

Source Code: heateq_simple.f90

```
1 program example
2   ! Example 10.3 from NMUM/Mathews.
3   ! Integrate the 1D heat equation forward in time given simple initial
4   ! conditions. This is a simple example of how to use Fortran90. There
5   ! are better ways to code this problem...
6
7   ! Begin the declaration section.
8   ! ALWAYS start with implicit none.
9   implicit none
10
11  ! Declare some variables. Set initial values on some.
12  ! Note that xLim is a 2-element vector.
13  real    :: dt=0.02, dx=0.2, tLimit=0.2, xLim(2)=(/0,1/)
14  real    :: r
15  integer :: nX, nT, i, j
16  logical :: DoTest = .true.
17
18  ! Create character variables. We must declare their size.
19  character(len=23) :: fmt1
20  character(len=17) :: fmt2
21
22  ! Domain array. We don't know the size yet because we need to
23  ! calculate that. So let's make them "allocatable."
24  real, allocatable :: Domain_II(:, :), xGrid(:)
25
26  ! Constants. Make them "parameters" that cannot be changed.
27  integer, parameter :: iUnitFile=10
28  real,    parameter :: cDiffusion = 1.0
29
30  ! Now, begin execution section.
31  !-----
32  ! Write a message to Standard Out with no defined format:
33  write(*,*) "Setting up simulation..."
34
35  ! Calculate the number of points in X and in time.
36  ! Ceiling rounds up and returns an integer, which matches the data
37  ! type of "ceiling". Without ceiling, our value would be a "real" type,
38  ! which may be rounded up, down, or truncated (compiler dependent).
39  nX = ceiling( (xLim(2)-xLim(1))/dx ) + 1
40  nT = ceiling( tLimit/dt ) + 1
41
42  ! If logical DoTest is .true., produce extra information to screen.
43  if(DoTest)write(*,*) '    Grid size (nX, nT) = ', nX, nT
44
45  ! Allocate arrays now that we know their size.
46  ! Remember: if we do not de-allocate, it's possible to create a mem leak.
47  allocate(Domain_II(nX, nT))
48  allocate(xGrid(nX))
49
50  !It's usually a good idea to fill matrices with zeros.
51  Domain_II = 0
52  xGrid     = 0
53
54  ! Set the grid values and initial conditions:
55  do i=1, nX
56    xGrid(i)      = (i-1) * dx
57    Domain_II(i,1) = 4.0*xGrid(i) - 4.0*xGrid(i)**2.0
58  end do
59
60  ! Check stability as described in class.
61  ! "if () then" means >1 line after if statement.
62  if ( dt > (dx**2.0 / (2.0*cDiffusion**2.0)) ) then
63    write(*,*) 'ERROR! WE ARE NOT STABLE!'
64    stop ! Remember, fortran's stop is not good for parallel programming.
65  end if
66
67  ! integrate. See notes from class on the meaning below.
68  write(*,*) 'Integrating...'
69  r = cDiffusion**2.0 * dt / dx**2.0
70  ! Loop from time 0 (j=1) to time t_final-deltaT.
71  do j=1, nT-1
72    Domain_II(2:nX-1, j+1) = (1.0 - 2.0*r) * Domain_II(2:nX-1, j) + &
73      r*(Domain_II(1:nx-2,j) + Domain_II(3:nx, j))
74  end do
75
76  ! Now we want to write our results to file.
77  write(*,*) 'Saving results to file.'
78
79  ! Start by opening file in replace mode (over write existing file).
80  ! Assign it to a file unit, iUnitFile.
81  open(iUnitFile, file='results.txt', status='replace')
```

```

82
83 ! Write a header line. Our write statement now writes to our
84 ! file unit and not "*" for standard out. We also use format codes
85 ! in place of our 2nd "*".
86 write(iUnitFile, '(a)') 'Example 10.3 Results.'
87
88 ! Write information about domain. Note the format code that fills in
89 ! brackets, commas, etc.
90 write(iUnitFile, "(a,['f3.0','f4.0','a,f5.1,a)") &
91 'Domain: x=',xLim, 't=[0.0,tLimit,']'
92 write(iUnitFile, "(a, i5.5, 'x',i5.5)") 'Domain size (x, Time) = ', nX, nT
93
94 ! Our next format code depends on the size of our domain, which
95 ! we won't know until run time. So, we'll write the format code
96 ! to a character variable of the right size:
97 write(fmt1, "(a, i6.6, a)") '(a13,', nX, '(1x, E12.6))'
98 if(doTest) write(*,*) 'fmt1 = ', fmt1
99 ! Write grid to file:
100 write(iUnitFile, fmt1) 'Grid:', xGrid
101
102 ! Create format code for time and result lines:
103 write(fmt2, "(a, i4.4,a)") '(', nX+1, '(1x, E12.6))'
104 if(doTest) write(*,*) 'fmt2 = ', fmt2
105 ! Loop over results and write to file.
106 do j=1, nT
107     write(iUnitFile, fmt2) (j-1)*dt, Domain_II(:,j)
108 end do
109
110 ! Close our file:
111 close(iUnitFile)
112
113 !Deallocate arrays. ALWAYS DO THIS FOR ALLOCATABLE ARRAYS!
114 deallocate(Domain_II)
115 deallocate(xGrid)
116
117 ! And that's it.
118 end program example

```