```
 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      !problem 2
14      !added feature to check for error from actual solution
15      !we have a square of given side
16      !the phi function we are supposed to be getting is x^2-y^2
17      !boundary conditions are set accordingly
18
19      !declarations
20      integer :: i,j,n,m,count
21      real(dp), dimension(:,:),allocatable ::                                  ↵
          phi,phi_dash,gmae,gmaw,gman,gmas,Del_x,Del_y,delxe,delxw,delyn,delys,sc,sp,phi_exp
22      real(dp), dimension(:),allocatable :: y
23      real(dp), dimension(:),allocatable :: x
24      real(dp) :: error,De
25      character(2) :: sweep
26
27      !side of the squeare
28      De = 1.0_dp
29
30      !input for grid size and sweep direction
31      write(*,*) "Enter number of rows"
32      read(*,*) n
33      write(*,*) "Enter number of columns"
34      read (*,*) m
```

```
35      write(*,*) "Choose the sweep direction"
36      read(*,*) sweep
37
38      !allocate arrays
39      allocate(x(m),y(n),phi(n,m),phi_dash(n,m),phi_exp(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))
40
41      !initialize count
42      count = 0
43
44      !problem defination
45      !initialize cell properties here
46      gmae = 1.0_dp
47      gmaw = 1.0_dp
48      gman = 1.0_dp
49      gmas = 1.0_dp
50      !>>cell dimensions
51      Del_x = De/(1.0_dp*(m-2))
52      Del_y = De/(1.0_dp*(n-2))
53      delxe = De/(1.0_dp*(m-2))
54      delxw = De/(1.0_dp*(m-2))
55      delyn = De/(1.0_dp*(n-2))
56      delys = De/(1.0_dp*(n-2))
57
58      !setting up cell dimensions for the boundary
59      do i = 1,m
60          !south boundary
61          delys(2,i) = 0.5_dp*delys(2,i)
62          !north boundary
63          delyn(n-1,i) = 0.5_dp*delyn(n-1,i)
64      end do
65      do i = 1,n
66         !west boundary
67         delxw(i,2) = 0.5_dp*delxw(i,2)
68         !east boundary
```

```
69          delxe(i,m-1) = 0.5_dp*delxe(i,m-1)
70      end do
71
72      !initializing boundary conditions
73      !initialize the 2D array phi here.
74
75      x(1) = 0.0_dp
76      x(2:m-1) = linspace(0.0_dp+(Del_x(1,1)/2),1.0_dp-(Del_x(1,1)/2),m-2)
77      x(m) = 1.0_dp
78      y(1) = 0.0_dp
79      y(2:n-1) = linspace(0.0_dp+(Del_y(1,1)/2),1.0_dp-(Del_y(1,1)/2),n-2)
80      y(n) = 1.0_dp
81
82      phi = 0.0_dp
83      do i = 1,m
84          phi(1,i) = x(i)**2
85          phi(n,i) = x(i)**2-1.0_dp
86      end do
87      do i = 1,n
88          phi(i,1) = -1.0_dp*(y(i)**2)
89          phi(i,m) = 1.0_dp - (y(i)**2)
90      end do
91
92      !expected solution
93      do i = 1,n
94          do j = 1,m
95              phi_exp(i,j) = x(j)**2-y(i)**2
96          end do
97      end do
98
99
100     !print input
101     do i = 1,n
102         print*, phi(i,:)
103     end do
```

```
104
105     !main loop start
106     !source term incorporation
107     do while (.true.)
108         !initialize error for source term loop
109         error = 0.0_dp
110         !count to keep track of iterations
111         count = count +1
112         print*,count, 'outer'
113         !backup array
114         phi_dash = phi
115         !declare source term here
116         sc = 0.0_dp
117         sp = 0.0_dp
118        !call twodtdma solver to solve your problem
119         call twodtdma(n,m,phi,gmae,gmaw,gman,gmas,Del_x,Del_y,delxe,delxw,delyn,delys,sc,sp,sweep)
120         !error using eucleadian distance
121         error = sqrt(sum((phi-phi_dash)**2))
122         !exit condition
123         if (error<1e-10) exit
124     end do
125
126     !print output
127     do i = 1,n
128        print*, phi(i,:)
129     end do
130
131     !print difference
132     print*,"difference from exact solution"
133     do i = 1,n
134        print*, phi_exp(i,:)-phi(i,:)
135     end do
136
137     !deallocate all the arrays
138     deallocate(x,y,phi,gmae,gmaw,gman,gmas,Del_x,Del_y,delxe,delxw,delyn,delys,sc,sp,phi_dash,phi_exp)
```

```
139
140
141
142
143  end program problem
144
```