

```
1 program problem
2
3     use iso_fortran_env, only: dp=>real64
4     use linspace_mod
5     use tdma
6
7     implicit none
8     !output utility
9     integer, parameter :: outfile = 16
10    integer, parameter :: outfile2 = 17
11
12    !enter problem you want to solve here
13    !problem6
14    !variable declarations
15    integer :: i,j,n,m,count,p,q
16    real(dp), dimension(:,,:),allocatable ::
17        phi,phi_dash,gmae,gmaw,gman,gmas,Del_x,Del_y,delxe,delxw,delyn,delys,sc,sp,theta
18    real(dp), dimension(:,),allocatable :: z
19    character(2) :: sweep
20    real(dp) :: error,quee,quew,quen,ques,De,hippq,omega,Tb,U,gmac,Tmw,Nu,quec
21
22    !defining the problem
23    !diffusion coefficient (comment out if not constant) can be set up later as a cell property
24    gmac = 10.0_dp
25    !velocity of the free stream
26    U = 1.0_dp
27    !boundary heat flux que: e,w,n,s for the respective directions, c for common
28    quec = 1000.0_dp
29    quee = quec
30    quew = quec
31    quen = quec
32    ques = quec
33    !side of the square
34    De = 1.0_dp
35    !underrelaxation parameter
```

```
35     omega = 0.5_dp
36
37     !inputs
38     write(*,*) "Enter number of rows"
39     read(*,*) n
40     write(*,*) "Enter number of columns"
41     read(*,*) m
42     write(*,*) "Choose the sweep direction"
43     read(*,*) sweep
44     write(*,*) "Choose the x-coordinate for the fixed point, must be inner"
45     read(*,*) p
46     write(*,*) "Choose the y-coordinate for the fixed point, must be inner"
47     read(*,*) q
48     write(*,*) "Choose the phi value taken by the fixed point"
49     read(*,*) phipq
50
51     !allocate dynamic arrays
52     allocate(phi(n,m),phi_dash(n,m),theta(n,m),gmae(n,m),gmaw(n,m),gman(n,m),gmas(n,m),Del_x(n,m),Del_y
        (n,m),delxe(n,m),delxw(n,m),delyn(n,m),delys(n,m),sc(n,m),sp(n,m),z(m))
53
54     !initialize count
55     count = 0
56
57     !initialize cell properties here: An example is shown here but the array can be initialized in any way
58     !does not have to be a uniform mesh, but has to be a structured mesh
59     !>>diffusion coefficient
60     gmae = gmac
61     gmaw = gmac
62     gman = gmac
63     gmas = gmac
64     !>>cell dimensions
65     Del_x = De/(1.0_dp*(m-2))
66     Del_y = De/(1.0_dp*(n-2))
67     delxe = De/(1.0_dp*(m-2))
68     delxw = De/(1.0_dp*(m-2))
```

```
69     delyn = De/(1.0_dp*(n-2))
70     delys = De/(1.0_dp*(n-2))
71
72     !initialize the matrix
73     phi = 100.0_dp
74
75     !setting up cell dimensions for the boundary
76     do i = 1,m
77         !south boundary
78         delys(2,i) = 0.5_dp*delys(2,i)
79         !north boundary
80         delyn(n-1,i) = 0.5_dp*delyn(n-1,i)
81     end do
82     do i = 1,n
83         !west boundary
84         delxw(i,2) = 0.5_dp*delxw(i,2)
85         !east boundary
86         delxe(i,m-1) = 0.5_dp*delxe(i,m-1)
87     end do
88
89     !main loop start
90     !source term incorporation
91     do while (.true.)
92         !initialize error for source term loop
93         error = 0.0_dp
94         !count to keep track of iterations
95         count = count +1
96         print*,count, 'outer'
97         !backup array
98         phi_dash = phi
99         !declare source term here
100        sc = (-4.0_dp*quec/De)
101        sp = 0.0_dp
102        !call newtdma2d solver to solve your problem
103        call newtdma2dinmat
```

```

      (n,m,phi,gmae,gmaw,gman,gmas,Del_x,Del_y,delxe,delxw,delyn,delys,sc,sp,p,q,hipq,quee,quew,quen,
      ques,omega,sweep)
104      !error calculation using eucleadian distance
105      error = sqrt(sum((phi-phi_dash)**2))
106      !exit condition for the main loop
107      if (error<1e-10) exit
108  end do
109
110      !print phi to terminal
111      do i = 1,n
112          print*,phi(i,:)
113      end do
114
115      !writing phi to a file
116      open(unit=outfile, file = 'phifor.dat', access='sequential',action = 'write')
117      do i=1,n
118          write (outfile,*) (phi(i,j), j=1,m)
119      end do
120      close(outfile)
121
122      !bulk temperature
123      Tb = U*sum(phi*Del_y*Del_x)/(U*sum(Del_y*Del_x))
124      print*,"The bulk temperature for the given problem is", Tb
125
126      !dimensionless temperature
127      theta = (phi-Tb)/(quee*De/gmac)
128
129      !in case you want to solve for the easiest test case, que = 0
130      !theta = (phi-Tb)/Tb
131
132      !midline in z dxn
133      z(2:m-1) = linspace(-1.0_dp*Del_x(5,5)*(m-3)/2,Del_x(5,5)*(m-3)/2,m-2)
134      z(1) = z(2)-(Del_x(5,5)/2)
135      z(m) = z(m-1)+(Del_x(5,5)/2)
136

```

```
137      !writing midline temperature
138      open(unit=outfile2, file = 'midtemp.dat', access='sequential',action = 'write')
139
140      if(mod(n,2)==1) then
141          do i=1,m
142              write (outfile2,*) z(i),theta((n+1)/2,i)
143          end do
144      end if
145      if(mod(n,2)==0) then
146          do i=1,m
147              write (outfile2,*) z(i),0.5_dp*(theta((n/2),i)+theta((n/2)+1,i))
148          end do
149      end if
150
151      close(outfile2)
152
153      !mean wall temperature
154      Tmw = 0.0_dp
155      do i = 1,n
156          Tmw = Tmw+phi(i,1)+phi(i,m)
157      end do
158      do i = 1,m
159          Tmw = Tmw+phi(1,i)+phi(n,i)
160      end do
161      Tmw = Tmw - phi(1,1) - phi(1,m) - phi(n,1) - phi(n,n)
162      Tmw = Tmw/(2*(m+n)-4)
163      print*, "The mean wall temperature is: ", Tmw
164
165      !Nusselt number
166      Nu = (quec*De/gmac)/(Tmw-Tb)
167      print*, "The nusselt number is: ", Nu
168
169      !deallocate the dynamic arrays
170      deallocate(phi,phi_dash,theta,gmae,gmaw,gman,gmas,Del_x,Del_y,delxe,delxw,delyn,delys,sc,sp,z)
171
```

172

173

174 end program problem

175