APPLYING FEM TO COMPUTE THE TEMPERATURE FIELD ABOUT A CYLINDER CONTAINING AN ENERGY SOURCE

Changjin Song and Jiang Zhu

Department of Mechanical and Materials Engineering, University of Western Ontario, Canada

ABSTRACT

Finite Element Method (FEM) is a very powerful tool for solving research and practical problems. In this paper, FEM has been used to compute the temperature field about a cylinder which contains an energy source by taking into account the effect of the fluid which passes by it. Matlab software is utilized to build up the mesh of the domain and to do the computation program. Analysis is given at each step of the whole computation procedure. And the result is well illustrated that demonstrates FEM is quite useful to solve fairly complicated problems. For future study for this particular problem, the effect of temperature to the fluid can be considered, that is, the fluid is viscous and compressible. And there should be iteration computation for this case.

PROBLEM DEFINITION

The problem in this project is actually seeking the solution of a partial differential equation. Fig. 1 shows the illustration of the problem. The fluid enters on the left with a uniform velocity and temperature. It flows around the pipe shown, which contains an energy source producing q units of energy per surface area of the pipe per unit time.

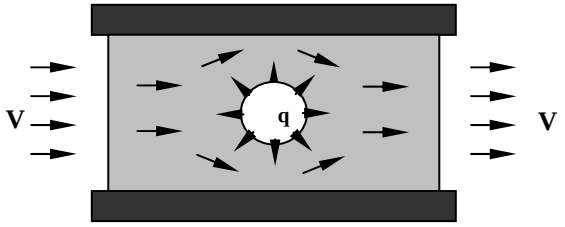

Fig. 1 Problem definition

Obviously, the temperature field could depend on the velocity field of the fluid. Although the velocity field could depend on the temperature field due to a temperature-dependent viscosity as well, we just assume the fluid to be nonviscous and incompressible, that is, unaffected by the temperature field. In this case, the flow can be completely determined by using the stream function (1), where u and v inside are specified by (2), (3).

$$\frac{\partial^2}{\partial x^2}[\Psi]+\frac{\partial^2}{\partial y^2}[\Psi]=0 \qquad (1)$$

$$u = \frac{\partial}{\partial y}[\Psi] \qquad (2)$$

$$v = -\frac{\partial}{\partial x}[\Psi] \qquad (3)$$

Once the flow is determined, the temperature field can be determined by (4)

$$\frac{\partial}{\partial x}\left(k\frac{\partial \Phi}{\partial x}\right)+\frac{\partial}{\partial x}\left(k\frac{\partial \Phi}{\partial y}\right)-\rho C_p u_x\frac{\partial \Phi}{\partial x} - \rho C_p u_y\frac{\partial \Phi}{\partial y}=0.0 \qquad (4)$$

where
Φ = temperature,
K = thermal conductivity of the fluid
ρC_p = heat capacity of the fluid
u_x = x component of velocity
u_y = y component of velocity

MESH GENERATION

When using FEM, mesh over the domain has to be generated. Firstly, the dimension of the domain and the diameter of the pipe are illustrated in Fig. 2.

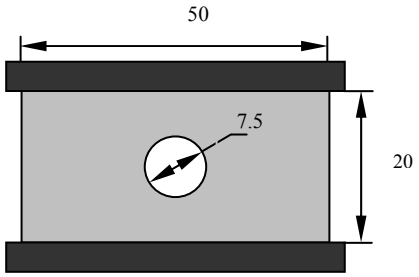

Fig. 2 Dimension of the domain

Due to the axial symmetry about the horizontal centerline, only half of the domain (which is divided into 3 LOOPS, shown as Fig. 3) need be used in the analysis.

Each LOOP can be obtained by assigning 8 points on its four sides. Also, **LOOPS** and **JOIN** Data files need to be modified. And then the mesh (Fig. 4) of 2376 elements and 2507 nodes of this domain are generated by the mesh generator **code mesh.m** from the text book using the specified **COORD** Data file.

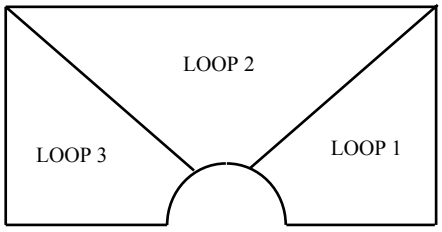

Fig. 3 Domain is divided into 3 LOOPS

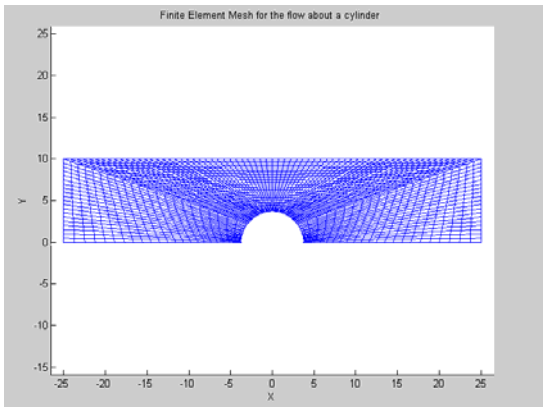

Fig. 4 Finite element mesh for the flow about a cylinder (2376 elements and 2507 nodes)

FLOW ANALYSIS

The streamlines have the property that flow perpendicular to a streamline is zero. So, the fixed walls correspond to streamlines. The fact that the velocity component perpendicular to the horizontal line of symmetry is equal to zero allows us to use that line as a streamline. Since the velocity field depends on the relative difference of two streamlines, we take the value of the streamline that coincides with the axis of symmetry of the cylinder to be zero. (Ψ=0). And then determine the value of Ψ on the upper wall from the condition

$$\frac{\partial \Psi}{\partial y} = V$$

where V is the velocity of the fluid parallel to streamline. Then,

$$\Psi = yV + C$$

If we take constant C equal to Zero, then

$$\Psi = yV$$

From this analysis, we can get the boundary conditions of each side of the domain. Figure 5 shows the boundary conditions on the stream function for V=1.0.

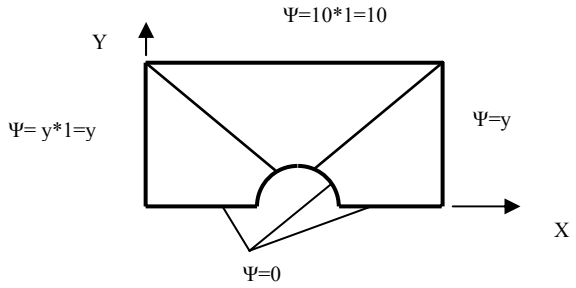

Fig.5 Boundary conditions for stream function

Use the above boundary conditions to make **INITIAL.m** code**.** Then modify **NPcode**.**m** to set NPcode on each side of the domain include the half circle. **MESHo, NODES** and **NP** can be obtained directly from **mesh.m**. **COEF** file is also needed to be modified. And then run **newnum.m** to get **NWLD** Data file. In our case, we use node 1 and node 22 of the first LOOP in the first wave when running newnum.m. Now, it is time to run **steady.m** for the stream function. The result for V=1.0 is plotted as shown in Fig. 6 using **topo.m**.

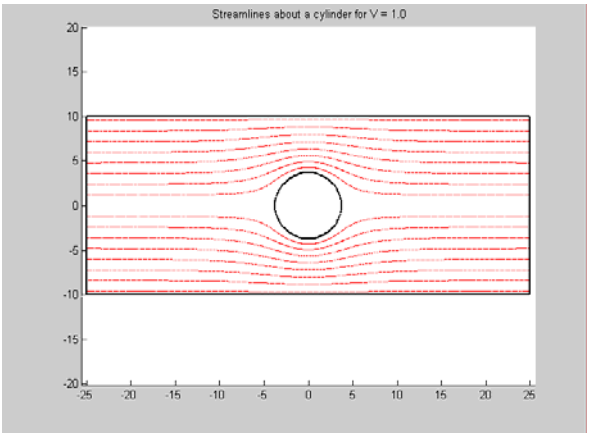

Fig. 6 Streamlines about a cylinder for V = 1.0

TEMPERATURE ANALYSIS

As we mentioned before, the temperature field can be determined by equation (4). To obtain the velocities u_x and u_y in this equation, we save the PHI values get from the flow analysis by **steady.m** in a Data file VP. Then load these values in the **INITIAL.m** code. Having these values, the velocity components can be obtained using (5) and (6).

$$u_x = \frac{\partial \Psi}{\partial y} = \lfloor DNDY \rfloor \{VP\} \qquad (5)$$

$$u_y = -\frac{\partial \Psi}{\partial y} = \lfloor -DNDY \rfloor \{VP\} \qquad (6)$$

And the boundary conditions are shown in Fig. 7.

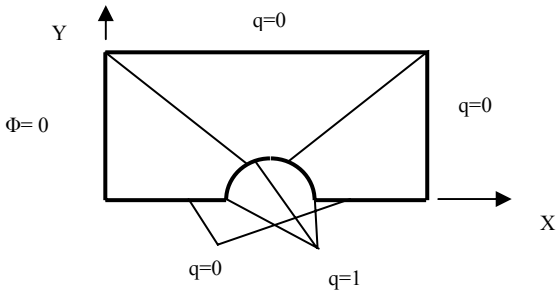

Fig. 7 Boundary conditions for temperature field

In this project, we assume that the thermal conductivity of the fluid k and heat capacity of the fluid ρC_p are of unit values. Then, run the **steady.m** code for the second time with the new **INITIAL.m** and **COEF.m** to get temperature Φ.

Fig.8~ Fig 11 show the temperature field for entrance velocities corresponding to 0.0, 0.3, 0.6 and 1.

CONCLUSIONS

1. From the result we can see that the temperature will decrease when the entrance velocity increases.

2. Even the entrance velocity is zero; the heat will be transferred towards the exit.

3. The result of the computation is pretty good by using FEM and the simulation looks perfect. Therefore, FEM can be very useful for solving some complex problems.

4. For further study, the effect of temperature to the fluid can be considered, that is, the fluid is viscous and compressible. And there should be iteration computation for this case.

REFERENCES

1. Erik. G. Thompson, INTRODUCTION TO THE FINITE ELEMENT METHOD- Theory, Programming and Application, WILEY, 2005
2. J. N. Reddy, AN INTRODUCTION TO THE FINITE ELEMENT METHOD, McGraw-Hill Book Company, 1984
3. Darrell W. Pepper and Juan C. Heinrich, The Finite Element Method-Basic Concepts and Applications, HEMISPHERE PUBLISHING CORPORATION, 1992

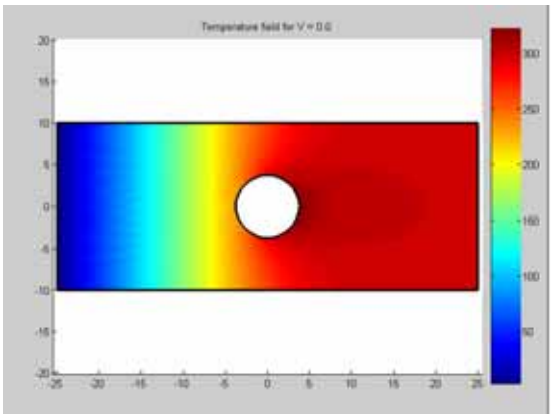

Fig. 8 Temperature field for V = 0.0

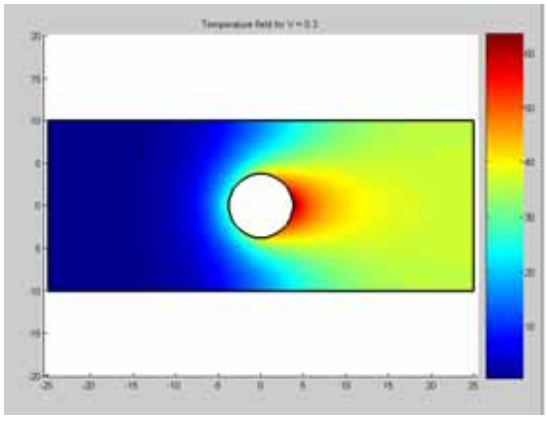

Fig. 9 Temperature field for V = 0.3

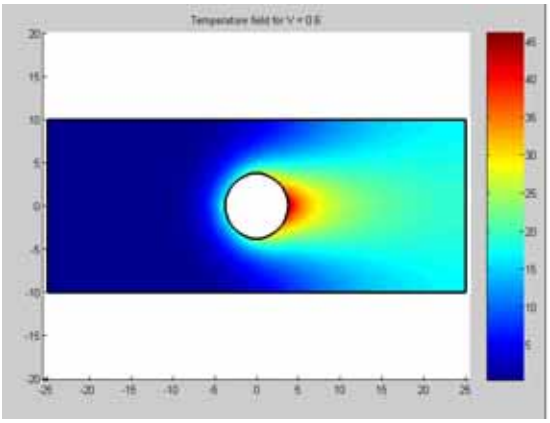

Fig. 10 Temperature field for V = 0.6

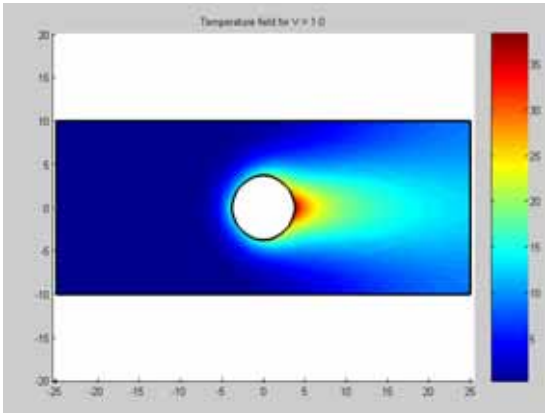

Fig. 11 Temperature field for V = 1.0

Appendix: User-Provided Codes:

For the stream function:

COORD:

```
%    LOOP Coordinates (8 per loop)
%-----------------------------------------------
     25         0
     25         5
     25         10
     15         6.47727
     3          2.25
     3.6404   0.9
     3.75       0
     12.5       0
%-----------------------------------------------
     3          2.25
     15         6.47727
     25         10
     0          10
   -25         10
   -15         6.47727
   -3          2.25
     0          3.75
%-----------------------------------------------
   -3.75       0
   -3.6404    0.9
   -3          2.25
   -15         6.47727
   -25         10
   -25         5
   -25         0
   -12.5       0
```

JOIN ARRAY:

```
     0 0   0 0   0 0   0 0
     1 2   0 0   0 0   0 0
     0 0   2 3   0 0   0 0
```

LOOPS:

```
%-----------------------
%    NUMLPS    NNPE
%-----------------------
         3      4
%-----------------------
%    NDIV-1    NDIV-2
%-----------------------
         20         22
         22         68
         20         22
     %    6          4
```

COEF.m:

```
%-------------------------------------------------------*
%   d    d@    d    d@      d@     d@                    *
%   --(RX--) + --(RY--) + BX-- + BY-- + G@ + HV = 0     *
%   dx   dx    dy   dy      dx     dy                    *
%                                                       *
%-------------------------------------------------------*
%-------------------------------------
%   Coefficients for solution to be:
%
%-------------------------------------

   RXJ = 1;
   RYJ = 1;

   BXJ=0;
   BYJ=0;

   GVJ = 0;
   HVJ = 0;
```

INITIAL.m:

```
%--------------------------------
%    INITIAL.m
%
%    Set problem parameters.
%--------------------------------

     for Ix=1:NUMNP
      if NPcode(Ix) == 1
                  NPBC(Ix)=1;
                   PHI(Ix) = YORD(Ix);
       elseif NPcode(Ix)== 2
          NPBC(Ix)=1;
          PHI(Ix)=10;
       elseif NPcode(Ix)== 3
          NPBC(Ix)=1;
          PHI(Ix)= YORD(Ix);
       elseif NPcode (Ix) == 4
          NPBC (Ix)=1;
          PHI (Ix) = 0;
       elseif NPcode (Ix) == 5
          NPBC (Ix)=1;
          PHI (Ix) = 0;
       else
              NPBC(Ix)=0;
       end
 end
```

NPCODE.m:

```
%---------------------------------
% NPCODE.m
%
% A user INCLUDE code
%
% LNP(I,J,K) = node K, on side J
% of element I.
%-------------------------------------
% --------------------------------
% Set n-factor for number of nodes
% on a side
% --------------------------------
  if NNPE == 6 | NNPE == 8
     n=2;
  else
```

```
    n=1;
  end

% ------------------------
% Initialize NPcode array
% ------------------------
  for i=1:NUMNP
    NPcode(i)=0;
  end

%  ----------------------------------
%  Set NPcode = 1 on  right side boundary
%  ----------------------------------
  IEND = n*NDIV(1,1)+1;
  for I=1:IEND
    NI=LNP(1,1,I);
    NPcode(NI)=1;
  end



%  ----------------------------------
%  Set NPcode = 2 on top side of mesh
%  ----------------------------------
  IEND = n*NDIV(2,2)+1;
  for I=1:IEND
    NI=LNP(2,2,I);
    NPcode(NI)=2;
  end
%  ----------------------------------
%  Set NPcode = 3 on left side boundary
%  ----------------------------------
  IEND = n*NDIV(3,3)+1;
  for I=1:IEND
    NI=LNP(3,3,I);
    NPcode(NI)=3;
  end

%  ----------------------------------
%  Set NPcode = 4 on lower side of mesh
%  ----------------------------------
  IEND = n*NDIV(1,4)+1;
  for I=1:IEND
    NI=LNP(1,4,I);
    NPcode(NI)=4;
  end

  IEND = n*NDIV(3,4)+1;
  for I=1:IEND
    NI=LNP(3,4,I);
    NPcode(NI)=4;
  end


 %%%%%%%%
 % Set NPcode for the circle
 %%%%%%%%


IEND =n*NDIV(1,3)+1;
  for I= 1:IEND
     NI=LNP(1,3,I);
     NPcode(NI)=5;
  end
   IEND =n*NDIV(2,4)+1;
  for I= 1:IEND
    NI=LNP(2,4,I);
    NPcode(NI)=5;
  end
   IEND =n*NDIV(3,1)+1;
  for I= 1:IEND
    NI=LNP(3,1,I);
    NPcode(NI)=5;
  end
```

For the temperature field:

INITIAL.m:

```
%--------------------------------
%    INITIAL.m
%
%    Set problem parameters.
%--------------------------------

 load VP    -ASCII
     for Ix=1:NUMNP
       if NPcode(Ix) == 1
                  NPBC(Ix)=0;

             Q(Ix)=0;
       elseif NPcode(Ix)== 2
          NPBC(Ix)=0;

          Q(Ix)=0;
       elseif NPcode(Ix)== 3
          NPBC(Ix)=1;
          PHI(Ix)= 0;
       elseif NPcode (Ix) == 4
          NPBC (Ix)=0;

          Q(Ix)=0
       elseif NPcode (Ix) == 5
          NPBC (Ix)=0;
          Q (Ix) = 1;
       else
              NPBC(Ix)=0;
       end
 end
```

COEF.m:

```
%-------------------------------------------------------*
%                                        *
%   d    d@    d    d@      d@     d@                *
%   --(RX--) + --(RY--) + BX-- + BY-- + G@ + HV = 0    *
%   dx   dx    dy   dy      dx     dy                *
%                                        *
%-------------------------------------------------------*

   RXJ = 1;
   RYJ = 1;
```

```
   GVJ = 0;
   HVJ = 0;
```

Modified code in steady.m:

```
……
%       ----------------------------------------------------
%       Determine derivative of shape functions in X-Y
plane
%       ----------------------------------------------------
          UX=0;
          UY=0;
        for K=1:NNPE;

DNDX(K)=RJACI(1,1)*SF(2,K,J)+RJACI(2,1)*SF(3,K,J);

DNDY(K)=RJACI(1,2)*SF(2,K,J)+RJACI(2,2)*SF(3,K,J);
          %%%%
          %ADDED PROGRAM
          %%%%%%%
          NPK=NP(I,K);

          UX=UX+DNDY(K)*VP(NPK)*0.3;
          UY=UY-DNDX(K)*VP(NPK)*0.3;
        end

%       ---------------------------------
%       Include user written coefficients
%       RXJ, RYJ, BXJ, BYJ, GVJ, HVJ
%       ---------------------------------
        COEF

        BXJ(I)=-UX;
        BYJ(I)=-UY;
……
```