Indian Institute of Technology, Madras

Finite Element Methods for Design

ED4030

Tutorial 2

Submitted by:
Athul Vijayan
ED11B004

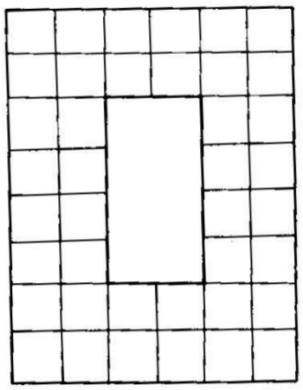
August 27, 2014

1 Problem 3

Notations used are:

1. Temperature at grid point (i, j) (where i along horizontal and j along vertical) is ϕ_{ij}
2. Derivatives are represented by $\frac{\partial \phi}{\partial x} \rightarrow \phi^i_{i,j}$, $\frac{\partial \phi}{\partial y} \rightarrow \phi^j_{i,j}$, $\frac{\partial^2 \phi}{\partial x^2} \rightarrow \phi^{ii}_{i,j}$, $\frac{\partial^2 \phi}{\partial xy} \rightarrow \phi^{ij}_{i,j}$, $\frac{\partial^2 \phi}{\partial y^2} \rightarrow \phi^{jj}_{i,j}$

The given grid for studying the heat distribution is:



With Length = 6 and Width = 8.
For finite elements with $N\Delta x = 6$ and $M\Delta y = 8$, Since internal heat generation is Zero, The governing equation is for heat flow is:

$$k\left(\frac{\partial^2 \phi}{\partial^2 x} + \frac{\partial^2 \phi}{\partial^2 y}\right) = 0$$

Where the boundary conditions for the grid are:
For $\Delta x = \{1, 1/2, 1/4...,\}$ and $\Delta y = \{1, 1/2, 1/4...,\}$

$$\phi_{\frac{2}{\Delta x}, \frac{j}{\Delta y}} = 100 \qquad \text{for } j = \{2, 3, 4, 5, 6\} \tag{1}$$

$$\phi_{\frac{4}{\Delta x}, \frac{j}{\Delta y}} = 100 \qquad \text{for } j = \{2, 3, 4, 5, 6\} \tag{2}$$

$$\phi_{\frac{3}{\Delta x}, \frac{2}{\Delta y}} = 100 \tag{3}$$

$$\phi_{\frac{3}{\Delta x}, \frac{6}{\Delta y}} = 100 \tag{4}$$

$$k\frac{\partial \phi}{\partial x} = -\alpha\phi \qquad \text{at } i = \{0, \frac{n}{\Delta x}\} \tag{5}$$

$$k\frac{\partial \phi}{\partial y} = -\alpha\phi \qquad \text{at } j = \{0, \frac{m}{\Delta y}\} \tag{6}$$

With finite elements aproximation, the governing equation can be written as:

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

with $\Delta x = \Delta y = h$

$$\phi_{i+1,j} - 4\phi_{i,j} + \phi_{i-1,j} + \phi_{i,j+1} + \phi_{i,j-1} = 0$$

Now we need to make the recursive equation into a matrix equation in the form of $AX = B$. For coding simplicity, we initially consider matrix $\mathbf{\Phi}$ as an (n-1)(m-1) x 1 matrix as:

$$\mathbf{\Phi} = \begin{bmatrix} \phi_{1,1} & \phi_{2,1} & \phi_{3,1} & \cdots & \phi_{n-1,1} & \phi_{1,2} & \phi_{2,2} & \phi_{n-1,2} & \cdots & \phi_{n-1,m-1} \end{bmatrix}^T$$

For now, we does not care about boundary condition; we will do that after a while.
And we define matric $B_{n-1Xn-1}$ as:

$$\mathbf{B_{n-1Xn-1}} = \begin{pmatrix} -4 & 1 & 0 & 0 & \cdots & 0 \\ 1 & -4 & 1 & 0 & \cdots & 0 \\ 0 & 1 & -4 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 & -4 \end{pmatrix}$$

And $I_{n-1Xn-1}$ as:

$$\mathbf{I_{n-1Xn-1}} = \begin{pmatrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \end{pmatrix}$$

and a zero matrix is defined as:

$$\mathbf{0_{mXm}} = \begin{pmatrix} 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 0 \end{pmatrix}$$

The governing equation with zero initial conditions (all border points are at zero temperature) applied to a rectangular sheet in 2D can be written as:

$$\mathbf{A\Phi = F} \quad .$$

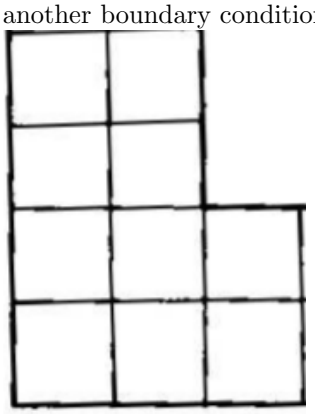
where A is defined as:

$$\mathbf{A_{(n-1)(m-1)x(n-1)(m-1)}} = \begin{pmatrix} \mathbf{B} & \mathbf{I} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{I} & \mathbf{B} & \mathbf{I} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{I} & \mathbf{B} & \mathbf{I} & \cdots & \mathbf{0} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{I} & \mathbf{B} \end{pmatrix}$$

and $\mathbf{F}$ is

$$\mathbf{A_{(n-1)(m-1)\times 1}} = \begin{bmatrix} 0 & 0 & 0 & 0 & \cdots & 0 \end{bmatrix}^{\mathbf{T}}$$

To work with the hollow part, we can use the symmetry of the given plate. If we divide the plate into four as shown below, we just need to add another boundary condition for the newly made sides.



Since the shape is not rectangular, we need to modify the $\mathbf{A}$ matrix for this.
Say the number of elements inside one block in the figure is $\frac{1}{\Delta x}$.

1. Length along x (i) direction is more on the bottom part, which implies the dimension of square matrix $\mathbf{B}$ will be more at the bottom.
2. Dimension of $\mathbf{B}$ and $\mathbf{I}$ on $j \leq \frac{2}{\Delta y}$ is $(\frac{3}{\Delta x} - 1) \times (\frac{3}{\Delta x} - 1)$
3. Length along x (i) direction is less on the top part, which implies the dimension of square matrix $\mathbf{B}$ will be less at the top.
4. Dimension of $\mathbf{B}$ and $\mathbf{I}$ on $j > \frac{2}{\Delta y}$ is $(\frac{2}{\Delta x} - 1) \times (\frac{2}{\Delta x} - 1)$

Finally, matrices for our problem with zero boundary conditions are:

$$\mathbf{\Phi} = \begin{bmatrix} \phi_{1,1} & \phi_{2,1} & \cdots & \phi_{(\frac{3}{\Delta x}-1),1} & \phi_{1,2} & \cdots & \phi_{(\frac{3}{\Delta x}-1),2} & \cdots & \phi_{(\frac{3}{\Delta x}-1),(\frac{2}{\Delta y}-1)} & \cdots \end{bmatrix}^T$$

matrix $\mathbf{\Phi}$ is all the points inside the mesh (excluding all boundary points) as shown in the order.
For e.g. for $\Delta x = \Delta y = 1$,

$$\mathbf{\Phi} = \begin{bmatrix} \phi_{1,1} & \phi_{2,1} & \phi_{1,2} & \phi_{1,3} \end{bmatrix}^T$$

matrix $\mathbf{F}$ is zero matrix.
So for our probem, with zero boundary conditions, the equation in matrix form can be formed by following code.

1. Step 1: Make the A matrix for bottom rectangular part and apply Derivative BC at $i = 0$ and $i = n - 1$

```
clear all
numGrids = 5;                   % number of different grids over which solution
    is sought

xArray = cell(numGrids,1);      % array storing the x values at grid locations
yArray = cell(numGrids,1);

phiFD = cell(numGrids,1);       % solution vectors
midValsFD = zeros(numGrids,1);  % an array storing all phi values at x , FD
stepVals = zeros(numGrids,1);   % array storing delta x values for different grids
phi_0 = 0;                      % phi b.c. at x = 0
i=2;
% for i = 1:numGrids
    % ============ First we produce the matrix A for the bottom rectangular part.
        ==============
    xNumFirstGridElems = 3;
    yNumFirstGridElems = 2;
    xNumElem = xNumFirstGridElems*2^(i-1);  % x number of elements
    yNumElem = yNumFirstGridElems*2^(i-1);  % y number of elements

    deltaX = 1/yNumElem;

    % make B matrix., so that we can just pile this to form A matrix
    BdiagVec = -4*ones(xNumElem-1,1);       % exclude two boundaries
    BdiagVec(1) = -3+deltaX;                %add derivative boundary condition, at
        all i = 1 for bottom part
    BdiagVec(xNumElem-1) = -3+deltaX;       %add derivative boundary condition, at
        all i = n-1 for bottom part
    if (xNumElem-1 > 1)
        B_bottom = gallery('tridiag',ones(xNumElem -2, 1),BdiagVec,ones(xNumElem
            -2, 1)); % matrix of coefficients
    else
        B_bottom = BdiagVec                 % case for $$\Delta x =1$$
    end

    A = B_bottom;                           %Initialize A matrix
    for k=1:yNumElem-2
        A = blkdiag(A, B_bottom);           %populate A matrix with B matrix as
            diagonal
    end
    % Now loop through the A matrix to add I matrix wherever necessary
    for k= 1: (xNumElem - 1)*(yNumElem-1)
        if (k + xNumElem -1 <= size(A, 2))  % leave the boundary, i=n and j=m
            A(k, k + xNumElem -1) = 1;
        end
        if (k > (xNumElem - 1))             % leave the boundary, i=0 and j=0
            A(k, k -( xNumElem -1)) = 1;
        end
```

```
    end
    % Now the vector A has the coefficients of $$\phi$$s inside the bottom
        rectangle.
       % ======================================================================
```

2. Step 2: populate A matrix to include equations for top rectangular part and apply Derivative BC at $i = 0$ and $i = n - 1$

```
% Now we add to this matrix, the coefficients for the equations of top rectangular
    part
% having different dimensions. So this will result in change of dimesion of B
    matrix and I matrix

% ========== then we produce the matrix A for the bottom rectangular part.
    ==============
xNumFirstGridElems = 2;
yNumFirstGridElems = 2;
yNumElemOld = yNumElem;
xNumElemOld = xNumElem;
xNumElem = xNumFirstGridElems*2^(i-1);  % number of x elements
yNumElem = yNumFirstGridElems*2^(i-1);  % number of y elements

BdiagVec = -4*ones(xNumElem-1,1);       % exclude two boundaries
BdiagVec(1) = -3+deltaX;                %add derivative boundary condition, at all
    i = 1 for bottom part
if (xNumElem-1 > 1)
    B_top = gallery('tridiag',ones(xNumElem -2, 1),BdiagVec,ones(xNumElem -2, 1));
        % matrix of coefficients
else
    B_top = BdiagVec;
end

for k=1: yNumElem
    A = blkdiag(A, B_top);              %populate A matrix with B matrix as
        diagonal
end
% Now loop through the A matrix to add I matrix wherever necessary
for k= (yNumElemOld-1)*(xNumElemOld-1) + 1: (yNumElemOld-1)*(xNumElemOld-1) +
    (xNumElem - 1)*(yNumElem)
    if (k + xNumElem -1 <= size(A, 2))
        A(k, k + xNumElem -1) = 1;
    end
    if (k > (yNumElemOld-1)*(xNumElemOld-1) + (xNumElem - 1))
        A(k, k -( xNumElem -1)) = 1;
    end
end
    % ======================================================================
```

3. Step 3: Even though we added two matrices, we haven't told them about the boundary they are sharing in the common area. if we don't do that, it will take zero boundaries (current situation) which will be false. So we add these common boundary to matrix.

```
% i.e. in the sheet, we haven't considered the fact that there is no boundary
% between this top and bottom plate upto a point. right now the matrix assumes,
    zero boundary condition.
% to fix this we can link by following code
for k= (xNumElemOld - 1)*(yNumElemOld-2): (xNumElemOld - 1)*(yNumElemOld-2) +
    (xNumElem - 1)
    if k > 0
        A(k, k + xNumElemOld -1) = 1;
    end
```

```
end
for k= (xNumElemOld - 1)*(yNumElemOld-1) + 1: (xNumElemOld - 1)*(yNumElemOld-1) +
    xNumElem -1
    A(k, k - (xNumElemOld -1)) = 1;
end
% Now the matrix is complete for the given plate with boundary condition as zero
    along boundaries.
```

Now we apply first boundary condition.
for $k = 1$ and $\alpha = 1$.

1. $\phi_{n-1} - (\Delta x + 1)\phi_n = 0$ at outer boundaries $i = 1$ and $j = 1$; using backward difference.
2. $\phi_{n+1} - (\Delta x + 1)\phi_n = 0$ at outer boundaries $i = n - 1$ and $j = m - 1$; using forward difference.

This can be included by adding $(1 + \Delta x)$ to -4 cells in the matrix where the boundary conditions apply. in code we can loop for these entries and add $(1 + \Delta x)$ to -4's:

1. At boundaries where $y = 1$:

```
% We now add boundary conditions one by one

 % Now add derivative boundary condition at j=1
 y = 1;
 for index = y:y + (xNumElemOld-2)
     id = find(A(index, :) == -4);
     A(index, id) = -3 + deltaX;
 end
```

2. At boundaries where $y = m - 1$:

```
% Now add derivative boundary condition at j=m-1
y = (xNumElemOld - 1)*(yNumElemOld-1) + (xNumElem -1)*(yNumElem-1) + 1;
for index = y :y + (xNumElem-2)
    id = find(A(index, :) == -4);
    A(index, id) = -3 + deltaX;
end
```

3. Now, always three corners will have terms coming from both boundaries conditions e.g. i=1 and j=1 or i=1, j=m-1 etc.. to solve that:

```
% 3 corner points, where two boundary terms will come
A(1, 1) = A(1, 1) + (1+deltaX);
A(xNumElemOld -1, xNumElemOld -1) = A(xNumElemOld -1, xNumElemOld -1) + (1+deltaX);
A(size(A,1)-(xNumElem-2), size(A,1)-(xNumElem-2)) = A(size(A,1)-(xNumElem-2),
    size(A,1)-(xNumElem-2)) + (1+deltaX);
clear('y', 'id', 'index');
```

Now second boundary condition, the temperatures of inner walls, can be included by adding constants to vector **F** on the right hand side.
new **F** is created by

```
% initialize F
F = zeros(size(A,1),1);
% assign -100 to points corresponding to ones near inner boundary
for k=(yNumElemOld-2)*(xNumElemOld-1)+
     xNumElem:(yNumElemOld-2)*(xNumElemOld-1)+xNumElemOld-1
    F(k) = -100;
end

for y=1:yNumElem
    id = (yNumElemOld-1)*(xNumElemOld-1)+ y*(xNumElem-1);
    F(id) = -100;
end
```

Invert the matrix to find unknown temperatures

```
% Now we have all the required matrices. we just need to invert A matrix to get temperature
    distribution
phiVector{i} = inv(A)*F;
```

1. Now we need to convert that row matrix into a 2D matrix for contour potting

```
% Now we need to convert that row matrix into a 2D matrix for contour potting.
% It is done by
for y=1:size(phi{i}, 1)
    for x=1:size(phi{i}, 2)
        % update interior points
        if ((y < yNumElemOld + 1) && (y >1))
            phi{i}(y, 2: xNumElemOld) = phiVector{i}((y-1)*(xNumElemOld-1) +1:
                y*(xNumElemOld-1) );
        end

        % update internal boundary
        if (y > yNumElemOld)
            phi{i}(y, 2: xNumElem) = phiVector{i}((y-1)*(xNumElem-1) +1:
                y*(xNumElem-1) );
            if (x == xNumElem +1)
                phi{i}(y, x) = -100;
            end
        end
        % update internal boundary
        if (y == yNumElemOld)
            phi{i}(y, xNumElem+1: end) = -100;
        end
    end
end
```

2. update outer boundaries

```
% update outer boundaries
for y=1:size(phi{i}, 1)
    phi{i}(y, 1) = (1+deltaX)* phi{i}(y, 2);
    if y<yNumElemOld
        phi{i}(y, xNumElemOld + 1) = (1+deltaX)* phi{i}(y, xNumElemOld);
    end
```

```
end

% update outer boundaries
for x=1:size(phi{i}, 2)
    phi{i}(1, x) = (1+deltaX)* phi{i}(2, x);
end
```

3. Since we divided sheet to 4, remake that full matrix

```
phi{i} = horzcat(phi{i}, flip(phi{i}(:, 2:end), 2));
phi{i} = vertcat(phi{i}, flip(phi{i}(2:end, :)));
```

4. Plot contour

```
% grid points
xArray{i} = 0:2*deltaX:6;
yArray{i} = 0:2*deltaX:8;

% plot the contour
figure;
[X,Y] = meshgrid(xArray{i},yArray{i});
contour(X, Y, phi{i}, 30);
```