110550126 曾家祐

Numerical methods Assignment 2

1.(a)

```
clear;
clc;
A = [2.51 1.48 4.53;
     1.48 0.93 -1.30;
     2.68 3.04 -1.48];
b = [0.05;1.03;-0.53];
Ab = [A b];
for i = 1:3-1
    for j = i+1:3
        Ab(j,i:4) = round(Ab(j,i:4) -Ab(j,i)/Ab(i,i)*Ab(i,i:4),2);
    end
    Ab
end
x = [0;0;0];
x(3) = round(Ab(3,4)/Ab(3,3),2);

for i = 2:-1:1
    x(i) = round((Ab(i,4)-Ab(i,i+1:3)*x(i+1:3))/Ab(i,i),2);
end
x
```

We first initalize the matrix Ax = b, then we use 2 for loop to make A into upper triangular matrix, the outer loop is the iteration (i) for column, inner loop (j)is to eliminate the row by minus value (A[j][i]/A[i][i])*row i.
Second is back substitution. We first calculate the value of x3 and use a for loop and X's value to caculate the X we want X = [1.62; -1.86; -0.28].
(note that we only calculate to two digits after float point)

```
Ab =

    2.5100    1.4800    4.5300    0.0500
         0    0.0600   -3.9700    1.0000
         0    1.4600   -6.3200   -0.5800


Ab =

    2.5100    1.4800    4.5300    0.0500
         0    0.0600   -3.9700    1.0000
         0         0   90.2800  -24.9100


x =

    1.6200
   -1.8600
   -0.2800
```

(b)

```
clear;
clc;
A = [2.51 1.48 4.53;
     1.48 0.93 -1.30;
     2.68 3.04 -1.48];
b = [0.05;1.03;-0.53];
Ab = [A b];
Ab = [0 0 1; 0 1 0; 1 0 0 ]*Ab;
i = 1;
for j = i+1:3
        Ab(j,i:4) = round(Ab(j,i:4) -Ab(j,i)/Ab(i,i)*Ab(i,i:4),2);
end
Ab
Ab = [1 0 0 ; 0 0 1; 0 1 0;]*Ab
Ab
i = 2;
for j = i+1:3
        Ab(j,i:4) = round(Ab(j,i:4) -Ab(j,i)/Ab(i,i)*Ab(i,i:4),2);
end
Ab
x = [0;0;0];
x(3) = round(Ab(3,4)/Ab(3,3),2);
for i = 2:-1:1
    x(i) = round((Ab(i,4)-Ab(i,i+1:3)*x(i+1:3))/Ab(i,i),2);
end
x
```

The main process is similar with 1(a), the different is I depart the outer loop and between each iteration I change the rows to partial pivoting.
And from the back subsitution the answer is X = [1.43; -1.57; -0.27]

```
Ab =

    2.6800    3.0400   -1.4800   -0.5300
         0   -0.7500   -0.4800    1.3200
         0   -1.3700    5.9200    0.5500


Ab =

    2.6800    3.0400   -1.4800   -0.5300
         0   -1.3700    5.9200    0.5500
         0   -0.7500   -0.4800    1.3200


Ab =

    2.6800    3.0400   -1.4800   -0.5300
         0   -1.3700    5.9200    0.5500
         0         0   -3.7200    1.0200


x =

    1.4300
   -1.5700
   -0.2700
```

(c)

```
clear;
clc;
A = [2.51 1.48 4.53;
     1.48 0.93 -1.30;
     2.68 3.04 -1.48];
b = [0.05;1.03;-0.53];
Ab = [A b];
Ab = [0 0 1; 0 1 0; 1 0 0 ]*Ab;
i = 1;
for j = i+1:3
        Ab(j,i:4) = fix(100*(Ab(j,i:4) -(Ab(j,i)/Ab(i,i))*Ab(i,i:4)))/100;
end
Ab = [1 0 0 ; 0 0 1; 0 1 0;]*Ab;
Ab
i = 2;
for j = i+1:3
        Ab(j,i:4) = fix(100*(Ab(j,i:4) -(Ab(j,i)/Ab(i,i))*Ab(i,i:4)))/100;
end
Ab
x = [0;0;0];
x(3) = fix(100*(Ab(3,4)/Ab(3,3)))/100;
for i = 2:-1:1
    x(i) = fix(100*(Ab(i,4)-Ab(i,i+1:3)*x(i+1:3))/Ab(i,i))/100;
end
x
```

The main process is similar with 1(a), the different is I chop the number rather than rounding. To chop the number in 2 decimal, I times 100 first and fix it then divide by 100. The answer X = [1.43 ; -1.57 ; -0.27]

```
Ab =

    2.6800    3.0400   -1.4800   -0.5300
         0   -1.3600    5.9100    0.5400
         0   -0.7400   -0.4800    1.3200


Ab =

    2.6800    3.0400   -1.4800   -0.5300
         0   -1.3600    5.9100    0.5400
         0         0   -3.6900    1.0200


x =

    1.4300
   -1.5700
   -0.2700
```

(d)

```
clear;
clc;
A = [2.51 1.48 4.53;
     1.48 0.93 -1.30;
     2.68 3.04 -1.48];
x1 = [1.62; -1.86; -0.28];
x2 = [1.43; -1.57; -0.27];
x3 = [1.43; -1.57; -0.27];
X = [1.45310; -1.58919; -0.27489];
sol = [0.05;1.03;-0.53]
sol1 = A*x1
sol2 = A*x2
sol3 = A*x3
error1 = sol1-sol
error2 = sol2-sol
error3 = sol3-sol
```

We set the matrix and the solution in (a)(b)(c) and calculate the output and error. We can see the error of (a) is bigger than (b) and (c). (b) and (c)have same solution and error.

```
sol1 =

    0.0450
    1.0318
   -0.8984


sol2 =

    0.0426
    1.0073
   -0.5408


sol3 =

    0.0426
    1.0073
   -0.5408
```

```
error1 =

   -0.0050
    0.0018
   -0.3684


error2 =

   -0.0074
   -0.0227
   -0.0108


error3 =

   -0.0074
   -0.0227
   -0.0108

fx >>
```

2. (a)

```
A = [4 -1 0 0 0 0;
     -1 4 -1 0 0 0;
     0 -1 4 -1 0 0;
     0 0 -1 4 -1 0;
     0 0 0 -1 4 -1;
     0 0 0 0 -1 4;
     ];
b = [100;200;200;200;200;100];
Ab = [A b];

d = Ab(1,1);
for i = 2:6
    scale = Ab(i,i-1)/d;
    Ab(i,i-1) = 0;
    Ab(i,i) = Ab(i,i)-scale*Ab(i-1,i);
    Ab(i,7) = Ab(i,7)-scale*Ab(i-1,7);
    d = Ab(i,i);
end
X = [0;0;0;0;0;0];
X(6) = Ab(6,7)/Ab(6,6);
Ab(6,6) = 1;
Ab(6,7) = X(6);
for i = 5:-1:1
    X(i) = (Ab(i,7)-Ab(i,i+1)*X(i+1))/Ab(i,i);
    Ab(i,i+1) = 0;
    Ab(i,i) = 1;
    Ab(i,7) = X(i);
end
Ab
X
```

Since the structure of the matric we can use the algorithm:
d1 = Ab[1][1];
for row 2 to n: the scale to multiply on last row to subtract is Ab[i][i-1]/d(i-1)
Since only Ab[i-1][i] and Ab[i-1][n+1] have value we only need to calculate Ab[i][i] and Ab[i][7]; and let di = Ab[i][i]
For back substitution: we can direct compute the X[n] = Ab[n][n+1]/Ab[n][n]
for row n-1 to 1: since only Ab[i][i+1] have value, so we can calculate
X[i] = Ab[i][n] – Ab[i][i+1]*X[i+1]/Ab[i][i];
Finally we get the solution X;

(b)

The solution X = [46.3415, 85.3659, 95.1220, 95.1220, 85.3659, 46.3415]

(c)

Elimination: (2^{nd}~Nth rows) 1 divide,2 multiplies,2 subtracts;
Back substitution: Nth row 1 divide, ((N-1)th ~ 1^{st} rows) 1 divide,1 multiply,1 subtract.
The total arithmetic operations (N-1)*5 +1 +(N-1)*3 = 8N-7

3.(a)

```
clear;
clc;
A = [4.63 -1.21 3.22;
-3.07 5.48 2.11;
1.26 3.11 4.57];
b = [2.22;-3.17;5.11];
init= [0;0;0];
N = 3;
X = [0;0;0];
now = 1;
tol = 0.001;
while 1
    for i =1:N
        X(i) = (b(i)/A(i,i))- (A(i,[1:i-1,i+1:N])*init([1:i-1,i+1:N]))/A(i,i);
    end
    fprintf('Iteration %d: ', now)
    fprintf("X = [%f, %f, %f ]\n",X(1,1),X(2,1),X(3,1))
    if abs(A*X-b)<tol
        break
    end
    init = X;
    now=now+1;
end
```

We first initalize the matrix Ax = b , and set initial guess as [0;0;0], then we use the while loop to iterate until the error of solution is less than tolerant value.

In the while loop we use the equation X = D^(-1)*b – D^(-1)(L+U)*init
And for loop to calculate the next iteration of X.
(X[i] = b[i]/A[i][i]- (A's ith row with A[i][i]= 0)*init's ith column/A[i][i])
(Let X(k+1) = X, X(k) = init)
The answer we get is [-8.985606, -9.480790, 10.047334] with 162 interation

```
Iteration 155: X = [-8.984101, -9.479299, 10.045833 ]
Iteration 156: X = [-8.984349, -9.479544, 10.046080 ]
Iteration 157: X = [-8.984584, -9.479777, 10.046315 ]
Iteration 158: X = [-8.984809, -9.480000, 10.046539 ]
Iteration 159: X = [-8.985023, -9.480212, 10.046752 ]
Iteration 160: X = [-8.985227, -9.480414, 10.046956 ]
Iteration 161: X = [-8.985421, -9.480606, 10.047149 ]
Iteration 162: X = [-8.985606, -9.480790, 10.047334 ]
```

(b)

```
clear;
clc;
A = [4.63 -1.21 3.22;
-3.07 5.48 2.11;
1.26 3.11 4.57];
b = [2.22;-3.17;5.11];
init= [0;0;0];
N = 3;
X = [0;0;0];
Y = [0;0;0];
now = 1;
tol = 0.001;
while 1
    for i =1:N
        X(i) = (b(i)/A(i,i)) - (A(i,[1:i-1,i+1:N])*init([1:i-1,i+1:N]))/A(i,i);
        init(i) = X(i);
    end
    fprintf('Iteration %d: ', now)
    fprintf("X = [%f, %f, %f ]\n",X(1,1),X(2,1),X(3,1))
    if abs(A*X-b)<tol
        break
    end
    Y = X;
    now=now+1;
end
```

The main process is similar with 3(a), the different is in the for loop
When we calculate the X[i-1] we can replace the init[i-1] with X[i-1], when we calculate X[i].
The answer we get is X = [-8.987252, -9.482488, 10.049119] with 78 iterations

```
Iteration 71: X = [-8.984857, -9.480191, 10.046896 ]
Iteration 72: X = [-8.985321, -9.480636, 10.047326 ]
Iteration 73: X = [-8.985737, -9.481035, 10.047712 ]
Iteration 74: X = [-8.986109, -9.481392, 10.048058 ]
Iteration 75: X = [-8.986443, -9.481712, 10.048369 ]
Iteration 76: X = [-8.986743, -9.482000, 10.048647 ]
Iteration 77: X = [-8.987011, -9.482257, 10.048896 ]
Iteration 78: X = [-8.987252, -9.482488, 10.049119 ]
```

4.

```matlab
clear;
clc;
A = [4.63 -1.21 3.22;
-3.07 5.48 2.11;
1.26 3.11 4.57];
b = [2.22;-3.17;5.11];
init= [0;0;0];
N = 3;
Y = [0;0;0];
X = [0;0;0];
w = 1.72;
now = 1;
tol = 0.001;
while 1
    for i =1:N
        X(i) = (b(i)/A(i,i)) - (A(i,[1:i-1,i+1:N])*init([1:i-1,i+1:N]))/A(i,i);
        init(i) = X(i);
    end
    X = Y+w*(X-Y);
    init = X;
    fprintf('Iteration %d: ', now)
    fprintf("X = [%f, %f, %f ]\n",X(1,1),X(2,1),X(3,1))
    if abs(A*X-b)<tol
        break
    end
    Y = X;
    now=now+1;
end
```

The main process is similar with 3(a), the different is I add a variable Y to keep the last iteration(e.g X[i-1] = Y[i]), and when the for loop finish we let X = Y+w*(X-Y) with the overrelaxation factor w to speed up the convergence.
The best w we get is w = 1.72 with X = [-8.987087, -9.482285, 10.048936] with 43 iterations.

```
Iteration 40: X = [-8.985149, -9.480566, 10.047230 ]
Iteration 41: X = [-8.986013, -9.481225, 10.047920 ]
Iteration 42: X = [-8.986513, -9.481837, 10.048471 ]
Iteration 43: X = [-8.987087, -9.482285, 10.048936 ]
```

5.

5.(a) $A = \begin{bmatrix} 10^{10} & 0 \\ 0 & 10^{-10} \end{bmatrix}$

$A^{-1} = \begin{bmatrix} 10^{-10} & 0 \\ 0 & 10^{10} \end{bmatrix}$

$\|A\| = 10^{10}$

$\|A^{-1}\| = 10^{10}$

$\Rightarrow \text{cond}(A) = 10^{20}$

$\Rightarrow$ ill-conditioned

(b) $A = \begin{bmatrix} 10^{10} & 0 \\ 0 & 10^{10} \end{bmatrix}$

$A^{-1} = 10^{-10} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$

$\|A\| = 10^{10}$

$\|A^{-1}\| = 10^{-10}$

$\text{cond}(A) = 10^{10} \times 10^{-10} = 1$

$\Rightarrow$ well-conditioned

(c) $A = \begin{bmatrix} 10^{-10} & 0 \\ 0 & 10^{-10} \end{bmatrix}$

$A^{-1} = 10^{10} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$

$\|A\| = 10^{-10}$

$\|A^{-1}\| = 10^{10}$

$\text{cond}(A) = 10^{-10} \times 10^{10} = 1$

$\Rightarrow$ well-conditioned

(d) $A = \begin{bmatrix} 1 & 2 \\ 2 & 4 \end{bmatrix}$

since A is singular

$\Rightarrow \text{cond}(A) = \infty$

$\Rightarrow$ ill-conditional

6.

```
clear;
clc;
A = [4 -1  1  0  0  0;
    -1  4 -1  1  0  0;
     1 -1  4 -1  1  0;
     0  1 -1  4 -1  1;
     0  0  1 -1  4 -1;
     0  0  0  1 -1  4;
     ];
b = [100;200;300;300;200;100];
Ab = [A b];
w = 2;

siz = length(b);
d = Ab(1,1);
for k = 1:siz-1% reference row: k
    for i = k+1:k+w %process eliminate row:
        if i>siz
            break
        end
        scale = Ab(i,k)/d;
        Ab(i,k) = 0;
        for j = k+1:k+w%process eliminate column:j
            if j >siz
                break
            end
            Ab(i,j) = Ab(i,j)-scale*Ab(k,j);
        end
        Ab(i,siz+1) = Ab(i,siz+1)-scale*Ab(k,siz+1);
    end
    d = Ab(k+1,k+1);
end
X = zeros(siz,1);
X(siz) = Ab(siz,siz+1)/Ab(siz,siz);
Ab(siz,siz) = 1;
Ab(siz,siz+1) = X(siz);
for i = siz:-1:2 %column
    for j = i-1:-1:i-w %row
        if j <1
            break
        end
        Ab(j,siz+1) = Ab(j,siz+1)-Ab(j,i)*X(i);
        Ab(j,i) = 0;
    end
    Ab(i-1,siz+1) = Ab(i-1,siz+1)/Ab(i-1,i-1);
    Ab(i-1,i-1) = 1;
    X(i-1) = Ab(i-1,siz+1);
end
Ab
X
A*X
```

First we initialize the matrix Ax = b and the bandwidth w.

Different from problem 2, we not only need to eliminate one row, we need to eliminate w row.

Set d = Ab[1][1]

The first for loop (k) represent the row that we need to subtract.

And the second for loop (i) represent the row we are process, we need to process row from k+1 to k+w. Each row, have to subtract scale*row k. (scale = Ab(i,k)/d). further more in the row we subtract from only need to process column k+1 to k+w, since other column in row k is zero, so we use third for loop to calculate it.

When the second for loop finish, we set d = Ab[k+1][k+1]

Last, is the back substitution: same as above we need to substitute w rows

The outer loop(i) represent the variable we calculate and inner loop (j) to calculate the upper w rows

When finish the loop we got the solution X.

(underneath is the test case and soltion)

```
A = [4 -1  1  0  0  0;
    -1  4 -1  1  0  0;
     1 -1  4 -1  1  0;
     0  1 -1  4 -1  1;
     0  0  1 -1  4 -1;
     0  0  0  1 -1  4;
     ];
b = [100;200;300;300;200;100];
Ab = [A b];
w = 2;
```

```
X =

   14.6341
   53.6585
   95.1220
   95.1220
   53.6585
   14.6341
```

```
ans =

  100.0000
  200.0000
  300.0000
  300.0000
  200.0000
  100.0000

fx >>
```