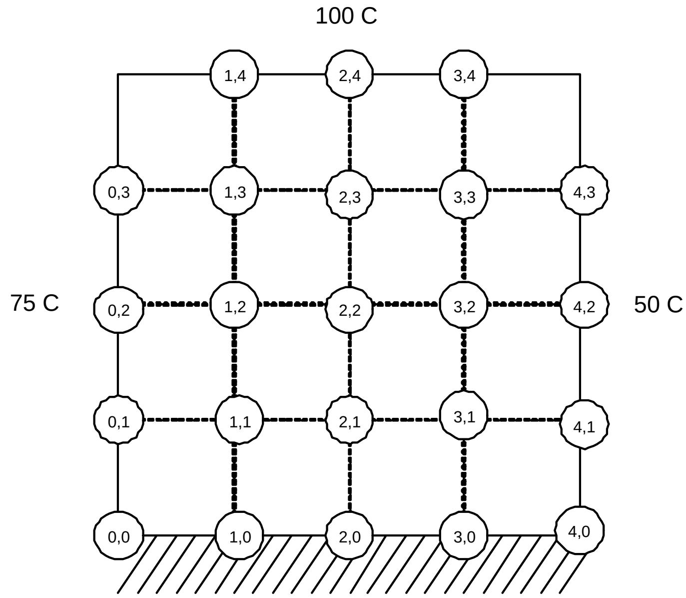

Figure 1: Initial state of system

Problem 29.3

Nasser Abbasi. UCI. MAE 185. May 24, 2003

Use Liebmann method (Gauss-Siedel) to solve for the temp of the heated plate shown. Employee overrelaxation with a value of 1.5 for the weighting factor and iterate to $\epsilon_s = 1\%$. The plate has the lower edge insulated.

Solution

The lower edge of the plate is insulated. This means the flux flow of heat is zero at that edge. This is a condition of the rate of change of the dependent variable T at those points. This is called Nuemann boundary conditions (vs. Drichlete boundary conditions which sets values on the indepedent variable T itself, not its rate of change).

This means we now have an additional 3 grid points that we do not know the tempurature at (those are the points $T_{1,0}, T_{2,0}, T_{3,0}$). So instead of only 9 equations to solve as before, now there are 12 equations to solve.

The tempreature at each one of those edge points is found using the method of images. See diagram.

To allow finding T at the insulated edge, and to be able to use the 4-point difference equation, one of the 4 points must then be outside the physical plate boundaries. Such a point is interoduced momenterraly, then solved for in terms of the inside point, resukting in a new 4-points differecne equation that can be used only to evaluate the T for those points on this special edge.

Looking at the diagram, this is done as follows.

Assume we want to find the 4-point difference equation for the point T_2, then as normally, it is $T_2 = \frac{T_4+T_3+T_1+T_x}{4}$, where T_x is the imaginary outside point. Now the flux at the point T_2 in the y-direction is defined as $\frac{T_1-T_x}{2h}$, where h is the vertrical distance between any 2 points.

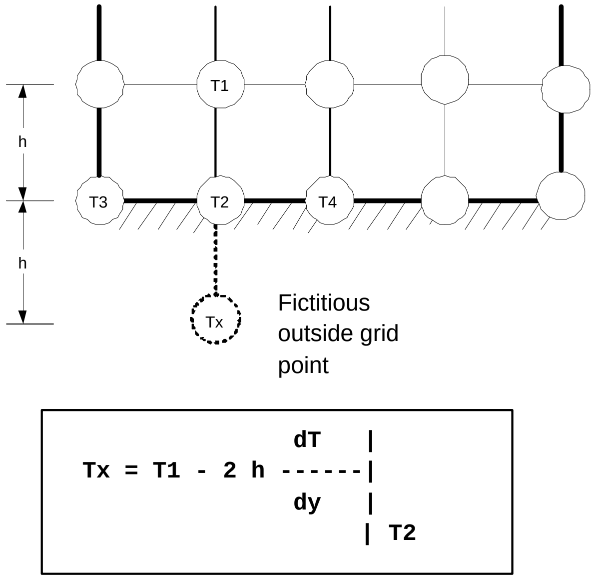

Figure 2: Finding T at edge with Nuemann condition

So, $\left.\frac{\partial T}{\partial y}\right|_{T_2} = \frac{T_1 - T_x}{2h}$
hence, solve for T_x results in

$$T_x = T_1 - 2h \left.\frac{\partial T}{\partial y}\right|_{T_2}$$

For insulated edge, the flux at the point is zero. This means no change of Temreature at the edge, hence $\left.\frac{\partial T}{\partial y}\right|_{T_2} = 0$, hence

$$T_x = T_1$$

So, now looking back at the original 4-point difference equation we get

$$T_2 = \frac{T_4 + T_3 + T_1 + T_1}{4} = \frac{T_4 + T_3 + 2T_1}{4}$$

This means that for the insulated edge, use the above modified difference equation to solve for T at each point on that edge.
Other than this change, the algorithm remains the same as problem 29.1

Start with all grid points at $T = 0$, this now includes those additional 3 points on the lower edge.
I will show the solution by hand for one iteration, then write a matlab function to solve.

For point (1,0): $T_{_{1,0}} = \frac{T_{2,0} + T_{0,0} + 2T_{1,1}}{4} = \frac{0+75+2(0)}{4} = 18.75$
Now, apply the overrelaxation step to improve convergence. Use $\lambda = 1.5$
$T_{i,j} = \lambda T_{i,j} + (1 - \lambda)\, T_{i,j}^{old}$
$T_{1,0} = (1.5)\, 18.75 + (1 - 1.5)\, 0 = 28.125$

For point (2,0): $T_{_{2,0}} = \frac{T_{3,0} + T_{1,0} + 2T_{2,1}}{4} = \frac{0+28.125+2(0)}{4} = 7.03125$

$T_{2,0} = (1.5)\,7.03125 + (1-1.5)\,0 = 10.546875$

For point (3,0): $T_{3,0} = \frac{T_{4,0}+T_{2,0}+2T_{3,1}}{4} = \frac{50+10.546875+2(0)}{4} = 15.13671875$
$T_{2,0} = (1.5)\,15.13671875 + (1-1.5)\,0 = 22.705078125$

For point (1,1): $T_{1,1} = \frac{T_{2,1}+T_{0,1}+T_{1,2}+T_{1,0}}{4} = \frac{0+75+0+28.125}{4} = 25.78125$
$T_{1,1} = (1.5)\,25.78125 + (1-1.5)\,0 = 38.671875$

For point (2,1): $T_{2,1} = \frac{T_{3,1}+T_{1,1}+T_{2,2}+T_{2,0}}{4} = \frac{0+28.125+0+10.546875}{4} = 9.66796875$
$T_{2,1} = (1.5)\,9.66796875 + (1-1.5)\,0 = 14.501953125$

For point (3,1): $T_{3,1} = \frac{T_{4,1}+T_{2,1}+T_{3,2}+T_{3,0}}{4} = \frac{50+10.546875+0+22.705078125}{4} = 20.81298828125$
$T_{3,1} = (1.5)\,20.81298828125 + (1-1.5)\,0 = 31.219482421875$

For point (1,2): $T_{1,2} = \frac{T_{2,2}+T_{0,2}+T_{1,3}+T_{1,1}}{4} = \frac{0+75+0+38.671875}{4} = 28.41796875$
$T_{1,2} = (1.5)\,28.41796875 + (1-1.5)\,0 = 42.626953125$

For point (2,2): $T_{2,2} = \frac{T_{3,2}+T_{1,2}+T_{2,3}+T_{2,1}}{4} = \frac{0+42.626953125+0+14.501953125}{4} = 14.2822265625$
$T_{2,2} = (1.5)\,14.2822265625 + (1-1.5)\,0 = 21.42333984375$

For point (3,2): $T_{3,2} = \frac{T_{4,2}+T_{2,2}+T_{3,3}+T_{3,1}}{4} = \frac{50+21.42333984375+0+31.219482421875}{4} = 25.6607$
$T_{3,2} = (1.5)\,25.6607 \ + (1-1.5)\,0 = 38.49105$

For point (1,3): $T_{1,3} = \frac{T_{2,3}+T_{0,3}+T_{1,4}+T_{1,2}}{4} = \frac{0+75+100+42.626953125}{4} = 54.40673828125$
$T_{1,3} = (1.5)\,54.40673828125 + (1-1.5)\,0 = 81.610107421875$

For point (2,3): $T_{2,3} = \frac{T_{3,3}+T_{1,3}+T_{2,4}+T_{2,2}}{4} = \frac{0+81.610107421875+100+21.42333984375}{4} = 50.75836$
$T_{2,3} = (1.5)\,50.7583618164063 + (1-1.5)\,0 = 76.1375427246094$

For point (3,3): $T_{3,3} = \frac{T_{4,3}+T_{2,3}+T_{3,4}+T_{3,2}}{4} = \frac{50+76.1375427246094+100+38.49105}{4} = 66.1571$
$T_{3,3} = (1.5)\,66.1571481811524 + (1-1.5)\,0 = 99.2357$

The maximum value above is $T_{3,3} = 99.2357$ C. Apply the tolerance test to this value
$\left|\frac{T_{3,3}^{new}-T_{3,3}^{old}}{T_{3,3}^{new}}\right| \times 100\% = \left|\frac{99.2357-0}{99.2357}\right| \times 100\% = 100\%$, which is larger than $\epsilon_s = 1\%$, so continue.

This process is repeated until solution T is found within the given ϵ_s.

A MATLAB function called nma_laplaceRectNeumann is written to fully solve this.
This function accepts as input the number of points in the x-direction, the number of points in the y-direction, and the values of the dependent variable at the 3 boundaries of the plate, 3 values, one for each side in the order: left, top, right. And the value λ, and a tolerance value ϵ_s. Notice no boundary value is given for the bottom edge since insulated.

The function returns back a matrix that contains the solution at each grid point.

This below shows a run of the function, it shows how the plate is being filled up with updated values of the solution $T(x, y)$ one iteration at a time, until the approximate solution converges to the desired accuracy. It took 5 iterations to get the desired accuracy.
THe final answer is (after 5 iterations):

```
ans =

                    0                   100                   100                   100                     0
                   75      83.1982181509375      83.3322936530749      74.1729399082487                    50
                   75      74.9399313353933        74.28252883401      64.3859843654354                    50
                   75      79.8756086267531      71.6580588603392      61.7159255445586                    50
                   75      77.2548146545887       79.256187658757      60.2580134407617                    50
```

Full run is :

```
>> nx=3; ny=3; left=75; top=100; right=50; lambda=1.5; tol=1;
>> nma_laplaceRectNuemann(nx,ny,left,top,right,lambda,tol)

Iteration number 1

A =

                    0                   100                   100                   100                     0
                   75      81.610107421875      76.6937255859375      100.069999694824                    50
                   75       42.626953125      22.906494140625      40.1596069335938                    50
                   75          38.671875          18.45703125      34.185791015625                    50
                   75             28.125            10.546875        22.705078125                    50


epsilonA =

   100

Iteration number 2

A =

                    0                   100                   100                   100                     0
                   75      77.7688980102539      93.1961059570313      72.0786541700363                    50
                   75      64.5034790039063      72.9423522949219      82.4403047561646                    50
                   75        49.32861328125      43.6981201171875      50.3746032714844                    50
                   75         47.021484375          34.716796875      46.0556030273438                    50


epsilonA =

         17.7071565401051

Iteration number 3

A =

                    0                   100                   100                   100                     0
                   75      90.4087901115417      83.2638680934906      74.8046586290002                    50
                   75       76.585865020752      83.8110119104385      62.3200938105583                    50
                   75      64.5223617553711      68.5302257537842      70.2176570892334                    50
                   75      54.6295166015625      53.1721115112305      53.4426927566528                    50


epsilonA =

         13.9808220922915

Iteration number 4

A =

                    0                   100                   100                   100                     0
                   75      83.0483330413699      82.6530944555998      74.9428286973853                    50
```

```
          75          83.7434068322182          73.5737510025501          66.9339935760945                    50
          75          76.2104362249374          78.6173164844513          63.2056983187795                    50
          75          69.1415548324585          70.7807064056396          71.2346613407135                    50


epsilonA =

          8.54699143755462

Iteration number 5

A =

           0                       100                       100                       100                     0
          75          83.1982181509375          83.3322936530749          74.1729399082487                    50
          75          74.9399313353933            74.28252883401          64.3859843654354                    50
          75          79.8756086267531          71.6580588603392          61.7159255445586                    50
          75          77.2548146545887           79.256187658757          60.2580134407617                    50


epsilonA =

         0.815049205656945


ans =

           0                       100                       100                       100                     0
          75          83.1982181509375          83.3322936530749          74.1729399082487                    50
          75          74.9399313353933            74.28252883401          64.3859843654354                    50
          75          79.8756086267531          71.6580588603392          61.7159255445586                    50
          75          77.2548146545887           79.256187658757          60.2580134407617                    50

>>
```

Now, I will show just the final solution A for increasing accuracy by making ϵ_s smaller and smaller.

```
>> nma_laplaceRectNuemann(nx,ny,left,top,right,lambda, 1)

ans =

           0                       100                       100                       100                     0
          75          83.1982181509375          83.3322936530749          74.1729399082487                    50
          75          74.9399313353933            74.28252883401          64.3859843654354                    50
          75          79.8756086267531          71.6580588603392          61.7159255445586                    50
          75          77.2548146545887           79.256187658757          60.2580134407617                    50

>> nma_laplaceRectNuemann(nx,ny,left,top,right,lambda, 0.1)

ans =

           0                       100                       100                       100                     0
          75          83.4689690724228          82.6177095784203          74.2788404925504                    50
          75           76.125764064726          72.9089171883304          64.3898390870131                    50
          75          72.9171200137559          68.4369427343141          60.6345477874855                    50
          75          71.8635556191911          67.0514674195008          59.5224147381791                    50

>> nma_laplaceRectNuemann(nx,ny,left,top,right,lambda, 0.01)

ans =

           0                       100                       100                       100                     0
          75          83.4131035840263          82.6228707034544          74.2604629696066                    50
          75          76.0216514060357          72.8312767430722          64.4153242811392                    50
          75          72.7898987811884           68.287227319965          60.5537485357759                    50
          75          71.9063488387076          66.9754797440981           59.522855252822                    50

>> nma_laplaceRectNuemann(nx,ny,left,top,right,lambda, 0.001)
```

```
ans =

                 0               100               100               100                 0
                75  83.4131035840263  82.6228707034544  74.2604629696066                50
                75  76.0216514060357  72.8312767430722  64.4153242811392                50
                75  72.7898987811884   68.287227319965  60.5537485357759                50
                75  71.9063488387076  66.9754797440981   59.522855252822                50

>> nma_laplaceRectNuemann(nx,ny,left,top,right,lambda, 0.0001)

ans =

                 0               100               100               100                 0
                75  83.4131035840263  82.6228707034544  74.2604629696066                50
                75  76.0216514060357  72.8312767430722  64.4153242811392                50
                75  72.7898987811884   68.287227319965  60.5537485357759                50
                75  71.9063488387076  66.9754797440981   59.522855252822                50

>> nma_laplaceRectNuemann(nx,ny,left,top,right,lambda, 0.00001)

ans =

                 0               100               100               100                 0
                75  83.4109244315597  82.6286024444071  74.2614411818133                50
                75  76.0151000103032  72.8420406416257  64.4171642922854                50
                75  72.8074352894845  68.3072942256936  60.5651671697711                50
                75  71.9073523997358  67.0145426018174  59.5362184236174                50

>>
```

source code

```
function A=nma_laplaceRectNuemann(nx,ny,left,top,right,lambda,tol)
%function A=nma_laplaceRectNuemann(nx,ny,left,top,right,lambda,tol)
%
% Function that solves the laplace PDE for rectangular region for
% Nuemann boundary conditions at the bottom edge only.
%
%INPUT:
%  nx: number of grid points in the x-direction
%  ny: number of grid points in the y-direction
%  right: right edge boundary value
%  top: top edge boundary value
%  left: left edge boundary value
%  lambda: the value lambda to use for the relaxation method of Liebmann
%  tol: the absolute tolerance to check for convergance.
%
%OUTPUT:
%  the solution matrix.
%
%Author: Nasser Abbasi
%May 25, 2003

TRUE  = 1;
FALSE = 0;

A     = zeros(nx+2,ny+2);
A_old = A;

% apply dirchlet conditions
A(1,:)   = top;
A(:,1)   = left;
A(end,:) = 0;
A(:,end) = right;

%patch the 3 corners, just for cosmotics purposes
A(1,1)   = 0;
A(end,1) = left;
A(1,end) = 0;
A(end,end) = right;

A_old    = A;

converged = FALSE;
nIter     = 0;
```

```
while(~converged)
    nIter = nIter+1;
    %fprintf('Iteration number %d\n',nIter);

    for(i=2+nx:-1:2)
        for(j=2:1:2+ny-1)

            if( i == 2+nx ) %insulated edge?
                A(i,j) = A(i,j+1) + A(i,j-1) + 2*A(i-1,j);
            else
                A(i,j) = A(i,j+1) + A(i,j-1) + A(i-1,j) + A(i+1,j);
            end

            A(i,j) = A(i,j)/4;
            A(i,j) = lambda*A(i,j) + (1-lambda)*A_old(i,j);
        end
    end

    [max_row , max_row_index]     = max(A(2:end-1,2:end-1));
    [max_element , max_col_index] = max(max_row);

    pos = [max_row_index(1)+1,max_col_index+1];

    epsilonA = abs(A(pos(1),pos(2)) - A_old(pos(1),pos(2)))/A(pos(1),pos(2))*100;

    if(  epsilonA  < tol )
        converged = TRUE;
    end

    A_old = A;
    %A
    %epsilonA
end

%A=A(2:end-1,2:end-1);
```