Chap two

ShangXiaojin

2022 年 12 月 14 日

目录

1 Linear Systems of Equations

1.1 Ex

Use the Gaussian Elimination Algorithm and single-precision arithmetic on a computer to solve the folowing linear systems

- C

$$
\begin{aligned}
x_1 + \frac{1}{2}x_2 + \frac{1}{3}x_3 + \frac{1}{4}x_4 &= \frac{1}{6} \\
\frac{1}{2}x_1 + \frac{1}{3}x_2 + \frac{1}{4}x_3 + \frac{1}{5}x_4 &= \frac{1}{7} \\
\frac{1}{3}x_1 + \frac{1}{4}x_2 + \frac{1}{5}x_3 + \frac{1}{6}x_4 &= \frac{1}{8} \\
\frac{1}{4}x_1 + \frac{1}{5}x_2 + \frac{1}{6}x_3 + \frac{1}{7}x_4 &= \frac{1}{9}
\end{aligned}
$$

- D

$$
\begin{aligned}
2x_1 + x_2 - x_3 + x_4 - 3x_5 &= 7 \\
x_1 + 2x_3 - x_4 + x_5 &= 2 \\
-2x_2 - x_3 + x_4 - x_5 &= -5 \\
3x_1 + x_2 - 4x_3 + 5x_5 &= 6 \\
x_1 - x_2 - x_3 - x_4 + x_5 &= 3
\end{aligned}
$$

1.2 code

1.2.1 main code

```
C = [1 1/2 1/3 1/4 1/6;
     1/2 1/3 1/4 1/5 1/7;
     1/3 1/4 1/5 1/6 1/8;
     1/4 1/5 1/6 1/7 1/9];
D = [2 1 -1 1 -3 7;
     1 0 2 -1 1 2;
     0 -2 -1 1 -1 -5;
     1 -1 -1 -1 1 3];
G_E_backward_single(C)
```

```
G_E_backward_single(D)
\subsubsection{function}

function X = G_E_backward_single(A)
m = size(A,1);
n = size(A,2);
M = [];
X=  [];
for i = 1:m-1
    p = i;
    if A(p,i) == 0
        p = p+1;
    end
   if p ~= i
       A([i p],:) = A([p i],:);
   end
   for j = i+1:m
       M(j,i) = single(A(j,i)/A(i,i));
       A(j,:) = single(A(j,:) - M(j,i).*A(i,:));
   end
end
if A(m,m) == 0
    fprintf('Error')
else
    X(m) =single(A(m,n)/A(m,m));
    for k = 1:m-1
        i = m-k;
        temp = 0;
        for j = i+1:m
            temp = single(A(i,j)*X(j)+temp);
        end
        X(i) = single((A(i,n) - temp)/A(i,i));
    end
end
```

end

1.3 Ans

$$\text{AnsC} = \begin{pmatrix} -0.0317 & 0.5953 & -2.3810 & 2.7778 \end{pmatrix}$$
$$\text{AnsD} = \begin{pmatrix} 2.6087 & 2.2609 & -1.0870 & -1.5652 \end{pmatrix}$$

2 Pivoting Strategles

2.1 Ex

Use Gaussian Elimination and three-digit chopping arithmetic to solve the following linear systems and compare the approximations to the actual solution

- e

$$\begin{aligned} 1.19x_1 + 2.11x_2 - 100x_3 + x_4 &= 1..12 \\ 14.2x_1 - 0.122x_2 + 12.2x_3 - x_4 &= 3.44 \\ 100x_2 - 99.9x_3 + x_4 &= 2.15 \\ 15.3x_1 + 0.11x_2 - 13.1x_3 - x_4 &= 4.16 \end{aligned}$$

- three-digit rounding arithmetic
- partial Pivoting

2.2 code

2.2.1 main code

```
A1 = [1.19  2.11  -100  1  1.12;
14.2  -0.122  12.2  -1  3.44;
0  100  -99.9  1   2.15;
15.3  0.11  -13.1  -1  4.16];
G_E_backward(A1)
G_E_P_P(A1)
```

```
G_E_backward_round3(A1)
G_E_backward_partial(A1)
```

2.2.2 function

- G_E_backward

```
function X = G_E_backward(A)
m = size(A,1);
n = size(A,2);
M = [];
X= [];
for i = 1:m-1
    p = i;
    if A(p,i) == 0
        p = p+1;
    end
    if p ~= i
        A([i p],:) = A([p i],:);
    end
    for j = i+1:m
        M(j,i) = A(j,i)/A(i,i);
        A(j,:) = A(j,:) - M(j,i).*A(i,:);
    end
end
if A(m,m) == 0
    fprintf('Error')
else
    X(m) =A(m,n)/A(m,m);
    for k = 1:m-1
        i = m-k;
        temp = 0;
        for j = i+1:m
            temp = A(i,j)*X(j)+temp;
        end
```

```
        X(i) = (A(i,n) - temp)/A(i,i);
    end
end
end
```

- G_E_backward_round3

```
function X = G_E_backward_round3(A)
m = size(A,1);
n = size(A,2);
M = [];
X=  [];
for i = 1:m-1
    p = i;
    if A(p,i) == 0
        p = p+1;
    end
   if p ~= i
       A([i p],:) = A([p i],:);
   end
   for j = i+1:m
       M(j,i) = round(A(j,i)/A(i,i),3,'significant');
       A(j,:) = round(A(j,:) - M(j,i).*A(i,:),3,'significant');
   end
end
if A(m,m) == 0
    fprintf('Error')
else
    X(m) =round(A(m,n)/A(m,m),3,'significant');
    for k = 1:m-1
        i = m-k;
        temp = 0;
        for j = i+1:m
            temp = round(A(i,j)*X(j)+temp,3,'significant');
```

```
            end
            X(i) = round((A(i,n) - temp)/A(i,i),3,'significant');
        end
    end
    end
```

- G_E_backward_partial

```
function X = G_E_backward_partial(A)
m = size(A,1);
n = size(A,2);
M = [];
X=  [];
for i = 1:m-1
    p = i;
    if A(p,i) == 0
        p = p+1;
    end
   if p ~= i
      A([i p],:) = A([p i],:);
   end
   for j = i+1:m
      M(j,i) = partial(A(j,i)/A(i,i),3);
      A(j,:) = partial(A(j,:) - M(j,i).*A(i,:),3);
   end
end
if A(m,m) == 0
    fprintf('Error')
else
    X(m) =partial(A(m,n)/A(m,m),3);
    for k = 1:m-1
        i = m-k;
        temp = 0;
        for j = i+1:m
```

```
                temp = partial(A(i,j)*X(j)+temp,3);
            end
            X(i) = partial((A(i,n) - temp)/A(i,i),3);
        end
    end
end
```

- G_E_P_P

```
function X = G_E_P_P(A)
X = [];
M= [];
NROW = [];
n = size(A,1);
for i =  1:n
    NROW(i) = i;
end
for i = 1:n-1
    p = i;
    Max = max(abs(A(i:n,i)));
    if abs(A(NROW(p),i)) < Max
        p = p+1;
    end
    if A(NROW(p),i) == 0
        fprintf('error1')
    end
    if NROW(i) ~= NROW(p)
        NCOPY = NROW(i);
        NROW(i) = NROW(p);
        NROW(p) = NCOPY;
    end
    for j = i+1:n
        M(NROW(j),i) = A(NROW(j),i)/A(NROW(i),i);
        A(NROW(j),:) = A(NROW(j),:)- M(NROW(j),i)*A(NROW(i),:);
```

```
    end
end
if A(NROW(n),n) == 0
    fprintf('error2')
end
X(n) = A(NROW(n),n+1)/A(NROW(n),n);
for k = 1:n-1
    i = n-k;
    temp = 0;
    for j = i+1:n
        temp = temp + A(NROW(i),j)*X(j);
    end
    X(i) = (A(NROW(i),n+1) - temp)/A(NROW(i),i);
end

end
```

- partial

```
function y = partial(x,N)
temp1 = zeros(size(x));
temp1(find(x>0)) = 1;
temp2 = ones(size(x)) - temp1;
x1 = temp1.*x;
x2 = temp2.*x;

n11 = floor(log10(x1))+1;
if n11 < N
    n12 = N-n11;
    y1 = floor(x1.*10.^(n12))./10.^(n12);
else
    n12 = n11 - N;
    y1 = floor(x1.*10.^(-n12)).*10.^(n12);
end
```

```
y1(isnan(y1)) = 0;

x2 = abs(x2);
     n21 = floor(log10(x2))+1;
if n21 < N
     n22 = N-n21;
     y2 = floor(x2.*10.^(n22))./10.^(n22);
else
     n22 = n1 - N;
     y2 = floor(x2.*10.^(-n22)).*10.^(n22);

end
y2(isnan(y2)) = 0;
y2 = -y2;
y = y1 + y2;
end
```

2.3 Ans

- G_E_backward(A1) = [0.1768 0.0127 -0.0207 -1.1826]
- G_E_P_P(A1) = [0.1768 0.0127 -0.0207 -1.1826]
- G_E_backward_round3(A1) = [0.1340 0.0095 -0.0275 -1.8100]
- G_E_backward_partial(A1) = [0.1850 0.0166 -0.0195 -1.0800]

3 Matrix Factorization

3.1 Ex

Factor the following matrices into the LU decomposition using the LU Factorization Algorithm with$l_{ii} = 1$.

[2.1756 4.0231 -2.1732 5.1967

-4.0231 6 0 1.1973

-1 -5.2107 1.1111 0
6.0235 7 0 -4.1561]

3.2 code

3.2.1 main code

```
A = [2.1756  4.0231  -2.1732  5.1967;-4.0231   6  0  1.1973;-1  -5.2107  1.1111  0;6.0
X0 = [1  1  1  1];
Y = LU(A,X0)
y = G_E_P_P([Y(1:4,1:4)  ones(4,1)]);
x = G_E_P_P([Y(5:8,5:8)  y'])
```

3.2.2 function

```
function   Y = LU(A,X0)
n = size(A,1);
L = [];
U = [];
L(1,1) = X0(1);
U(1,1) = A(1,1)/L(1,1);
if A(1,1) == 0
    fprintf('error')
end
for j = 2:n
    U(1,j) = A(1,j)/L(1,1);
    L(j,1) = A(j,1)/U(1,1);
end
for i = 2:n-1
    L(i,i)= X0(i);
    temp1 = 0;
    for k = 1:i-1
        temp1 = temp1 + L(i,k)*U(k,i);
    end
```

```
    if A(i,i)-temp1 == 0
        fprintf('error')
    else
    U(i,i) = (A(i,i)-temp1)/L(i,i);
    end
    for j = i+1:n
        temp2 = 0;
        temp3 = 0;
        for k = 1:i-1
            temp2 = temp2 + L(i,k)*U(k,j);
        end
        U(i,j) = (A(i,j) - temp2)/L(i,i);
        for k = 1:i-1
            temp3 = temp3 + L(j,k)*U(k,i);
        end
        L(j,i) = (A(j,i) - temp3)/U(i,i);
    end
end
L(n,n)= X0(n);
temp = 0;
for k =1:n-1
    temp = temp + L(n,k)*U(k,n);
end
U(n,n) = (A(n,n) - temp)/L(n,n);
Y = [L zeros(n,n);zeros(n,n) U];
end
```

3.3 Ans

其中左上矩阵为 L，右下矩阵为 U.x 为最终值

Y =

1.0000	0	0	0	0	0	0	0
−1.8492	1.0000	0	0	0	0	0	0
−0.4596	−0.2501	1.0000	0	0	0	0	0
2.7687	−0.3079	−5.3523	1.0000	0	0	0	0
0	0	0	0	2.1756	4.0231	−2.1732	5.1967
0	0	0	0	0	13.4395	−4.0187	10.8070
0	0	0	0	0	0	−0.8930	5.0917

```
        0        0        0        0        0        0        0    12.0361

x = 0.4466    0.2881    2.6532    0.8919
```

4 Special Types of Matrices

4.1 Ex

Use the LDL Factorization Algorithm to find A =LDL or LL'

A = [6 2 1 -1
2 4 1 0
1 1 4 -1
-1 0 -1 3]

4.2 code

4.2.1 main code

```
A = [6  2  1  -1;
     2  4  1  0;
     1  1  4  -1;
     -1  0  -1  3];
LDL(A)
l = LLt(A)
```

4.2.2 function

```
function P = LDL(A)
n = size(A,1);
v = [];
l = eye(3);
d = [];
D = [];
for i  = 1:n
```

```
    for j = 1:i-1
        v(j) =  l(i,j) * d(j);
    end
    temp = 0;
    for j = 1:i-1
        temp = temp + l(i,j)*v(j);
    end
    d(i) = A(i,i) - temp;
    for j = i+1: n
        temp = 0;
        for k = 1:i-1
            temp = temp + l(j,k)*v(k);
        end
        l(j,i) = (A(j,i) - temp)/d(i);
    end
end
for i = 1:n
    D(i,i) = d(i);
end
P = [l D];
end

function P = LLt(A)
n = size(A);
l = [];
l(1,1) = sqrt(A(1,1));
for j = 2:n
    l(j,1) = A(j,1)/l(1,1);
end
for i = 2:n-1
    temp = 0;
    for k = 1:i-1
        temp = temp + l(i,k).^2;
```

```
    end
    l(i,i) = sqrt(A(i,i) - temp);
    for j = i+1:n
        temp = 0;
        for k = 1:i-1
            temp  = temp  + l(j,k)*l(i,k);
        end
        l(j,i)  =( A(j,i) - temp)/l(i,i);
    end
end
temp = 0;
for k = 1:n-1
    temp = temp  + l(n,k).^2;
end
l(n,n) = sqrt(A(n,n) - temp);
P = l;
end
```

4.3 Ans

其中前 4*4 为 L，后 4*4 为 D

```
LDL = [
1.0000       0        0       0   6.0000       0        0       0
0.3333  1.0000        0       0        0  3.3333        0       0
0.1667  0.2000   1.0000       0        0       0   3.7000       0
-0.1667  0.1000  -0.2432  1.0000        0       0        0  2.5811]
```

对于 LL^t

```
L =
 2.4495        0        0        0
 0.8165   1.8257        0        0
 0.4082   0.3651   1.9235        0
-0.4082   0.1826  -0.4679   1.6066
```