

수치 해석 과제#6

2015170305 김태희

과제6-1

1)코드와 결과

The first screenshot shows the initial code and the first part of the output. The code defines a 50x50 matrix 'a' and performs row elimination. The output shows the matrix 'a' being read and the first row being processed.

```

1 program gauss_eli
2 implicit none
3 real :: a(50,50),c,temp,soln(50),factor
4 integer :: pivot,t,n,n2,k,p,o,i,row,col,a_i,a_j,rmax
5
6 print *, "How much is Matrix's size? i,j"
7 read *,a_i,a_j
8 n=a_i
9 n2=a_j
10 do p=1,a_i
11 do o=1,a_j
12 print *, "a<<< ",p," ",o,">>> = ?"
13 read *,c
14 a(p,o)=c
15 end do
16 end do
17
18 do pivot=1,n-1
19 rmax=pivot
20 do row=pivot,a_i
21 if (ABS(a(rmax,pivot))<cabs(a(row,pivot))) then
22 rmax=row
23 end if
24 if (rmax /= pivot) then
25 do k=1,a_j
26 temp=a(rmax,k)
27 a(rmax,k)=a(pivot,k)
28 a(pivot,k)=temp
29 end do
30 end if
31 !elimination
32 do row=pivot+1,n
33 factor=a(row,pivot)/a(pivot,pivot)
34 do col=1,n2
35 a(row,col)=a(row,col)-a(pivot,col)*factor
36 end do
37 end do
38 end do
39
40 !back substitution
41 soln=0
42 do o=n,1,-1
43 do p=n2,o,-1
44 a(o,n2)=a(o,n2)-soln(p)*a(o,p)
45 end do
46 soln(o)=a(o,n2)/a(o,o)
47 end do
48 do i=1,n
49 print *,soln(i)
50 end do
51
52
53
54
55 end program gauss_eli

```

The second screenshot shows the continuation of the code and the final output. The code performs back substitution and prints the solution. The output shows the solution vector 'soln'.

```

20 do row=pivot,a_i
21 if (ABS(a(rmax,pivot))<cabs(a(row,pivot))) then
22 rmax=row
23 end if
24 if (rmax /= pivot) then
25 do k=1,a_j
26 temp=a(rmax,k)
27 a(rmax,k)=a(pivot,k)
28 a(pivot,k)=temp
29 end do
30 end if
31 !elimination
32 do row=pivot+1,n
33 factor=a(row,pivot)/a(pivot,pivot)
34 do col=1,n2
35 a(row,col)=a(row,col)-a(pivot,col)*factor
36 end do
37 end do
38 end do
39
40 !back substitution
41 soln=0
42 do o=n,1,-1
43 do p=n2,o,-1
44 a(o,n2)=a(o,n2)-soln(p)*a(o,p)
45 end do
46 soln(o)=a(o,n2)/a(o,o)
47 end do
48 do i=1,n
49 print *,soln(i)
50 end do
51
52
53
54
55 end program gauss_eli

```

The output shows the matrix 'a' being read and the first row being processed. The solution vector 'soln' is printed.

```

a<<< 3, 2>>> = ?
a<<< 3, 3>>> = ?
a<<< 3, 4>>> = ?
a<<< 3, 5>>> = ?
a<<< 4, 1>>> = ?
a<<< 4, 2>>> = ?
a<<< 4, 3>>> = ?
a<<< 4, 4>>> = ?
a<<< 4, 5>>> = ?
-0.500000
1.00000
0.333333
-2.00000
Press RETURN to close window...

```

2)코드 설명

(1)program gauss_eli를 시작한다.

(2)implicit none을 하고 실수변수로 a(50,50),c,temp,soln(50),factor 변수를 선언하고 정수 변수로 pivot,t,n,n2,k,p,o,i,row,col,a_i,a_j,rmax를 선언한다. a는 2차원배열로 행렬 a를 나타내며 soln은 방정식의 해의 값을 1차원 배열로 저장한다. temp는 변수 값 교환 시 사용되며 factor는 pivot,row,col는 이름 그대로를 의미하고 factor는 elimination시 pivot에 곱하

는 값, $a_{i,j}$ 는 각각 augmented 행렬의 행과, 열의 개수 n, n_2 는 각각 이를 저장하는 변수이다. T, k, o, p, i 는 do로 반복문을 실행하기 위해 쓰이는 변수이고 r_{max} 는 어떤 열내에서 최대 값을 가지는 row를 의미한다.

(3)read $a_{i,j}$ 로 augmented matrix의 정보를 받고 그것을 각각 n, n_2 에 저장한다.

(4)행렬의 성분을 a 변수에 받기위해 do문을 먼저 행의 범위로 $p=1, a_i$ 열의 범위로 $o=1, a_j$ 로 하고 print 문의 문자열이 매번 출력되며 그 문자열에 해당되는 행렬의 성분을 입력하면 c 로 저장되고 이 값을 $a(p,o)$ 로 할당한다.

(5)실제로 elimination이 구현되는 부분은 수업자료에서는 4단 loop문으로 되어있지만 사실 $T=1$ 일 때만 해도 elimination이 모두 구현되므로 최외각 do $T=1, a_i$ 는 제거하고 do $pivot=T, n-1$ 을 do $pivot=1, n-1$ 로 수정하였다. $pivot$ 은 행렬의 column을 의미하고 이 loop 문은 elimination을 $1 \sim n-1$ column 까지 진행 할 것이라는 의미이다. r_{max} 는 일단 pivot column과 같게 초기화하고

```
do row=pivot, a_i
    if(ABS(a(rmax,pivot))<abs(a(row,pivot))) then
        rmax=row
    end if
end do
```

코드를 통해 어떤 col 내에 성분중 절대값이 최대일때의 row값을 r_{max} 로 반환한다.

```
if(rmax /= pivot) then
    do k=1, a_j
        temp=a(rmax,k)
        a(rmax,k)=a(pivot,k)
        a(pivot,k)=temp
    end do
end if
```

그리고 그 값을 현재 pivot 값을 row값으로 가지는 부분과 교환한다. 이를 r_{max} 행과 pivot 행에 있는 모든 성분에 대해 실시한다.

```
do row=pivot+1, n
    factor=a(row,pivot)/a(pivot,pivot)
    do col=1, n2
        a(row,col)=a(row,col)-a(pivot,col)*factor
    end do
end do
```

$pivot$ 값에 해당하는 column의 $pivot$ 열에 절대 값이 최대인 성분이 옮겨졌으면 elimination을 진행한다 elimination은 각 row의 값을 $pivot$ 에 있는 값으로 나누어 factor를 구한 후 그 factor를 $pivot$ 행에 모든 성분에 곱하여 $pivot$ 행 밑에 있는 모든 행의 모든 성분에 빼서 구할 수 있다. 계산후에 $pivot$ 값의 column의 $pivot$ 값 밑은 모두 '0'으로 소거 될 것이다.

위와 같은 과정을

```
do pivot=1,n-1
```

를 통해 column 1부터 n-1까지 시행하면 대각 성분 밑으로 모두 0인 행렬이 얻어진다.

```
(6)soln=0
```

```
do o=n,1,-1
```

```
do p=n2,o,-1
```

```
a(o,n2)=a(o,n2)-soln(p)*a(o,p)
```

```
end do
```

```
soln(o)=a(o,n2)/a(o,o)
```

```
end do
```

soln행렬을 모두 0으로 초기화 한 후 $Ax=b$ 방정식의 해인 x 값을 soln에 저장할 것이다. 먼저 back substitution을 할 것이며 여기서 o 값은 행을 의미하고 p 값은 그 행에서 $n2$ 열부터 o 열까지 역으로 loop문을 쓸 것을 의미한다. $a(o,n2)$ 는 elimination 후 우변값을 의미하며 행렬 A 값의 대각성분 위부분을 모두 0으로 만들도록 하면 b 의 성분값은 자연스럽게 soln 성분에서 행 o 에 pivot을 제외한 성분을 곱하여 빼준것과 같다. 이 과정을 지나면 행렬은 대각성분만이 남게 되고 각각의 대각성분 $a(o,o)$ 로 우변성분이 $a(o,n2)$ 를 나누어주면 $soln(o)$ 즉 해가 나오게 된다 이값을 soln에 입력한다.이

```
(7)do i=1,n
```

```
print *,soln(i)
```

```
end do
```

모든 해를 출력한다. n 값이 행의 수와 일치하므로 모든 해가 출력된다.

(3)수 계산과 비교

수 계산은 뒤쪽에 첨부 하였고 값은 프로그래밍으로 계산한것과 동일하게 나온다.

과제6-2

1)코드와 실행 결과

The image shows two screenshots of a Fortran program named 'gauss_el1' running in the Plato IDE. The first screenshot displays the initial code, which sets up a 50x50 matrix and asks for its dimensions. The second screenshot shows the code with elimination and back substitution logic, and a window displaying the solutions for a 4x7 augmented matrix.

First Screenshot: Initial Code and Input

```

1 program gauss_el1
2 implicit none
3 real :: a(50,50), c, temp, soln(50), factor
4 integer :: pivot, s, n, n2, k, p, o, i, j, row, col, a_i, a_j, xmax
5
6 print *, "How much is Matrix's size? i,j"
7 read *, a_i, a_j
8 n=a_i
9 n2=a_j
10 do p=1,a_i
11 do o=1,a_j
12 print *, "a<<< ", p, ", ", o, ">>> = ?"
13 read *, c
14 a(p,o)=c
15 end do
16 end do
17
18 do pivot=1,n-1
19 xmax=pivot
20 do row=pivot,a_i
21 if (ABS(a(xmax,pivot)) < ABS(a(row,pivot))) then
22 xmax=row
23 end if
24 end do
25 if (xmax /= pivot) then
26 do k=1,a_j
27 temp=a(xmax,k)
28 a(xmax,k)=a(pivot,k)
29 a(pivot,k)=temp
30 end do
31 end if

```

Second Screenshot: Elimination and Back Substitution Code

```

32 !elimination
33 do row=pivot+1,n
34 factor=a(row,pivot)/a(pivot,pivot)
35 do col=1,n2
36 a(row,col)=a(row,col)-a(pivot,col)*factor
37 end do
38 end do
39
40 !back substitution
41 do i=1,n2-n
42 do k=1,n
43 a(k,n+1)=a(k,n+1)
44 end do
45
46 soln=0
47 do o=n,1,-1
48 do p=n+1,o,-1
49 a(o,n+1)=a(o,n+1)-soln(p)*a(o,p)
50 end do
51 soln(o)=a(o,n+1)/a(o,o)
52 end do
53 print *, "1, 'th solution"
54 do j=1,n
55 print *, soln(j)
56 end do
57
58 end program gauss_el1

```

Third Screenshot: Output Window

How much is Matrix's size? i,j
4 7
a<<< 1. 1>>> = ?
a<<< 1. 2>>> = ?
a<<< 1. 3>>> = ?
a<<< 1. 4>>> = ?
a<<< 1. 5>>> = ?
a<<< 1. 6>>> = ?
a<<< 1. 7>>> = ?
a<<< 2. 1>>> = ?
a<<< 2. 2>>> = ?
a<<< 2. 3>>> = ?
a<<< 2. 4>>> = ?
a<<< 2. 5>>> = ?

Fourth Screenshot: Output Window

1th solution
0.136364
-0.113636
0.500000
0.159091
2th solution
-0.590909
-1.34091
4.50000
3.47227
3th solution
0.272727
0.772727
-1.00000
-0.681818
Press RETURN to close window...

(2)변형된 코드 부분

먼저 대각 성분 아랫부분의 elimination 은 augmented matrix를 4*7로 만들고 그대로 코드를 진행하면 elimination 이 진행된다.

바뀐 부분은 이후에 back substitution 부분 이다.

```
do i=1,n2-n
```

먼저 구하려는 경우가 $n2-n$ 개이므로 do l=1,n2-n까지 범위로 한다.

```
do k=1,n
```

```
  a(k,n+1)=a(k,n+i)
```

```
end do
```

$Ax=b$ 에서 b 값이 $n+i$ 열에 있으므로 그 열에 있는 성분을 3번의 시행마다 각각 $n+1$ 열로 옮겨
와야한다.

```
soln=0
```

```
do o=n,1,-1
```

```
  do p=n+1,o,-1
```

```
    a(o,n+1)=a(o,n+1)-soln(p)*a(o,p)
```

```
  end do
```

```
  soln(o)=a(o,n+1)/a(o,o)
```

```
end do
```

soln=0으로 초기화 한 후 o 변수는 행을 역으로 지정하고 p변수는 $n+1$ 열부터 1열까지 열을
역으로 지정한다. 앞에 과정에서 $n+1$ 열에 우변의 값을 모두 옮겨왔으므로 이전에 $a(o,n2)$ 를
 $a(o,n+1)$ 로만 바꾸어주면 나머지 과정은 동일하게 진행된다.

```
print *,i,'th solution'
```

```
do j=1,n
```

```
  print *,soln(j)
```

```
end do
```

```
end do
```

구하려는 방정식의 해는 각각 print *, soln(j)로 출력되며 이 경우에는 soln이 4개씩 나올것
이며 서로 다른 3개의 방정식에서 각각 진행되어 출력될 것이다.

[illegible]