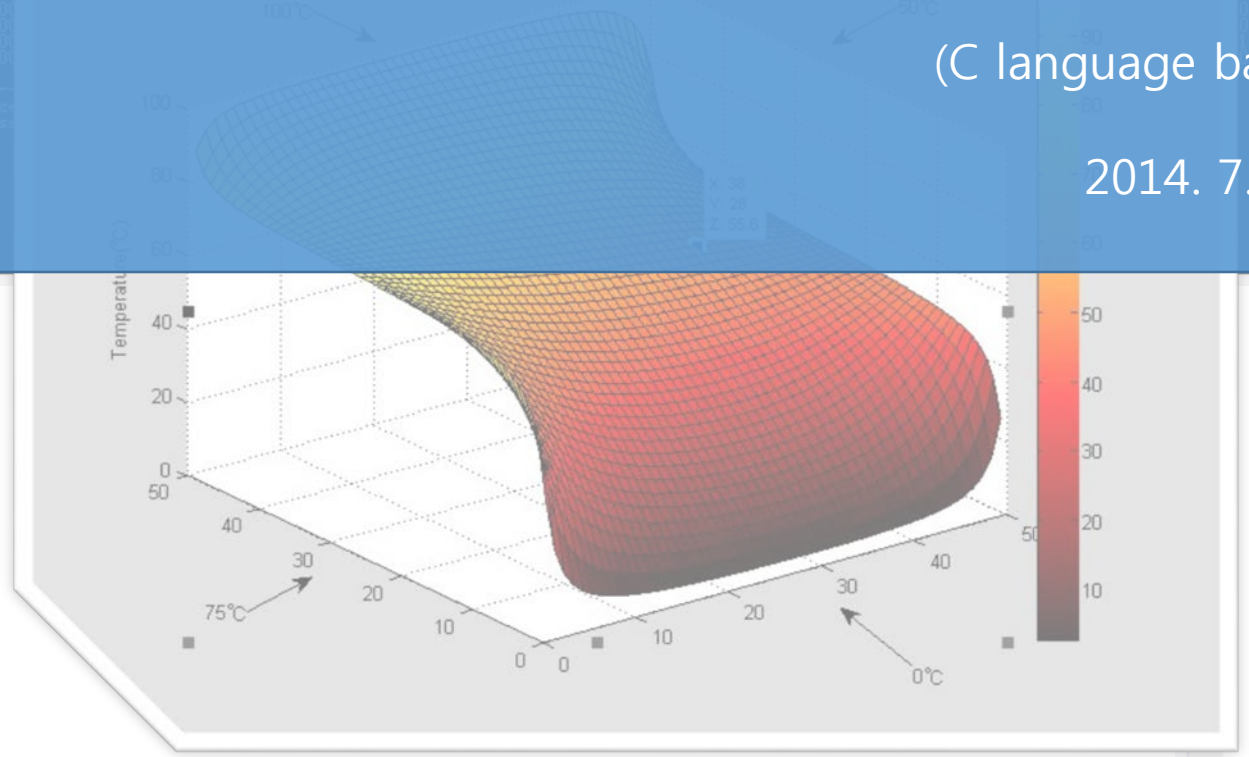
The Solution of an Elliptic PDE by using Finite Difference Method

(C language based programming)

2014. 7. 20. ~ 8. 20.




```
enter the number of points where you'll make equation:
5
enter the initial value of A's ele.:
0
enter the initial value of T's ele.:
25
enter the initial value of b's ele.:
0
enter the boudary temperature:
(left,right,upper,lower)
25
50
100
0
values of A, b are substituted well!
[A|b] is initialized as
 4.00  -1.00   0.00   0.00   0.00  -1.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00 | 75.00
-1.00   4.00  -1.00   0.00   0.00   0.00  -1.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00 |  0.00
 0.00  -1.00   4.00  -1.00   0.00   0.00   0.00  -1.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00 |  0.00
 0.00   0.00  -1.00   4.00  -1.00   0.00   0.00   0.00  -1.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00 |  0.00
 0.00   0.00   0.00  -1.00   4.00   0.00   0.00   0.00   0.00  -1.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00 | 50.00
-1.00   0.00   0.00   0.00   0.00   4.00  -1.00   0.00   0.00   0.00  -1.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00 | 75.00
 0.00  -1.00   0.00   0.00   0.00  -1.00   4.00  -1.00   0.00   0.00   0.00  -1.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00 |  0.00
 0.00   0.00  -1.00   0.00   0.00   0.00  -1.00   4.00  -1.00   0.00   0.00   0.00  -1.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00 |  0.00
 0.00   0.00   0.00  -1.00   0.00   0.00   0.00  -1.00   4.00  -1.00   0.00   0.00   0.00  -1.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00 |  0.00
 0.00   0.00   0.00   0.00  -1.00   0.00   0.00   0.00  -1.00   4.00  -1.00   0.00   0.00   0.00  -1.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00 |  0.00
 0.00   0.00   0.00   0.00   0.00  -1.00   0.00   0.00   0.00  -1.00   4.00  -1.00   0.00   0.00   0.00  -1.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00 |  0.00
 0.00   0.00   0.00   0.00   0.00   0.00  -1.00   0.00   0.00   0.00  -1.00   4.00  -1.00   0.00   0.00   0.00  -1.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00 |  0.00
 0.00   0.00   0.00   0.00   0.00   0.00   0.00  -1.00   0.00   0.00   0.00  -1.00   4.00  -1.00   0.00   0.00   0.00  -1.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00 |  0.00
 0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00  -1.00   0.00   0.00   0.00  -1.00   4.00  -1.00   0.00   0.00   0.00  -1.00   0.00   0.00   0.00   0.00   0.00   0.00 |  0.00
 0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00  -1.00   0.00   0.00   0.00  -1.00   4.00  -1.00   0.00   0.00   0.00  -1.00   0.00   0.00   0.00   0.00   0.00 |  0.00
 0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00  -1.00   0.00   0.00   0.00  -1.00   4.00  -1.00   0.00   0.00   0.00  -1.00   0.00   0.00   0.00   0.00 |  0.00
 0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00  -1.00   0.00   0.00   0.00  -1.00   4.00  -1.00   0.00   0.00   0.00  -1.00   0.00   0.00   0.00 |  0.00
 0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00  -1.00   0.00   0.00   0.00  -1.00   4.00  -1.00   0.00   0.00   0.00  -1.00   0.00   0.00 |  0.00
 0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00  -1.00   0.00   0.00   0.00  -1.00   4.00  -1.00   0.00   0.00   0.00  -1.00   0.00 |  0.00
 0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00  -1.00   0.00   0.00   0.00  -1.00   4.00  -1.00   0.00   0.00   0.00  -1.00 |  0.00
 0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00  -1.00   0.00   0.00   0.00  -1.00   4.00  -1.00   0.00   0.00   0.00 |  0.00
 0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00  -1.00   0.00   0.00   0.00  -1.00   4.00  -1.00   0.00   0.00 |  0.00
 0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00  -1.00   0.00   0.00   0.00  -1.00   4.00  -1.00   0.00 |  0.00
 0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00  -1.00   0.00   0.00   0.00  -1.00   4.00  -1.00 |  0.00
 0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00  -1.00   0.00   0.00   0.00  -1.00   4.00 |  0.00
```

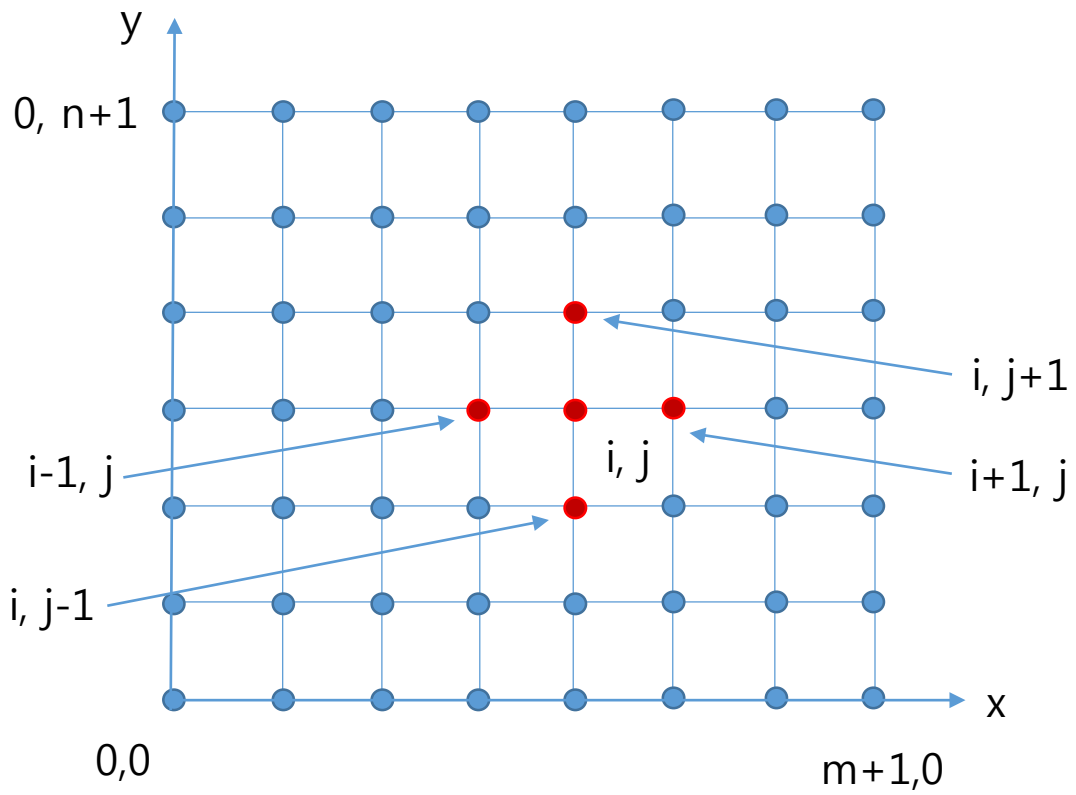
```
93              A[(i-1)*N+j-1][i*N+j-1]=-1;//right
94              //A[(i-1)*N+j-1][(i-2)*N+j-1]=-1;//left
95              //A[(i-1)*N+j-1][(i-1)*N+j]=-1;//upper
96              A[(i-1)*N+j-1][(i-1)*N+j-2]=-1;//lower
97              A[(i-1)*N+j-1][(i-1)*N+j-1]=4;//middle
98              b[(j-1)*N+i-1]=Tleft+Tupper;
99          }
100         else if(i==1 && j==1)//case3
101         {
102             A[(i-1)*N+j-1][i*N+j-1]=-1;//right
103             //A[(i-1)*N+j-1][(i-2)*N+j-1]=-1;//left
104             A[(i-1)*N+j-1][(i-1)*N+j]=-1;//upper
105             //A[(i-1)*N+j-1][(i-1)*N+j-2]=-1;//lower
106             A[(i-1)*N+j-1][(i-1)*N+j-1]=4;//middle
107             b[(j-1)*N+i-1]=Tleft+Tlower;
108         }
109         else if(i==N && j==N)//case4
110         {
```

$$\nabla^2 T = 0$$

in 2 dimensional space(T : temperature)

$$\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} = 0$$

Laplace equation(Elliptic PDE)



When boundary conditions are given, this problem describes the distribution of heated plate's temperature.

by using finite difference expression,

$$\frac{\partial^2 T}{\partial x^2} = \frac{T_{i+1,j} - 2T_{i,j} + T_{i-1,j}}{\Delta x^2}$$

$$\frac{\partial^2 T}{\partial y^2} = \frac{T_{i,j+1} - 2T_{i,j} + T_{i,j-1}}{\Delta y^2}$$

then the PDE is written as

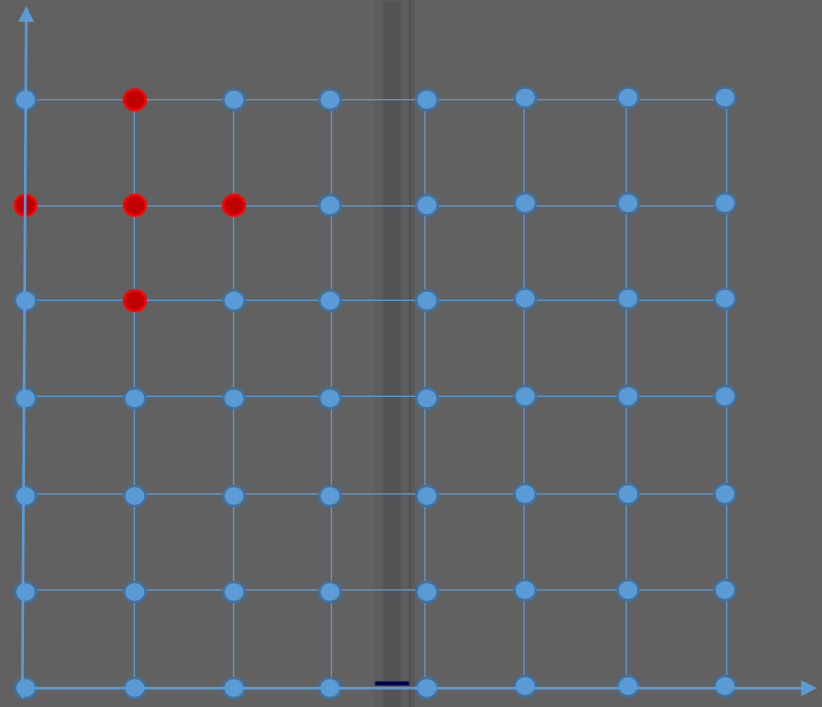
$$\frac{T_{i+1,j} - 2T_{i,j} + T_{i-1,j}}{\Delta x^2} + \frac{T_{i,j+1} - 2T_{i,j} + T_{i,j-1}}{\Delta y^2} = 0$$

$$\boldsymbol{T_{i+1,j} + T_{i-1,j} + T_{i,j+1} + T_{i,j-1} - 4T_{i,j} = 0} \quad \text{when} \quad \Delta x = \Delta y$$

Laplacian difference equation

```
71
72      /*confirming that the dynamic allocation is well performed*/
73      print2Darray(A,N*N,N*N);//this function receives the dynamically allocated array and show it.
74      print2Darray(T,N,N);
75      print1Darray(b,N*N);
76
77      /*T[i+1][j]+T[i-1][j]+T[i][j+1]+T[i][j-1] - 4*T[i][j] == 0*/
78      for(i=1;i<N+1;i++)
79      {
80          for(j=1;j<N+1;j++)
81          {
82              if(i!=1 && i!=N && j!=1 && j!=N)//case1
83              {
84                  A[(i-1)*N+j-1][i*N+j-1]=-1;//right
85                  A[(i-1)*N+j-1][(i-2)*N+j-1]=-1;//left
86                  A[(i-1)*N+j-1][(i-1)*N+j]=-1;//upper
87                  A[(i-1)*N+j-1][(i-1)*N+j-2]=-1;//lower
88                  A[(i-1)*N+j-1][(i-1)*N+j-1]=4;//middle
89                  b[(j-1)*N+i-1]=0;//research needed on the reason why b's index is reversed.
90              }
91              else if(i==1 && j==N)//case2
92              {
93                  A[(i-1)*N+j-1][i*N+j-1]=-1;//right
94                  //A[(i-1)*N+j-1][(i-2)*N+j-1]=-1;//left
95                  //A[(i-1)*N+j-1][(i-1)*N+j]=-1;//upper
96                  A[(i-1)*N+j-1][(i-1)*N+j-2]=-1;//lower
97                  A[(i-1)*N+j-1][(i-1)*N+j-1]=4;//middle
98                  b[(j-1)*N+i-1]=Tleft+Tupper;
99              }
100             else if(i==1 && j==1)//case3
101             {
102                 A[(i-1)*N+j-1][i*N+j-1]=-1;//right
103                 //A[(i-1)*N+j-1][(i-2)*N+j-1]=-1;//left
104                 A[(i-1)*N+j-1][(i-1)*N+j]=-1;//upper
105                 //A[(i-1)*N+j-1][(i-1)*N+j-2]=-1;//lower
106                 A[(i-1)*N+j-1][(i-1)*N+j-1]=4;//middle
107                 b[(j-1)*N+i-1]=Tleft+Tlower;
108             }
109             else if(i==N && j==N)//case4
110             {
111                 //A[(i-1)*N+j-1][i*N+j-1]=-1;//right
112                 A[(i-1)*N+j-1][(i-2)*N+j-1]=-1;//left
113                 //A[(i-1)*N+j-1][(i-1)*N+j]=-1;//upper
114                 A[(i-1)*N+j-1][(i-1)*N+j-2]=-1;//lower
115                 A[(i-1)*N+j-1][(i-1)*N+j-1]=4;//middle
```

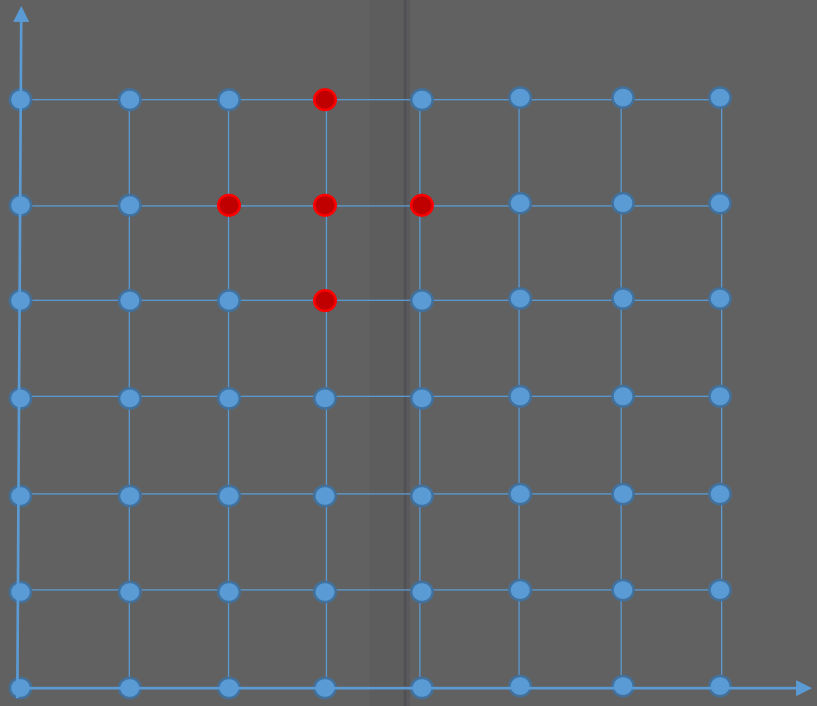
Categorize the situation of (i, j) as 9 cases



```
129             A[(i-1)*N+j-1][i*N+j-1]=-1;//right
130             //A[(i-1)*N+j-1][(i-2)*N+j-1]=-1;//left
131             A[(i-1)*N+j-1][(i-1)*N+j]=-1;//upper
132             A[(i-1)*N+j-1][(i-1)*N+j-2]=-1;//lower
133             A[(i-1)*N+j-1][(i-1)*N+j-1]=4;//middle
134             b[(j-1)*N+i-1]=Tleft;
135         }
136         else if(i==N && j!=N && j!=1)//case7
137         {
138             //A[(i-1)*N+j-1][i*N+j-1]=-1;//right
139             A[(i-1)*N+j-1][(i-2)*N+j-1]=-1;//left
140             A[(i-1)*N+j-1][(i-1)*N+j]=-1;//upper
141             A[(i-1)*N+j-1][(i-1)*N+j-2]=-1;//lower
142             A[(i-1)*N+j-1][(i-1)*N+j-1]=4;//middle
143             b[(j-1)*N+i-1]=Tright;
144         }
145         else if(j==N && i!=1 && i!=N)//case8
146         {
147             A[(i-1)*N+j-1][i*N+j-1]=-1;//right
148             A[(i-1)*N+j-1][(i-2)*N+j-1]=-1;//left
149             //A[(i-1)*N+j-1][(i-1)*N+j]=-1;//upper
150             A[(i-1)*N+j-1][(i-1)*N+j-2]=-1;//lower
151             A[(i-1)*N+j-1][(i-1)*N+j-1]=4;//middle
152             b[(j-1)*N+i-1]=Tupper;
153         }
154         else if(i==1 && j!=1 && j!=N)//case9
155         {
156             A[(i-1)*N+j-1][i*N+j-1]=-1;//right
157             A[(i-1)*N+j-1][(i-2)*N+j-1]=-1;//left
158             A[(i-1)*N+j-1][(i-1)*N+j]=-1;//upper
159             //A[(i-1)*N+j-1][(i-1)*N+j-2]=-1;//lower
160             A[(i-1)*N+j-1][(i-1)*N+j-1]=4;//middle
161             b[(j-1)*N+i-1]=Tlower;
162         }
163         else
164         printf("error%d%d:none of cases chosen!\n",i,j);
165     }
166 }
167 printf("values of A, b are substituted well!\n");
168
169 /*confirming that substitution is well performed*/
170 printf("[A|b] is initialized as\n");
```

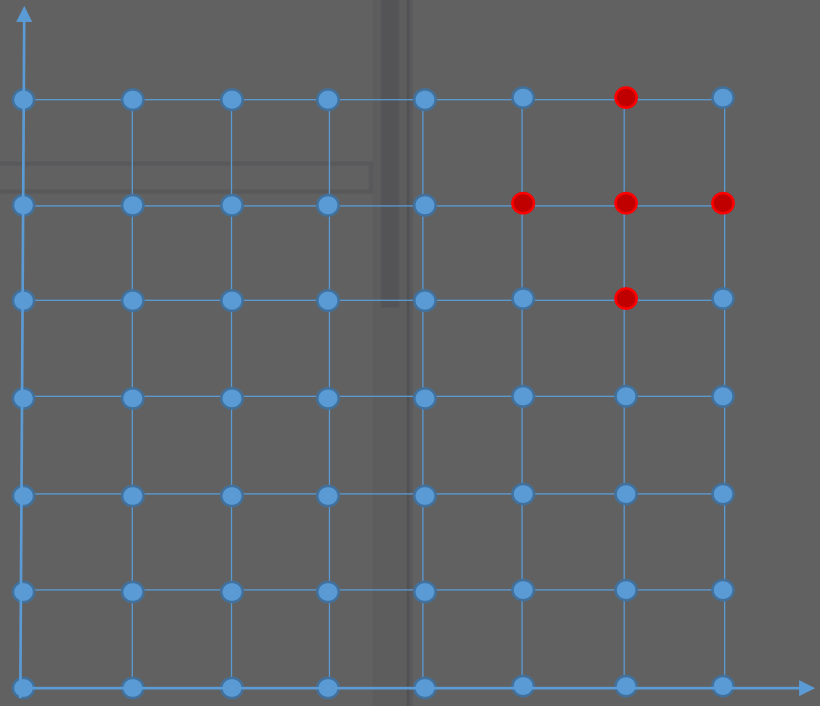
100 %

Categorize the situation of (i, j) as 9 cases



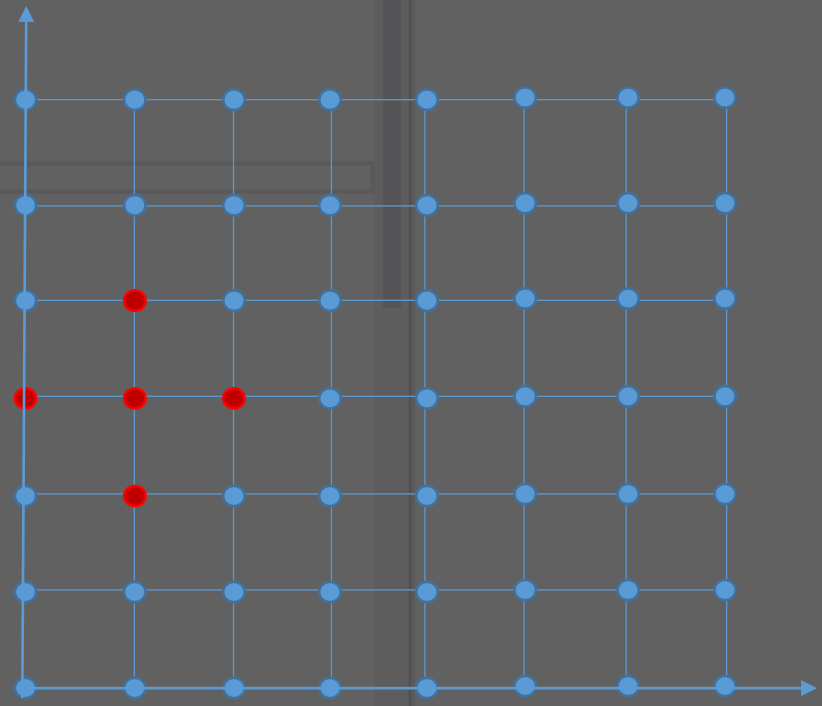
```
101         {
102             A[(i-1)*N+j-1][i*N+j-1]=-1;//right
103             //A[(i-1)*N+j-1][(i-2)*N+j-1]=-1;//left
104             A[(i-1)*N+j-1][(i-1)*N+j]=-1;//upper
105             //A[(i-1)*N+j-1][(i-1)*N+j-2]=-1;//lower
106             A[(i-1)*N+j-1][(i-1)*N+j-1]=4;//middle
107             b[(j-1)*N+i-1]=Tleft+Tlower;
108         }
109         else if(i==N && j==N)//case4
110         {
111             //A[(i-1)*N+j-1][i*N+j-1]=-1;//right
112             A[(i-1)*N+j-1][(i-2)*N+j-1]=-1;//left
113             //A[(i-1)*N+j-1][(i-1)*N+j]=-1;//upper
114             A[(i-1)*N+j-1][(i-1)*N+j-2]=-1;//lower
115             A[(i-1)*N+j-1][(i-1)*N+j-1]=4;//middle
116             b[(j-1)*N+i-1]=Tupper+Tright;
117         }
118         else if(i==N && j==1)//case5
119         {
120             //A[(i-1)*N+j-1][i*N+j-1]=-1;//right
121             A[(i-1)*N+j-1][(i-2)*N+j-1]=-1;//left
122             A[(i-1)*N+j-1][(i-1)*N+j]=-1;//upper
123             //A[(i-1)*N+j-1][(i-1)*N+j-2]=-1;//lower
124             A[(i-1)*N+j-1][(i-1)*N+j-1]=4;//middle
125             b[(j-1)*N+i-1]=Tright+Tlower;
126         }
127         else if(i==1 && j!=N && j!=1)//case6
128         {
129             A[(i-1)*N+j-1][i*N+j-1]=-1;//right
130             //A[(i-1)*N+j-1][(i-2)*N+j-1]=-1;//left
131             A[(i-1)*N+j-1][(i-1)*N+j]=-1;//upper
132             A[(i-1)*N+j-1][(i-1)*N+j-2]=-1;//lower
133             A[(i-1)*N+j-1][(i-1)*N+j-1]=4;//middle
134             b[(j-1)*N+i-1]=Tleft;
135         }
136         else if(i==N && j!=N && j!=1)//case7
137         {
138             //A[(i-1)*N+j-1][i*N+j-1]=-1;//right
139             A[(i-1)*N+j-1][(i-2)*N+j-1]=-1;//left
140             A[(i-1)*N+j-1][(i-1)*N+j]=-1;//upper
141             A[(i-1)*N+j-1][(i-1)*N+j-2]=-1;//lower
142             A[(i-1)*N+j-1][(i-1)*N+j-1]=4;//middle
143             b[(j-1)*N+i-1]=Tright;
144         }
145         else if(j==N && i!=1 && i!=N)//case8
```

Categorize the situation of (i, j) as 9 cases



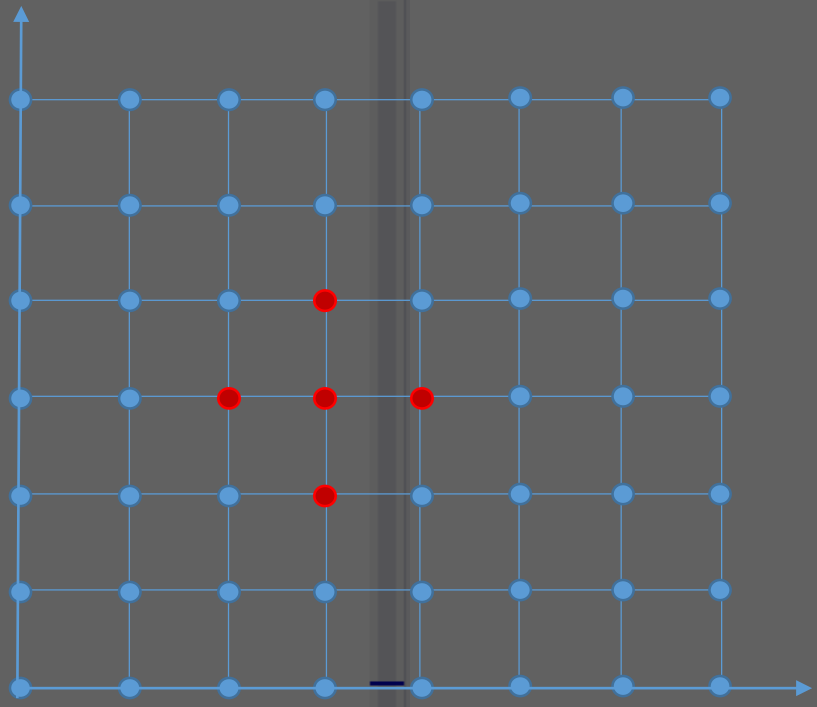
```
101         {
102             A[(i-1)*N+j-1][i*N+j-1]=-1;//right
103             //A[(i-1)*N+j-1][(i-2)*N+j-1]=-1;//left
104             A[(i-1)*N+j-1][(i-1)*N+j]=-1;//upper
105             //A[(i-1)*N+j-1][(i-1)*N+j-2]=-1;//lower
106             A[(i-1)*N+j-1][(i-1)*N+j-1]=4;//middle
107             b[(j-1)*N+i-1]=Tleft+Tlower;
108         }
109         else if(i==N && j==N)//case4
110         {
111             //A[(i-1)*N+j-1][i*N+j-1]=-1;//right
112             A[(i-1)*N+j-1][(i-2)*N+j-1]=-1;//left
113             //A[(i-1)*N+j-1][(i-1)*N+j]=-1;//upper
114             A[(i-1)*N+j-1][(i-1)*N+j-2]=-1;//lower
115             A[(i-1)*N+j-1][(i-1)*N+j-1]=4;//middle
116             b[(j-1)*N+i-1]=Tupper+Tright;
117         }
118         else if(i==N && j==1)//case5
119         {
120             //A[(i-1)*N+j-1][i*N+j-1]=-1;//right
121             A[(i-1)*N+j-1][(i-2)*N+j-1]=-1;//left
122             A[(i-1)*N+j-1][(i-1)*N+j]=-1;//upper
123             //A[(i-1)*N+j-1][(i-1)*N+j-2]=-1;//lower
124             A[(i-1)*N+j-1][(i-1)*N+j-1]=4;//middle
125             b[(j-1)*N+i-1]=Tright+Tlower;
126         }
127         else if(i==1 && j!=N && j!=1)//case6
128         {
129             A[(i-1)*N+j-1][i*N+j-1]=-1;//right
130             //A[(i-1)*N+j-1][(i-2)*N+j-1]=-1;//left
131             A[(i-1)*N+j-1][(i-1)*N+j]=-1;//upper
132             A[(i-1)*N+j-1][(i-1)*N+j-2]=-1;//lower
133             A[(i-1)*N+j-1][(i-1)*N+j-1]=4;//middle
134             b[(j-1)*N+i-1]=Tleft;
135         }
136         else if(i==N && j!=N && j!=1)//case7
137         {
138             //A[(i-1)*N+j-1][i*N+j-1]=-1;//right
139             A[(i-1)*N+j-1][(i-2)*N+j-1]=-1;//left
140             A[(i-1)*N+j-1][(i-1)*N+j]=-1;//upper
141             A[(i-1)*N+j-1][(i-1)*N+j-2]=-1;//lower
142             A[(i-1)*N+j-1][(i-1)*N+j-1]=4;//middle
143             b[(j-1)*N+i-1]=Tright;
144         }
145         else if(j==N && i!=1 && i!=N)//case8
```

Categorize the situation of (i, j) as 9 cases



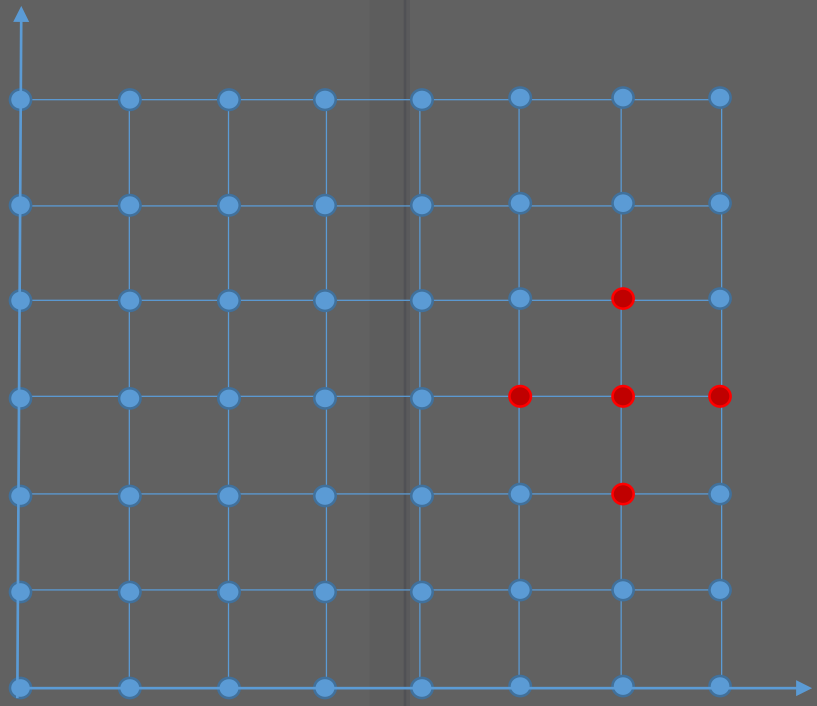
```
71
72      /*confirming that the dynamic allocation is well performed*/
73      print2Darray(A,N*N,N*N);//this function receives the dynamically allocated array and show it.
74      print2Darray(T,N,N);
75      print1Darray(b,N*N);
76
77      /*T[i+1][j]+T[i-1][j]+T[i][j+1]+T[i][j-1] - 4*T[i][j] == 0*/
78      for(i=1;i<N+1;i++)
79      {
80          for(j=1;j<N+1;j++)
81          {
82              if(i!=1 && i!=N && j!=1 && j!=N)//case1
83              {
84                  A[(i-1)*N+j-1][i*N+j-1]=-1;//right
85                  A[(i-1)*N+j-1][(i-2)*N+j-1]=-1;//left
86                  A[(i-1)*N+j-1][(i-1)*N+j]=-1;//upper
87                  A[(i-1)*N+j-1][(i-1)*N+j-2]=-1;//lower
88                  A[(i-1)*N+j-1][(i-1)*N+j-1]=4;//middle
89                  b[(j-1)*N+i-1]=0;//research needed on the reason why b's index is reversed.
90              }
91              else if(i==1 && j==N)//case2
92              {
93                  A[(i-1)*N+j-1][i*N+j-1]=-1;//right
94                  //A[(i-1)*N+j-1][(i-2)*N+j-1]=-1;//left
95                  //A[(i-1)*N+j-1][(i-1)*N+j]=-1;//upper
96                  A[(i-1)*N+j-1][(i-1)*N+j-2]=-1;//lower
97                  A[(i-1)*N+j-1][(i-1)*N+j-1]=4;//middle
98                  b[(j-1)*N+i-1]=Tleft+Tupper;
99              }
100             else if(i==1 && j==1)//case3
101             {
102                 A[(i-1)*N+j-1][i*N+j-1]=-1;//right
103                 //A[(i-1)*N+j-1][(i-2)*N+j-1]=-1;//left
104                 A[(i-1)*N+j-1][(i-1)*N+j]=-1;//upper
105                 //A[(i-1)*N+j-1][(i-1)*N+j-2]=-1;//lower
106                 A[(i-1)*N+j-1][(i-1)*N+j-1]=4;//middle
107                 b[(j-1)*N+i-1]=Tleft+Tlower;
108             }
109             else if(i==N && j==N)//case4
110             {
111                 //A[(i-1)*N+j-1][i*N+j-1]=-1;//right
112                 A[(i-1)*N+j-1][(i-2)*N+j-1]=-1;//left
113                 //A[(i-1)*N+j-1][(i-1)*N+j]=-1;//upper
114                 A[(i-1)*N+j-1][(i-1)*N+j-2]=-1;//lower
115                 A[(i-1)*N+j-1][(i-1)*N+j-1]=4;//middle
```

Categorize the situation of (i, j) as 9 cases



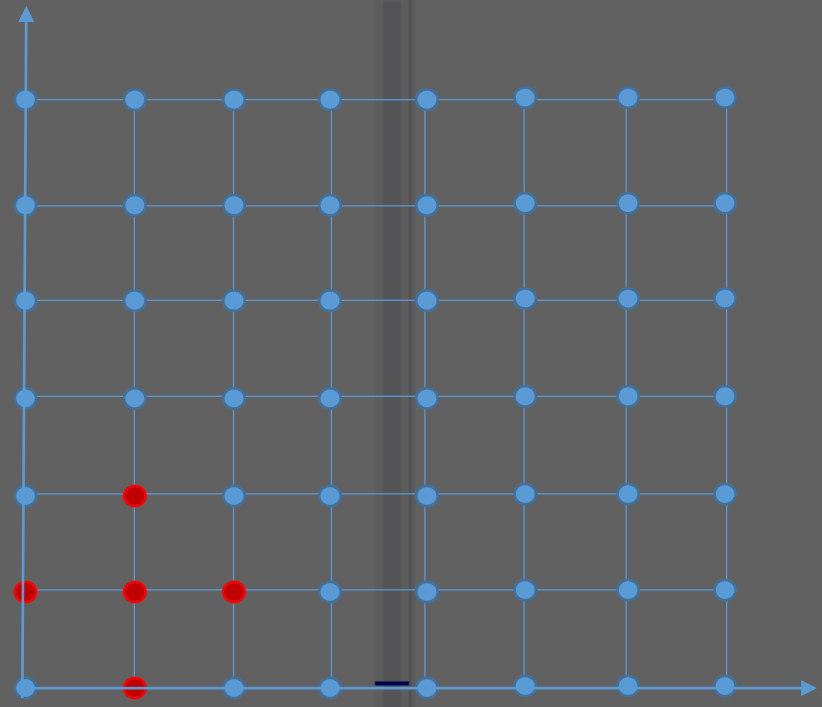
```
            {
                A[(i-1)*N+j-1][i*N+j-1]=-1;//right
                //A[(i-1)*N+j-1][(i-2)*N+j-1]=-1;//left
                A[(i-1)*N+j-1][(i-1)*N+j]=-1;//upper
                A[(i-1)*N+j-1][(i-1)*N+j-2]=-1;//lower
                A[(i-1)*N+j-1][(i-1)*N+j-1]=4;//middle
                b[(j-1)*N+i-1]=Tleft;
            }
            else if(i==N && j!=N && j!=1)//case7
            {
                //A[(i-1)*N+j-1][i*N+j-1]=-1;//right
                A[(i-1)*N+j-1][(i-2)*N+j-1]=-1;//left
                A[(i-1)*N+j-1][(i-1)*N+j]=-1;//upper
                A[(i-1)*N+j-1][(i-1)*N+j-2]=-1;//lower
                A[(i-1)*N+j-1][(i-1)*N+j-1]=4;//middle
                b[(j-1)*N+i-1]=Tright;
            }
            else if(j==N && i!=1 && i!=N)//case8
            {
                A[(i-1)*N+j-1][i*N+j-1]=-1;//right
                A[(i-1)*N+j-1][(i-2)*N+j-1]=-1;//left
                //A[(i-1)*N+j-1][(i-1)*N+j]=-1;//upper
                A[(i-1)*N+j-1][(i-1)*N+j-2]=-1;//lower
                A[(i-1)*N+j-1][(i-1)*N+j-1]=4;//middle
                b[(j-1)*N+i-1]=Tupper;
            }
            else if(j==1 && i!=1 && i!=N)//case9
            {
                A[(i-1)*N+j-1][i*N+j-1]=-1;//right
                A[(i-1)*N+j-1][(i-2)*N+j-1]=-1;//left
                A[(i-1)*N+j-1][(i-1)*N+j]=-1;//upper
                //A[(i-1)*N+j-1][(i-1)*N+j-2]=-1;//lower
                A[(i-1)*N+j-1][(i-1)*N+j-1]=4;//middle
                b[(j-1)*N+i-1]=Tlower;
            }
            else
            printf("error%d%d:none of cases chosen!\n",i,j);
        }
    }
    printf("values of A, b are substituted well!\n");

    /*confirming that substitution is well performed*/
    printf("[A|b] is initialized as\n");
```

Categorize the situation of (i, j) as 9 cases



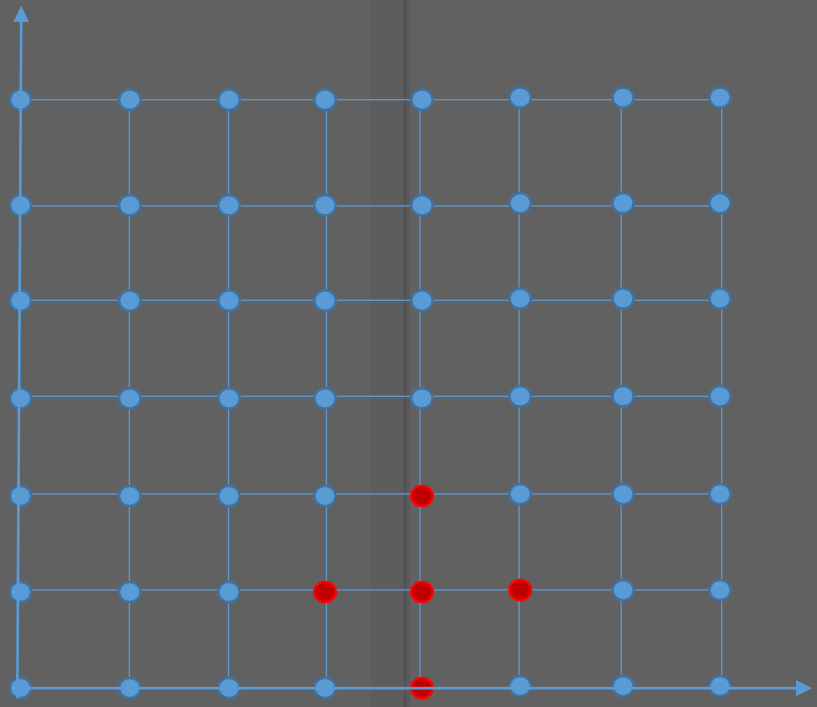
```
71
72      /*confirming that the dynamic allocation is well performed*/
73      print2Darray(A,N*N,N*N);//this function receives the dynamically allocated array and show it.
74      print2Darray(T,N,N);
75      print1Darray(b,N*N);
76
77      /*T[i+1][j]+T[i-1][j]+T[i][j+1]+T[i][j-1] - 4*T[i][j] == 0*/
78      for(i=1;i<N+1;i++)
79      {
80          for(j=1;j<N+1;j++)
81          {
82              if(i!=1 && i!=N && j!=1 && j!=N)//case1
83              {
84                  A[(i-1)*N+j-1][i*N+j-1]=-1;//right
85                  A[(i-1)*N+j-1][(i-2)*N+j-1]=-1;//left
86                  A[(i-1)*N+j-1][(i-1)*N+j]=-1;//upper
87                  A[(i-1)*N+j-1][(i-1)*N+j-2]=-1;//lower
88                  A[(i-1)*N+j-1][(i-1)*N+j-1]=4;//middle
89                  b[(j-1)*N+i-1]=0;//research needed on the reason why b's index is reversed.
90              }
91              else if(i==1 && j==N)//case2
92              {
93                  A[(i-1)*N+j-1][i*N+j-1]=-1;//right
94                  //A[(i-1)*N+j-1][(i-2)*N+j-1]=-1;//left
95                  //A[(i-1)*N+j-1][(i-1)*N+j]=-1;//upper
96                  A[(i-1)*N+j-1][(i-1)*N+j-2]=-1;//lower
97                  A[(i-1)*N+j-1][(i-1)*N+j-1]=4;//middle
98                  b[(j-1)*N+i-1]=Tleft+Tupper;
99              }
100             else if(i==1 && j==1)//case3
101             {
102                 A[(i-1)*N+j-1][i*N+j-1]=-1;//right
103                 //A[(i-1)*N+j-1][(i-2)*N+j-1]=-1;//left
104                 A[(i-1)*N+j-1][(i-1)*N+j]=-1;//upper
105                 //A[(i-1)*N+j-1][(i-1)*N+j-2]=-1;//lower
106                 A[(i-1)*N+j-1][(i-1)*N+j-1]=4;//middle
107                 b[(j-1)*N+i-1]=Tleft+Tlower;
108             }
109             else if(i==N && j==N)//case4
110             {
111                 //A[(i-1)*N+j-1][i*N+j-1]=-1;//right
112                 A[(i-1)*N+j-1][(i-2)*N+j-1]=-1;//left
113                 //A[(i-1)*N+j-1][(i-1)*N+j]=-1;//upper
114                 A[(i-1)*N+j-1][(i-1)*N+j-2]=-1;//lower
115                 A[(i-1)*N+j-1][(i-1)*N+j-1]=4;//middle
```

Categorize the situation of (i, j) as 9 cases



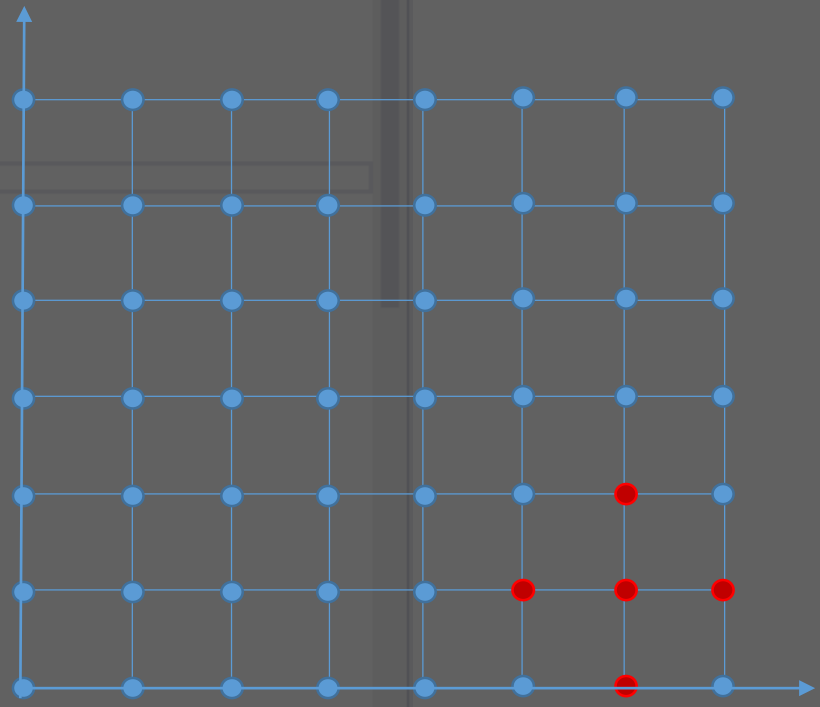
```
            {
                A[(i-1)*N+j-1][i*N+j-1]=-1;//right
                //A[(i-1)*N+j-1][(i-2)*N+j-1]=-1;//left
                A[(i-1)*N+j-1][(i-1)*N+j]=-1;//upper
                A[(i-1)*N+j-1][(i-1)*N+j-2]=-1;//lower
                A[(i-1)*N+j-1][(i-1)*N+j-1]=4;//middle
                b[(j-1)*N+i-1]=Tleft;
            }
            else if(i==N && j!=N && j!=1)//case7
            {
                //A[(i-1)*N+j-1][i*N+j-1]=-1;//right
                A[(i-1)*N+j-1][(i-2)*N+j-1]=-1;//left
                A[(i-1)*N+j-1][(i-1)*N+j]=-1;//upper
                A[(i-1)*N+j-1][(i-1)*N+j-2]=-1;//lower
                A[(i-1)*N+j-1][(i-1)*N+j-1]=4;//middle
                b[(j-1)*N+i-1]=Tright;
            }
            else if(j==N && i!=1 && i!=N)//case8
            {
                A[(i-1)*N+j-1][i*N+j-1]=-1;//right
                A[(i-1)*N+j-1][(i-2)*N+j-1]=-1;//left
                //A[(i-1)*N+j-1][(i-1)*N+j]=-1;//upper
                A[(i-1)*N+j-1][(i-1)*N+j-2]=-1;//lower
                A[(i-1)*N+j-1][(i-1)*N+j-1]=4;//middle
                b[(j-1)*N+i-1]=Tupper;
            }
            else if(j==1 && i!=1 && i!=N)//case9
            {
                A[(i-1)*N+j-1][i*N+j-1]=-1;//right
                A[(i-1)*N+j-1][(i-2)*N+j-1]=-1;//left
                A[(i-1)*N+j-1][(i-1)*N+j]=-1;//upper
                //A[(i-1)*N+j-1][(i-1)*N+j-2]=-1;//lower
                A[(i-1)*N+j-1][(i-1)*N+j-1]=4;//middle
                b[(j-1)*N+i-1]=Tlower;
            }
            else
            printf("error%d%d:none of cases chosen!\n",i,j);
        }
    }
    printf("values of A, b are substituted well!\n");

    /*confirming that substitution is well performed*/
    printf("[A|b] is initialized as\n");
```

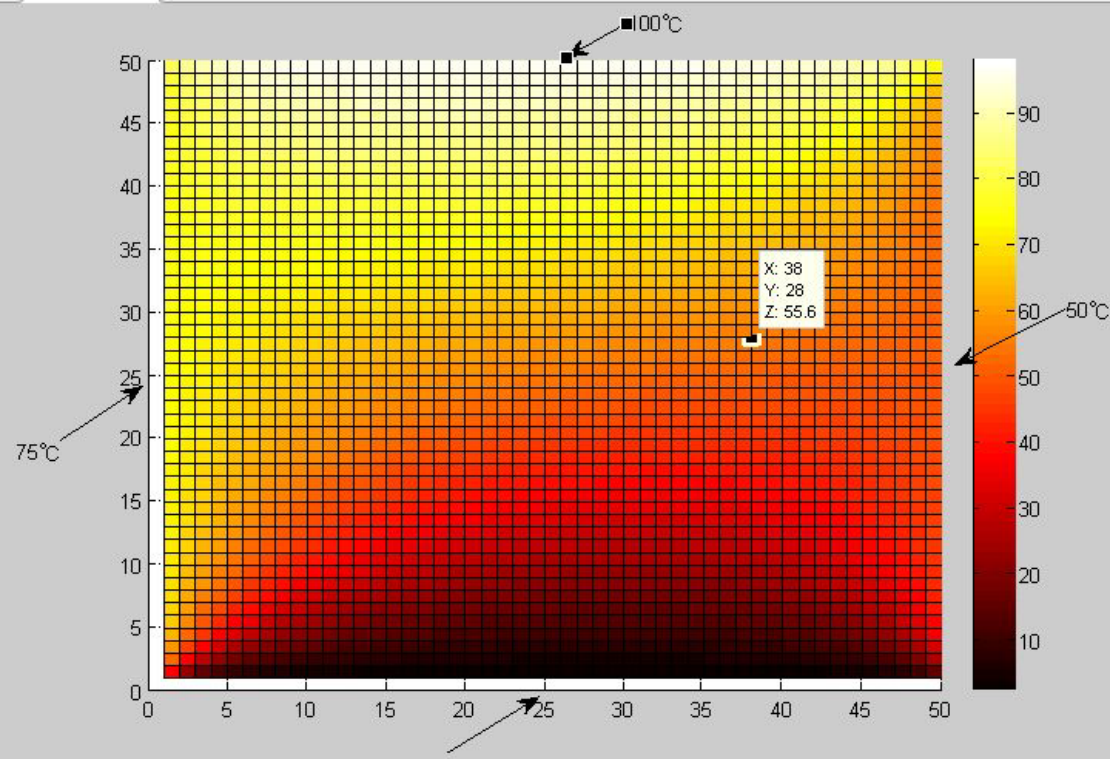
Categorize the situation of (i, j) as 9 cases



```
101         {
102             A[(i-1)*N+j-1][i*N+j-1]=-1;//right
103             //A[(i-1)*N+j-1][(i-2)*N+j-1]=-1;//left
104             A[(i-1)*N+j-1][(i-1)*N+j]=-1;//upper
105             //A[(i-1)*N+j-1][(i-1)*N+j-2]=-1;//lower
106             A[(i-1)*N+j-1][(i-1)*N+j-1]=4;//middle
107             b[(j-1)*N+i-1]=Tleft+Tlower;
108         }
109         else if(i==N && j==N)//case4
110         {
111             //A[(i-1)*N+j-1][i*N+j-1]=-1;//right
112             A[(i-1)*N+j-1][(i-2)*N+j-1]=-1;//left
113             //A[(i-1)*N+j-1][(i-1)*N+j]=-1;//upper
114             A[(i-1)*N+j-1][(i-1)*N+j-2]=-1;//lower
115             A[(i-1)*N+j-1][(i-1)*N+j-1]=4;//middle
116             b[(j-1)*N+i-1]=Tupper+Tright;
117         }
118         else if(i==N && j==1)//case5
119         {
120             //A[(i-1)*N+j-1][i*N+j-1]=-1;//right
121             A[(i-1)*N+j-1][(i-2)*N+j-1]=-1;//left
122             A[(i-1)*N+j-1][(i-1)*N+j]=-1;//upper
123             //A[(i-1)*N+j-1][(i-1)*N+j-2]=-1;//lower
124             A[(i-1)*N+j-1][(i-1)*N+j-1]=4;//middle
125             b[(j-1)*N+i-1]=Tright+Tlower;
126         }
127         else if(i==1 && j!=N && j!=1)//case6
128         {
129             A[(i-1)*N+j-1][i*N+j-1]=-1;//right
130             //A[(i-1)*N+j-1][(i-2)*N+j-1]=-1;//left
131             A[(i-1)*N+j-1][(i-1)*N+j]=-1;//upper
132             A[(i-1)*N+j-1][(i-1)*N+j-2]=-1;//lower
133             A[(i-1)*N+j-1][(i-1)*N+j-1]=4;//middle
134             b[(j-1)*N+i-1]=Tleft;
135         }
136         else if(i==N && j!=N && j!=1)//case7
137         {
138             //A[(i-1)*N+j-1][i*N+j-1]=-1;//right
139             A[(i-1)*N+j-1][(i-2)*N+j-1]=-1;//left
140             A[(i-1)*N+j-1][(i-1)*N+j]=-1;//upper
141             A[(i-1)*N+j-1][(i-1)*N+j-2]=-1;//lower
142             A[(i-1)*N+j-1][(i-1)*N+j-1]=4;//middle
143             b[(j-1)*N+i-1]=Tright;
144         }
145         else if(j==N && i!=1 && i!=N)//case8
```

Categorize the situation of (i, j) as 9 cases





With the same logic, augmented matrix for the whole Equations is

```
C:₩Users₩Dongjae₩Desktop₩C_code₩Array₩A_b_generator₩A_b_generato
enter the number of points where you'll make equation:
3
enter the initial value of A's ele.:
0
enter the initial value of T's ele.:
75
enter the initial value of b's ele.:
0
enter the boudary temperature:
(left,right,upper,lower)
75
50
100
0
values of A, b are substituted well!
[A|b] is initialized as
 4.00  -1.00   0.00  -1.00   0.00   0.00   0.00   0.00   0.00   | 75.00
-1.00   4.00  -1.00   0.00  -1.00   0.00   0.00   0.00   0.00   |  0.00
 0.00  -1.00   4.00   0.00   0.00  -1.00   0.00   0.00   0.00   | 50.00
-1.00   0.00   0.00   4.00  -1.00   0.00  -1.00   0.00   0.00   | 75.00
 0.00  -1.00   0.00  -1.00   4.00  -1.00   0.00  -1.00   0.00   |  0.00
 0.00   0.00  -1.00   0.00  -1.00   4.00   0.00   0.00  -1.00   | 50.00
 0.00   0.00   0.00  -1.00   0.00   0.00   4.00  -1.00   0.00   | 175.00
 0.00   0.00   0.00   0.00  -1.00   0.00  -1.00   4.00  -1.00   | 100.00
 0.00   0.00   0.00   0.00   0.00  -1.00   0.00  -1.00   4.00   | 150.00

--------------------------------
Process exited with return value 0
```

Ex. For 4 by 4 grid

Equation for (1,1) is

$$T_{21} + T_{01} + T_{12} + T_{10} - 4T_{11} = 0$$

and the boundary conditions are

$$T_{10} = 0, \quad T_{01} = 75$$

$$\Rightarrow -4T_{11} + T_{12} + T_{21} = -75$$

```
175                 //A[(i-1)*N+j-1][(i-1)*N+j-2]=-1;//lower
176                 A[(i-1)*N+j-1][(i-1)*N+j-1]=4;//middle
177                 b[(j-1)*N+i-1]=Tlower;
178             }
179             else
180             printf("error%d%d:none of cases chosen!\n",i,j);
181         }
182     }
183     printf("values of A, b are substituted well!\n");
184
185     /*confirming that substitution is well performed*/
186     printf("[A|b] is initialized as\n");
187     /*for(k=0;k<N*N;k++)
188     {
189         for(l=0;l<N*N;l++)
190         {
191             printf(" %6.3f",A[k][l]);
192         }
193         printf(" | %5.3f\n",b[k]);
194     }*/
195
196     /*Solving AT=b using jacobi method*/
197     for(k=0;k<100000;k++)
198     {
199         deltaT = 0;
200         for(i=1;i<N+1;i++)
201         {
202             for(j=1;j<N+1;j++)
203             {
204                 Tnew[i-1][j-1] = T[i-1][j-1] + (b[(j-1)*N+i-1] - sigma(A,T,i,j,N))/A[(i-1)*N+j-1][(i-1)*N+j-1];
205                 deltaT += fabs((Tnew[i-1][j-1] - T[i-1][j-1])/T[i-1][j-1]);
206             }
207         }
208
209         /*
210         x1new = x[0] + (B[0]-sigma(A,3,x,3,1,3))/A[0][0]; //iterative process
211         x2new = x[1] + (B[1]-sigma(A,3,x,3,2,3))/A[1][1];
212         x3new = x[2] + (B[2]-sigma(A,3,x,3,3,3))/A[2][2];
213         */
214
215         if(deltaT < eps)// convergence criterion
216         {
217             printf("solution converged!!\n");
218             print2Darray(T,N,N);
219             break;
```

Using Jacobi method, solve the equation **AT** = **b**

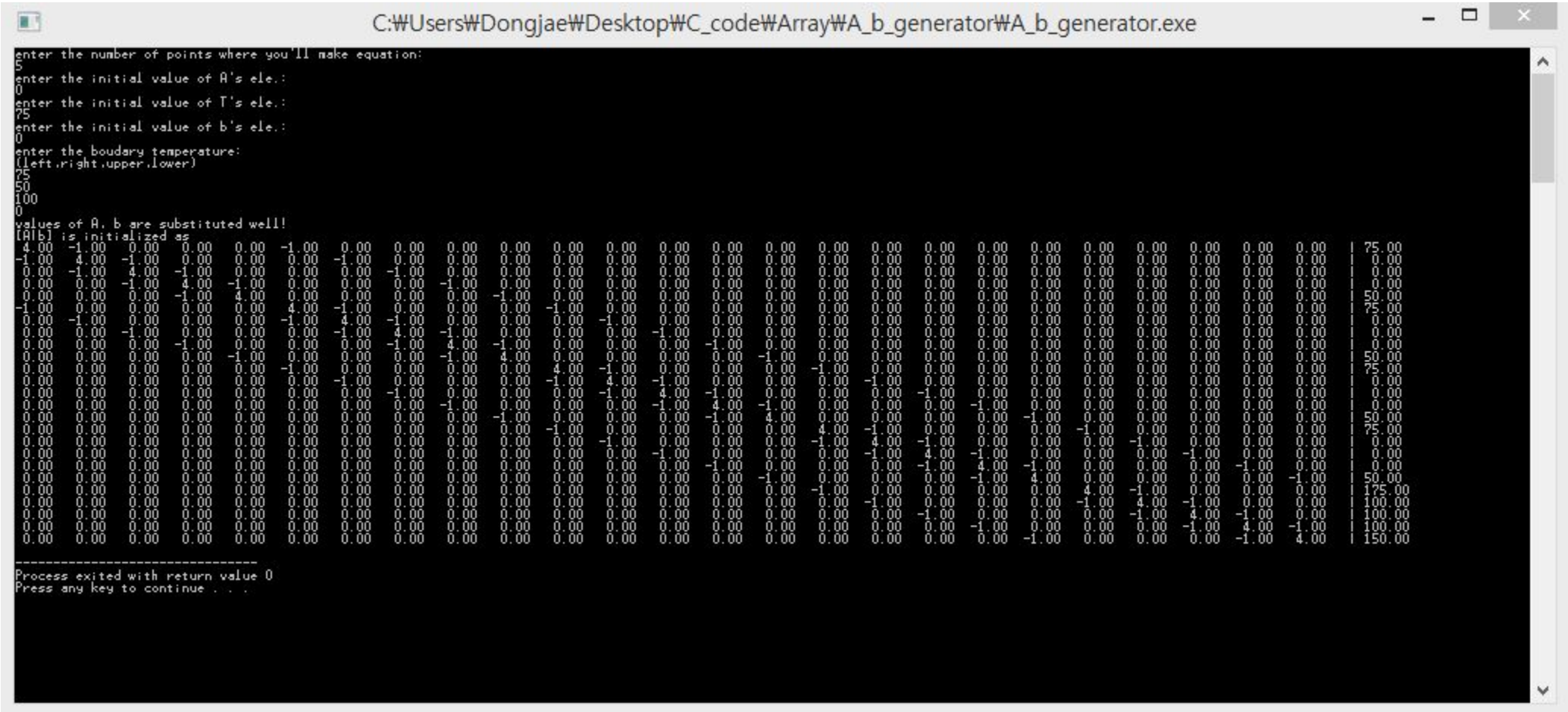
```
207         }
208
209         /*
210         x1new = x[0] + (B[0]-sigma(A,3,x,3,1,3))/A[0][0]; //iterative process
211         x2new = x[1] + (B[1]-sigma(A,3,x,3,2,3))/A[1][1];
212         x3new = x[2] + (B[2]-sigma(A,3,x,3,3,3))/A[2][2];
213         */
214
215         if(deltaT < eps)// convergence criterion
216         {
217             printf("solution converged!!\n");
218             print2Darray(T,N,N);
219             break;
220         }
221         else if(k==99999)
222             printf("solution not converged!!\n");
223
224         for(i=1;i<N+1;i++)
225         {
226             for(j=1;j<N+1;j++)
227             {
228                 T[i-1][j-1] = Tnew[i-1][j-1];
229             }
230         }
231         printf("iteration %d, change : %5.3f\n",k+1,deltaT);
232     }
233
234
235     /*freeing the memories dynamically allocated on 'heap' area*/
236     for(k=0;k<N;k++)
237     free(T[k]),free(A[k]);
238
239     free(T),free(A),free(b);
240
241     return 0;
242 }
243
```

Using Jacobi method, solve the equation **AT** = **b**

C:₩Users₩Dongjae₩Desktop₩C_code₩Array₩Jacobi_Method₩Jacobi_metho...

```
iteration 1 : x= (3.116667e+000, -2.614286e+000, 7.090000e+000)
iteration 2 : x= (3.002190e+000, -2.497810e+000, 6.994214e+000)
iteration 3 : x= (2.999687e+000, -2.500279e+000, 6.999978e+000)
iteration 4 : x= (2.999989e+000, -2.499996e+000, 7.000004e+000)
iteration 5 : x= (3.000000e+000, -2.500000e+000, 7.000000e+000)
iteration 6 : x= (3.000000e+000, -2.500000e+000, 7.000000e+000)
iteration 7 : x= (3.000000e+000, -2.500000e+000, 7.000000e+000)
iteration 8 : x= (3.000000e+000, -2.500000e+000, 7.000000e+000)
iteration 9 : x= (3.000000e+000, -2.500000e+000, 7.000000e+000)
iteration 10 : x= (3.000000e+000, -2.500000e+000, 7.000000e+000)
iteration 11 : x= (3.000000e+000, -2.500000e+000, 7.000000e+000)
iteration 12 : x= (3.000000e+000, -2.500000e+000, 7.000000e+000)
iteration 13 : x= (3.000000e+000, -2.500000e+000, 7.000000e+000)
solution converged!!
 x1 = 3.000000e+000
 x2 = -2.500000e+000
 x3 = 7.000000e+000
change : 0.000000e+000
```

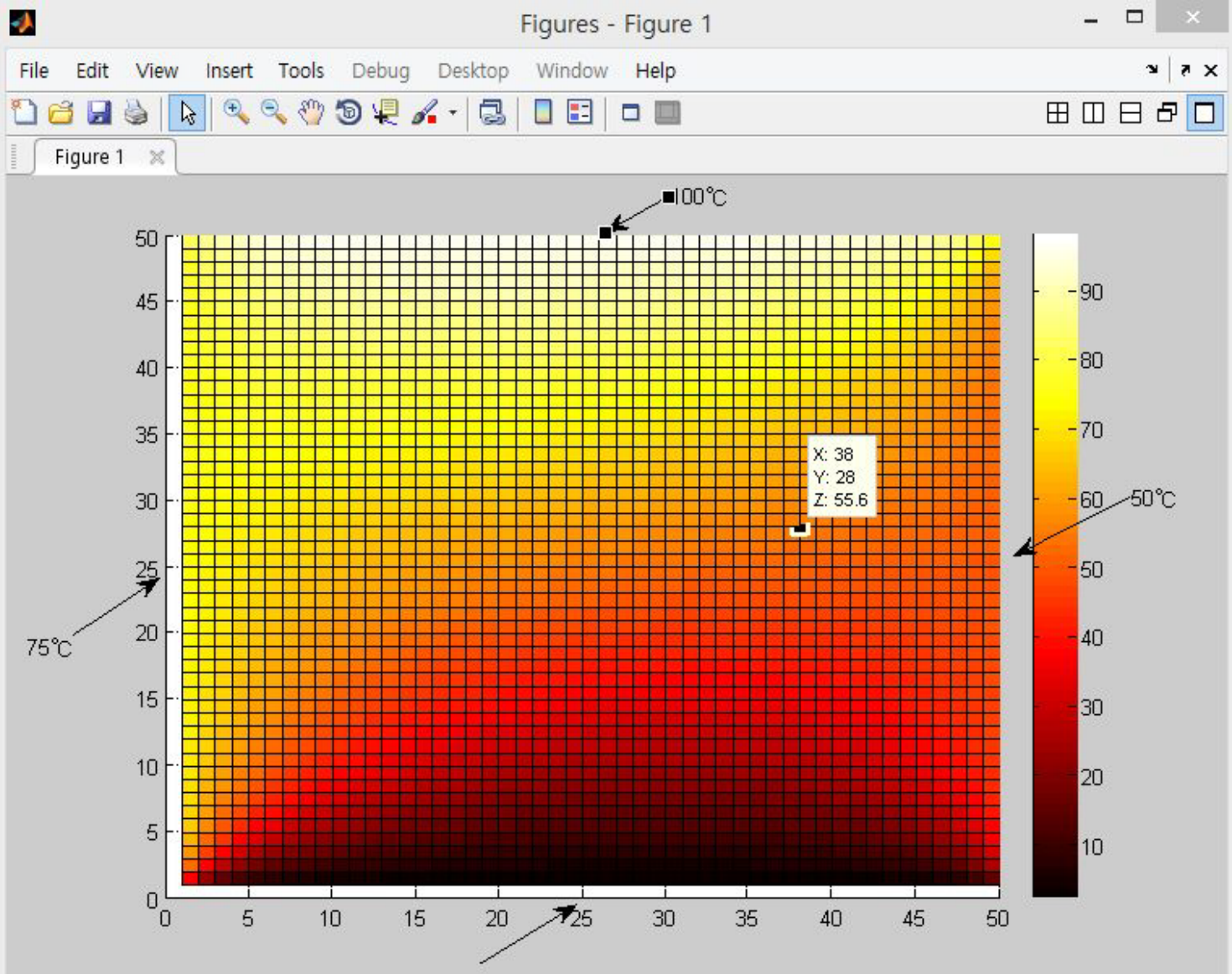
Calculation for the system where more grids are applied (25 by 25)



Solution of the PDE (by Matlab)

파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)

63.239	52.943	44.578	38.005	32.886	28.883	25.722	23.193	21.143	19.461	18.066	16.8
65.624	57.018	49.581	43.377	38.289	34.140	30.750	27.966	25.663	23.742	22.127	20.7
67.238	59.924	53.350	47.632	42.754	38.636	35.173	32.259	29.800	27.716	25.941	24.4
68.403	62.088	56.265	51.046	46.460	42.478	39.046	36.096	33.562	31.380	29.499	27.8
69.286	63.761	58.577	53.829	49.560	45.771	42.438	39.519	36.970	34.745	32.803	31.1
69.981	65.094	60.451	56.132	52.179	48.609	45.414	42.572	40.054	37.828	35.863	34.1
70.543	66.183	62.003	58.067	54.417	51.073	48.037	45.300	42.845	40.650	38.692	36.9
71.010	67.091	63.310	59.717	56.349	53.227	50.361	47.747	45.376	43.235	41.307	39.5
71.404	67.863	64.428	61.141	58.034	55.127	52.432	49.950	47.678	45.608	43.727	42.0
71.743	68.529	65.398	62.386	59.519	56.816	54.289	51.944	49.779	47.790	45.970	44.3
72.038	69.111	66.251	63.486	60.840	58.329	55.966	53.757	51.703	49.803	48.053	46.4
72.299	69.626	67.008	64.467	62.024	59.694	57.488	55.414	53.473	51.667	49.993	48.4
72.532	70.087	67.687	65.351	63.096	60.935	58.880	56.937	55.109	53.399	51.805	50.3
72.742	70.504	68.302	66.153	64.073	62.072	60.160	58.344	56.628	55.014	53.502	52.0
72.934	70.883	68.863	66.888	64.970	63.119	61.344	59.651	58.044	56.526	55.099	53.7
73.109	71.232	69.380	67.566	65.799	64.090	62.445	60.871	59.371	57.949	56.605	55.3
73.272	71.555	69.859	68.195	66.572	64.997	63.477	62.017	60.622	59.293	58.033	56.8
73.424	71.856	70.307	68.785	67.296	65.849	64.448	63.099	61.805	60.569	59.392	58.2
73.566	72.140	70.729	69.340	67.979	66.654	65.368	64.125	62.930	61.784	60.690	59.6
73.701	72.408	71.128	69.866	68.628	67.419	66.243	65.105	64.006	62.949	61.936	60.9
73.829	72.664	71.508	70.368	69.248	68.151	67.082	66.044	65.039	64.069	63.137	62.2
73.952	72.909	71.874	70.851	69.844	68.856	67.891	66.950	66.037	65.153	64.299	63.4
74.071	73.146	72.227	71.318	70.421	69.539	68.674	67.829	67.006	66.206	65.430	64.6
74.187	73.377	72.571	71.772	70.982	70.204	69.438	68.687	67.952	67.235	66.536	65.8
74.301	73.603	72.908	72.217	71.533	70.856	70.187	69.528	68.880	68.245	67.622	67.0
74.413	73.826	73.240	72.657	72.076	71.499	70.926	70.358	69.796	69.241	68.694	68.1
74.524	74.048	73.571	73.094	72.616	72.138	71.660	71.182	70.705	70.230	69.758	69.2
74.635	74.270	73.902	73.531	73.156	72.777	72.393	72.005	71.613	71.217	70.818	70.4
74.748	74.494	74.236	73.972	73.700	73.420	73.131	72.832	72.524	72.207	71.881	71.5
74.863	74.723	74.576	74.420	74.253	74.073	73.879	73.669	73.445	73.205	72.951	72.6
74.981	74.957	74.925	74.880	74.819	74.740	74.641	74.521	74.380	74.218	74.035	73.8
75.104	75.201	75.287	75.356	75.404	75.427	75.425	75.394	75.336	75.250	75.138	75.0
75.232	75.457	75.665	75.852	76.012	76.141	76.236	76.295	76.320	76.310	76.266	76.1
75.369	75.727	76.066	76.376	76.652	76.888	77.082	77.231	77.338	77.402	77.426	77.4
75.516	76.018	76.495	76.935	77.331	77.678	77.971	78.211	78.398	78.534	78.623	78.6
75.676	76.335	76.960	77.538	78.060	78.520	78.915	79.244	79.509	79.715	79.864	79.9
75.854	76.685	77.471	78.197	78.852	79.428	79.923	80.340	80.681	80.951	81.157	81.304	81.399	81.447	81.453	81.422	81.358	81.263	81.142	80.995	80.824	80.631	80.416	80.180	79.920	79.637	79.330	78
76.055	77.078	78.043	78.929	79.722	80.416	81.011	81.511	81.923	82.252	82.508	82.699	82.832	82.914	82.951	82.948	82.909	82.840	82.741	82.617	82.468	82.296	82.102	81.885	81.646	81.383	81.094	80
76.287	77.531	78.695	79.752	80.690	81.503	82.195	82.772	83.246	83.627	83.925	84.152	84.316	84.426	84.488	84.508	84.493	84.444	84.367	84.263	84.135	83.984	83.810	83.614	83.396	83.153	82.886	82
76.563	78.064	79.452	80.696	81.783	82.712	83.492	84.137	84.662	85.083	85.414	85.667	85.855	85.985	86.066	86.106	86.108	86.078	86.019	85.935	85.826	85.694	85.541	85.366	85.169	84.949	84.705	84
76.901	78.709	80.353	81.798	83.033	84.069	84.924	85.621	86.183	86.630	86.980	87.249	87.450	87.593	87.687	87.740	87.756	87.741	87.698	87.630	87.539	87.426	87.293	87.139	86.965	86.769	86.550	86
77.333	79.517	81.455	83.108	84.483	85.607	86.514	87.241	87.818	88.273	88.627	88.899	89.103	89.250	89.350	89.410	89.435	89.431	89.401	89.348	89.273	89.179	89.065	88.933	88.782	88.612	88.420	88
77.914	80.571	82.840	84.698	86.185	87.361	88.285	89.010	89.576	90.016	90.357	90.617	90.812	90.955	91.053	91.114	91.144	91.148	91.128	91.087	91.027	90.950	90.856	90.746	90.619	90.475	90.312	90
78.751	82.015	84.635	86.658	88.198	89.367	90.256	90.937	91.459	91.860	92.167	92.400	92.575	92.703	92.792	92.849	92.880	92.887	92.875	92.845	92.799	92.739	92.664	92.576	92.473	92.357	92.225	92
80.077	84.102	87.027	89.103	90.582	91.652	92.437	93.022	93.463	93.797	94.050	94.242	94.385	94.490	94.564	94.612	94.638	94.647	94.640	94.619	94.586	94.541	94.485	94.419	94.342	94.254	94.153	94
82.453	87.290	90.267	92.143	93.377	94.220	94.818	95.252	95.574	95.814	95.996	96.132	96.234	96.308	96.361	96.395	96.415	96.422	96.419	96.406	96.384	96.354	96.318	96.273	96.222	96.162	96.095	96
87.447	92.336	94.608	95.828	96.560	97.036	97.362	97.595	97.765	97.891	97.986	98.057	98.110	98.149	98.176	98.194	98.204	98.208	98.207	98.201	98.190	98.175	98.157	98.135	98.109	98.079	98.045	98



References

Richard G. Rice, Applied Mathematics And Modeling For Chemical Engineers 2nd edition

Steven C. Chapra et al., Numerical Methods for Engineers 6th edition