatrix} A & \mathbf{a} \\ \mathbf{a}^T & \alpha \end{bmatrix}$$

of order $n + 1$ is positive definite.

(a) Show that the scalar α must be positive and the $n \times n$ matrix A must be positive definite.

(b) What is the Cholesky factorization of B in terms of the constituent submatrices?

Proof. (a) Since B is symmetric positive definite, we have

$$\forall \mathbf{x} \in \mathbb{R}^{n+1}, \mathbf{x} \neq \mathbf{0}, \quad \mathbf{x}^TB\mathbf{x} > 0.$$

In particular, choose $\mathbf{x} = \mathbf{e}_{n+1}$, and thus

$$0 < \mathbf{e}_{n+1}^TB\mathbf{e}_{n+1} = \alpha.$$

Similar, choosing $\mathbf{x} = (\mathbf{y}, 0)$, where $\mathbf{y} \in \mathbb{R}^n, \mathbf{y} \neq \mathbf{0}$, yields

$$0 < \mathbf{x}^TB\mathbf{x} = \mathbf{y}^TA\mathbf{y}, \quad \forall \mathbf{y} \in \mathbb{R}^n, \mathbf{y} \neq \mathbf{0},$$

which shows that A is a positive definite matrix.

(b) Assume that the Cholesky factorization of A is given by

$$A = LL^T,$$

let

$$B = \begin{bmatrix} L & \mathbf{0} \\ \mathbf{b}^T & \sqrt{\alpha} \end{bmatrix}\begin{bmatrix} L^T & \mathbf{b} \\ \mathbf{0}^T & \sqrt{\alpha} \end{bmatrix} = \begin{bmatrix} LL^T & L\mathbf{b} \\ (L\mathbf{b})^T & \alpha \end{bmatrix},$$

and hence

$$L\mathbf{b} = \mathbf{a} \Rightarrow \mathbf{b} = L^{-1}\mathbf{a}.$$

Therefore, the Cholesky factorization of B is

$$B = \begin{bmatrix} L & \mathbf{0} \\ (L^{-1}\mathbf{a})^T & \sqrt{\alpha} \end{bmatrix} \begin{bmatrix} L^T & L\mathbf{a} \\ \mathbf{0}^T & \sqrt{\alpha} \end{bmatrix},$$

where L is obtained by the Cholesky factorization of A, i.e., $A = LL^T$.

□

Problem 7 (Programming assignment)**.** *Consider a horizontal cantilevered beam that is clamped at one end but free along the remainder of its length. A discrete model of the forces on the beam yields a system of linear equations $A\mathbf{x} = \mathbf{b}$, where the $n \times n$ matrix A has the banded form*

$$\begin{bmatrix} 9 & -4 & 1 & 0 & \cdots & \cdots & 0 \\ -4 & 6 & -4 & 1 & \ddots & & \vdots \\ 1 & -4 & 6 & -4 & 1 & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & 1 & -4 & 6 & -4 & 1 \\ \vdots & & \ddots & 1 & -4 & 5 & -2 \\ 0 & \cdots & \cdots & 0 & 1 & -2 & 1 \end{bmatrix},$$

the n-vector $\mathbf{b}$ is the known load on the bar (including its own weight), and the n-vector $\mathbf{x}$ represents the resulting deflection of the bar that is to be determined. We will take the bar to be uniformly loaded, with $b_i = \frac{1}{n^4}$ for each component of the load vector.

(a) Letting $n = 100$, solve this linear system using both a standard library routine for dense linear systems and a library routine designed for banded (or more general sparse) systems. How do the two routines compare in the time required to compute the solution? How well do the answers obtained agree with each other?

(b) Verify that the matrix A has the UL factorization $A = RR^T$, where R is an upper triangular matrix of the form

$$\begin{bmatrix} 2 & -2 & 1 & 0 & \cdots & 0 \\ 0 & 1 & -2 & 1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & 0 \\ \vdots & & \ddots & 1 & -2 & 1 \\ \vdots & & & \ddots & 1 & -2 \\ 0 & \cdots & \cdots & \cdots & 0 & 1 \end{bmatrix}.$$

Letting $n = 1000$, solve the linear system using this factorization (two triangular solves will be required). Also solve the system in its original form using a banded (or general sparse) system solver as in part (a). How well do the answers obtained agree with each other? Which approach seems more accurate? What is the condition number of A, and what accurary does it suggest that you should expect? Try iterative refinement to see if the accurary or residual improves for the less accurate method.

Solution. (a) The matlab routines are as follows. The time for the standard library routine for dense linear systems and the standard routines designed for banded systems we use are given approximately as follows.

```
Dense: 0.000226 seconds;
Sparse: 0.000163 seconds.
```

The answers obtained almost agree with each other, since the ∞-norm of the difference between the two results we get is

```
6.01e-11.
```

```
% This routine solves the linear system
% in Programming Assignment (a)
% using a dense storage for the coefficient matrix

% Dimension of the matrix
n = 100;

% set up the coefficient matrix
A = zeros(n, n);
for i=1:n-2
    A(i,i)=6;
    A(i,i+1)=-4; A(i+1,i)=-4;
    A(i, i+2)=1; A(i+2,i)=1;
end
A(1,1)=9; A(n-1,n-1)=5; A(n,n)=1;
A(n,n-1)=-2; A(n-1,n)=-2;

% set up the right-hand side
rhs = 1.0/(n*n*n*n)*ones(n, 1);
tic
x = A\rhs;
toc

disp(['The condition number of A is ', num2str(cond(A))])
```

```
% This routine solves the linear system
% in Programming Assignment (a)
% using a sparse storage for the coefficient matrix

% Dimension of the matrix
n = 1000;

% set up the coefficient matrix
a = ones(n, 1);
b = -4*ones(n, 1);
b(end-1) = -2;
c = 6*ones(n, 1);
c(1) = 9; c(end-1) = 5; c(end) = 1;
d = -4*ones(n, 1);
d(end) = -2;
A = spdiags([a b c d a], -2:2, n, n);

% set up the right-hand side
rhs = 1.0/(n*n*n*n) * ones(n, 1);
tic
x = A\rhs;
toc
```

(b) Verifying that $A = RR^T$ is straightforward.

The matlab routines are as follows. The ∞-norm of the difference between the two results we get is

```
1.59e-07.
```

The condition number of the matrix A is

```
1.31e+08
```

```
n = 1000;

% set up the right-hand side
rhs = 1.0/(n*n*n*n)*ones(n, 1);

% backward substitution
rhs(n-1) = rhs(n-1) + 2*rhs(n);
for k = n-2:-1:1
    rhs(k) = rhs(k) + 2*rhs(k+1) - rhs(k+2);
end
rhs(1) = rhs(1)/2;

% forward substitution
rhs(1) = rhs(1)/2;
rhs(2) = rhs(2) + 2*rhs(1);
for k = 3:n
```

```
    rhs(k) = rhs(k) + 2*rhs(k-1) - rhs(k-2);
end

% set up the coefficient matrix
a = ones(n, 1);
b = -4*ones(n, 1);
b(end-1) = -2;
c = 6*ones(n, 1);
c(1) = 9; c(end-1) = 5; c(end) = 1;
d = -4*ones(n, 1);
d(end) = -2;
A = spdiags([a b c d a], -2:2, n, n);

% set up the right-hand side
rhs2 = 1.0/(n*n*n*n) * ones(n, 1);
x = A\rhs2;
norm(A*x-rhs2, Inf)
norm(A*rhs-rhs2, Inf)

disp(['inf-norm of the difference of the two solutions is ', num2str(norm(x-rhs, Inf))])
```

□