TRANSPORT PHENOMENA PROJECT REPORT
METALLURGICAL ENGINEERING AND MATERIALS SCIENCE DEPARTMENT
INDIAN INSTITUTE OF TECHNOLOGY, BOMBAY

By,
Swastika Agarwal(190110095)
Labdhi Gandhi (190110041)
Rutuja Thakur (190110077)

Group number - G19
Problem no. - 1

INDEX

Table of Contents

1. Problem Statement----------3

2. Abstract ----------3

3. Solution----------4

4. Code----------6

5. Result and Conclusion----------9

6. References----------10

PROBLEM STATEMENT

A wall 0.12m thick having a thermal diffusivity of 1.5 × 10-6 m2/s is initially at a uniform
temperature of 85°C. Suddenly one face is lowered to 20°C while the other face is perfectly insulated.
FIND: (a) Using the explicit finite-difference method with space and time increments of 30 mm and 300s, respectively, determine the temperature distribution at t = 45 min.
(b)With Δx = 30mm and Δt = 300s, compute T(x,t) for 0<=t<=t_{ss}, where t_{ss} is the time required for the temperature at each nodal point to reach a value that is within 1°C of the steady-state temperature. Repeat the foregoing calculations for Δt = 75 s. For each value of Δt, plot temperature histories for each face and the midplane.

ABSTRACT

In this project, we have plotted the temperature distribution as a function of both space and time across a thick wall using a 3d plot. For doing this we have used Python. We have first derived the finite difference equation for the given problem. Then we utilized this equation in our code to plot the temperature distribution for delta_T = 300s and delta_T = 75s. In this process we also arrived at a matrix depicting temperature values at each node at different time steps, which can be seen in the output when the code is run.

SOLUTION

ASSUMPTIONS :(1) One-dimensional transient conduction
(2) Constant properties.

ANALYSIS: Considering the two-dimensional system, under transient conditions with constant properties and no internal generation, the appropriate form of the heat equation

$$(1/\alpha)*(dT/dt)=(d^2T/dx^2) + (d^2T/dy^2)$$

To obtain the finite-difference form of this equation, we may use the central-difference approximations to the spatial derivatives.The integer *p* is introduced for this purpose

$$t= p\Delta t$$

finite-difference approximation to the time derivative is expressed as

$$dT/dt\,|_{m,n} = (T^{p+1}_{m,n}-T^{p}_{m,n})/\Delta t$$

The superscript p is used to denote the time dependence of T, and the time deriva- tive is expressed in terms of the difference in temperatures associated with the new (p+1) and previous (p) times. Hence calculations must be performed at successive times separated by the interval Δt, and just as a finite-difference solution restricts temperature determination to discrete points in space, it also restricts it to discrete points in time.

The explicit form of the finite-difference equation for the interior node m, n is

$$1/\alpha((T^{p+1}_{m,n}-T^{p}_{m,n})/\Delta t) = (T^{p}_{m+1,n}+T^{p}_{m-1,n}-2*T^{p}_{m,n})/(\Delta x^2) + (T^{p}_{m,n+1}+T^{p}_{m,n-1}-2*T^{p}_{m,n})/(\Delta y^2)$$

Solving for the nodal temperature at the new (p+1) time and assuming that $\Delta x=\Delta y$, it follows that explicit form of the finite-difference equation for an interior node m reduces to

$$T_m^{p+1} = F(T_{m-1}^{p}+T_{m+1}^{p})+(1-2*F)T_m^{p}$$

The stability criterion for a one- dimensional interior node is

$$(1 - 2F) >= 0, \text{ or}$$

$$F<=1/2$$

Therefore, the finite-difference equations for the interior points, nodes 0, 1, 2, and 3, can be determined from the equation given below

$$T_m^{p+1} = F(T_{m-1}^p + T_{m+1}^p) + (1-2F)T_m^p \quad \text{----------(1)}$$

$$F = (\alpha * \Delta t)/\Delta x^2 = 1.5*10^{-6}*300/0.03^2 = 0.5 \quad \text{----------(2)}$$

As, F<=0.5, the stability criterion is satisfied.

On combining eq (1) and eq (2), we get

$$T_m^{p+1} = 0.5(T_{m-1}^p + T_{m+1}^p) \text{ for } m = 0,1,2,3$$

Since the adiabatic plane at x = 0 can be treated a symmetry plane, $T_{m-1} = T_{m+1}$

The finite-difference solution is generated in the table

p	t(min)	T_0	T_1	T_2	T_3	T_L(°C)
0	0	85	85	85	85	20
1	5	85	85	85	52.5	20
2	10	85	85	68.75	52.5	20
3	15	85	76.875	68.75	44.375	20
4	20	76.875	76.875	60.625	44.375	20
5	25	76.875	68.75	60.625	40.3125	20
6	30	68.75	68.75	54.5312	40.3125	20
7	35	68.75	61.6406	54.5312	37.2656	20
8	40	61.6406	61.6406	49.4531	37.2656	20
9	45	61.6406	55.5469	49.4531	34.7266	20

The temperature distribution can also be determined from the one-term approximation of the exact solution. The insulated surface is equivalent to the midplane of a wall of thickness 2L. Thus,

$$F = (\alpha * t)/(L^2) = 1.5*10^{-6}*(45*60)/(0.12^2) = 0.28 \text{ and } Bi \rightarrow \infty.$$

From Table in Incorpera, $\zeta_1 = 1.5707$, $C_1 = 1.2733$

Putting the values in the equation, we get,

$$\theta_0^* = C_1 \exp(\zeta_1^2 * F) = 1.2733 \exp(-1.5707^2*0.28) = 0.64$$

$$T(0,t) = T_\infty + \theta_0^*(T_i - T_\infty) = 20°C + 0.64(85-20)°C = 61.5°C$$

This value shows excellent agreement with 61.7°C for the finite-difference method.

CODE

The code for our project is
For delta_T = 300 s

In [1]:

```
def fd1d_heat_explicit_alpha ( k, t_num, t_min, t_max, x_num, x_min, x_max ):
  from sys import exit
  x_step = ( x_max - x_min ) / ( x_num - 1 )
  t_step = ( t_max - t_min ) / ( t_num - 1 )
  alpha = k * t_step / x_step / x_step
  if ( 0.5 < alpha ):
    print ( '' )
    print ( 'FD1D_HEAT_EXPLICIT_alpha - Fatal error!' )
    print ( '  alpha condition failed.' )
    print ( '  0.5 < K * dT / dX / dX = %f' % ( alpha ) )
    exit ( 'FD1D_HEAT_EXPLICIT_alpha - Fatal error!' )
  return alpha
def fd1d_heat_explicit ( x_num, x, t, dt, alpha, bc, h ):
  import numpy as np
  h_new = np.zeros ( x_num )
  for c in range ( 1, x_num - 1 ):
    l = c - 1
    r = c + 1
    h_new[c] = h[c] + alpha * ( h[l] - 2.0 * h[c] + h[r] )
  h_new = bc ( x_num, x, t + dt, h_new, h[1] )
  return h_new
def fd1d_heat_explicit_test ( ):
  import matplotlib.pyplot as plt
  import numpy as np
  import platform
  from mpl_toolkits.mplot3d import Axes3D
  from matplotlib import cm
  from matplotlib.ticker import LinearLocator, FormatStrFormatter
  print ( '  one dimensional heat equation:' )
  print ( '' )
  print ( '    dH/dt - K * d2H/dx2 = 0' )
  print ( '' )
  k = 1.5*(10**(-6))
  x_num = 5
  x_min = 0.0
  x_max = 0.12
  dx = ( x_max - x_min ) / ( x_num - 1 )
  x = np.linspace ( x_min, x_max, x_num )
  t_num = 10
  t_min = 0.0
  t_max = 45*60
  dt = ( t_max - t_min ) / ( t_num - 1 )
  t = np.linspace ( t_min, t_max, t_num )
  alpha = fd1d_heat_explicit_alpha ( k, t_num, t_min, t_max, x_num, x_min, x_max )
  print ( '' )
  print ( '  Number of X nodes = %d' % ( x_num ) )
  print ( '  X interval is [%f,%f]' % ( x_min, x_max ) )
  print ( '  X spacing is %f' % ( dx ) )
  print ( '  Number of T values = %d' % ( t_num ) )
  print ( '  T interval is [%f,%f]' % ( t_min, t_max ) )
  print ( '  T spacing is %f' % ( dt ) )
  print ( '  Constant K = %g' % ( k ) )
  print ( '  alpha = %g' % ( alpha ) )
  hmat = np.zeros ( ( x_num, t_num ) )
  for j in range ( 0, t_num ):
    if ( j == 0 ):
      h = ic_test ( x_num, x, t[j] )
      h = bc_test ( x_num, x, t[j], h , h[1])
    else:
      h = fd1d_heat_explicit ( x_num, x, t[j-1], dt, alpha, bc_test, h )
    for i in range ( 0, x_num ):
```

```
      hmat[i,j] = h[i]
  tmat, xmat = np.meshgrid ( t, x )
  fig = plt.figure ( )
  ax = Axes3D ( fig )
  surf = ax.plot_surface ( xmat, tmat, hmat )
  plt.xlabel ( '<---X--->' )
  plt.ylabel ( '<---Time--->' )
  plt.title ( 'Temperature' )
  plt.savefig ( 'plot_test01.png' )
  plt.show ( block = False )
  r8mat_write ( 'h_test01.txt', x_num, t_num, hmat )
  r8vec_write ( 't_test01.txt', t_num, t )
  r8vec_write ( 'x_test01.txt', x_num, x )
  print ( '' )
  print ( '  H(X,T) written to "h_test01.txt"' )
  print ( '  T values written to "t_test01.txt"' )
  print ( '  X values written to "x_test01.txt"' )
  print('Temperature Matrix :- ')
  print(hmat)
  print ( '' )
  return
```

```
def bc_test ( x_num, x, t, h ,m):
  h[0]       = m
  h[x_num-1] = 20.0
  return h
def ic_test ( x_num, x, t ):
  import numpy as np
  h = np.zeros ( x_num )
  for i in range ( 0, x_num ):
    h[i] = 85.0
  return h
def r8mat_write ( filename, m, n, a ):
  output = open ( filename, 'w' )
  for i in range ( 0, m ):
    for j in range ( 0, n ):
      s = '  %g' % ( a[i,j] )
      output.write ( s )
    output.write ( '\n' )
  output.close ( )
  return
def r8mat_write_test ( ):
  import numpy as np
```

```
  import platform
  print ( '' )
  print ( 'R8MAT_WRITE_TEST:' )
  print ( '  Python version: %s' % ( platform.python_version ( ) ) )
  print ( '  Test R8MAT_WRITE, which writes an R8MAT to a file.' )
  filename = 'r8mat_write_test.txt'
  m = 5
  n = 3
  a = np.array ( (  \
    ( 1.1, 1.2, 1.3 ), \
    ( 2.1, 2.2, 2.3 ), \
    ( 3.1, 3.2, 3.3 ), \
    ( 4.1, 4.2, 4.3 ), \
    ( 5.1, 5.2, 5.3 ) ) )
  r8mat_write ( filename, m, n, a )
  print ( '' )
  print ( '  Created file "%s".' % ( filename ) )
  print ( '' )
  return
def r8vec_write ( filename, n, a ):
  output = open ( filename, 'w' )
```

```
  for i in range ( 0, n ):
    s = '  %g\n' % ( a[i] )
    output.write ( s )
  output.close ( )
  return
def r8vec_write_test ( ):
  import numpy as np
  import platform
  print ( '' )
  print ( 'R8VEC_WRITE_TEST:' )
  print ( '  Python version: %s' % ( platform.python_version ( ) ) )
  print ( '  Test R8VEC_WRITE, which writes an R8VEC to a file.' )
  filename = 'r8vec_write_test.txt'
  n = 5
  a = np.array ( ( 1.1, 2.2, 3.3, 4.4, 5.5 ) )
  r8vec_write ( filename, n, a )
  print ( '' )
  print ( '  Created file "%s".' % ( filename ) )
  print ( '' )
  return
if ( __name__ == '__main__' ):
```

```
  return
if ( __name__ == '__main__' ):
  fd1d_heat_explicit_test ( )
```

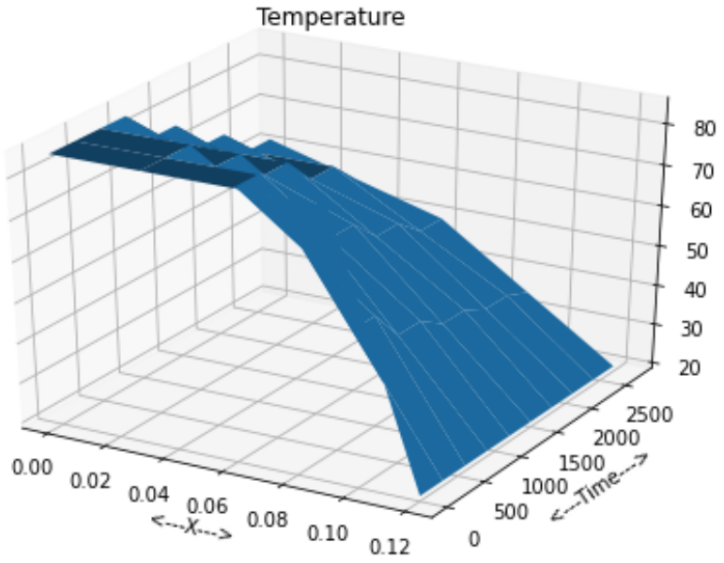
For delta_T = 75 s

The code is the same except that we have changed the value of delta_T to 75s.

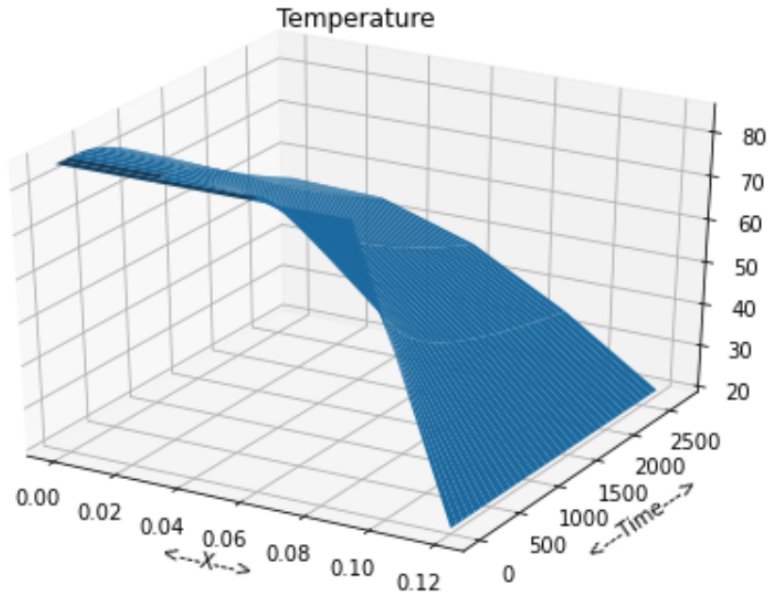
In code we have changed T_num from 10 to 37.

RESULTS AND CONCLUSION

The graph plotted for delta_T = 300s is shown below



The graph for delta_T = 75s is shown below



Temperature matrix for delta_T = 300s

```
  H(X,T) written to "h_test01.txt"
  T values written to "t_test01.txt"
  X values written to "x_test01.txt"
Temperature Matrix :-
[[85.        85.        85.        85.        76.875     76.875
  68.75      68.75      61.640625  61.640625 ]
 [85.        85.        85.        76.875     76.875     68.75
  68.75      61.640625  61.640625  55.546875 ]
 [85.        85.        68.75      68.75      60.625     60.625
  54.53125   54.53125   49.453125  49.453125 ]
 [85.        52.5       52.5       44.375     44.375     40.3125
  40.3125    37.265625  37.265625  34.7265625]
 [20.        20.        20.        20.        20.        20.
  20.        20.        20.        20.       ]]
```

Temperature matrix for delta_T = 75s

```
Temperature Matrix :-
[[85.         85.         85.         85.         84.87304688 84.58740234
  84.13909912 83.54103088 82.81291008 81.97633982 81.05219096 80.05938269
  79.01442911 77.93139613 76.82206237 75.69616801 74.56168839 73.42509942
  72.29161945 71.16542167 70.04981629 68.94740461 67.86020773 66.78977345
  65.73726451 64.7035311  63.68917042 62.69457534 61.71997423 60.76546336
  59.83103338 58.91659089 58.02197594 57.14697625 56.29133885 55.45477941
  54.63698987]
 [85.         85.         85.         84.87304688 84.58740234 84.13909912
  83.54103088 82.81291008 81.97633982 81.05219096 80.05938269 79.01442911
  77.93139613 76.82206237 75.69616801 74.56168839 73.42509942 72.29161945
  71.16542167 70.04981629 68.94740461 67.86020773 66.78977345 65.73726451
  64.7035311  63.68917042 62.69457534 61.71997423 60.76546336 59.83103338
  58.91659089 58.02197594 57.14697625 56.29133885 55.45477941 54.63698987
  53.83764461]
 [85.         85.         83.984375   82.4609375  80.71533203 78.90625
  77.11799622 75.39222717 73.74657869 72.18557596 70.70694573 69.30521168
  67.97369308 66.70557377 65.49443666 64.33449699 63.22067074 62.1485572
  61.11438086 60.11491745 59.1474179  58.20953666 57.29926762 56.41488827
  55.55491223 54.71804916 53.90317138 53.10928612 52.33551268 51.5810635
  50.84522874 50.12736349 49.42687737 48.74322593 48.07590366 47.42443823

 46.78838569]
[85.         76.875      70.78125    66.08398438 62.37060547 59.36737061
 56.8888092  54.80635643 53.02879572 51.48991913 50.14063634 48.94384547
 47.87103556 46.89998831 46.01318795 45.19669555 44.43933378 43.73208418
 43.06763279 42.4400222  41.84438133 41.27671323 40.73372701 40.21270371
 39.71138882 39.22790564 38.76068538 38.30841045 37.86996861 37.44441554
 37.03094459 36.62886204 36.23756696 35.85653489 35.48530441 35.12346627
 34.77065448]
[20.         20.         20.         20.         20.         20.
 20.         20.         20.         20.         20.         20.
 20.         20.         20.         20.         20.         20.
 20.         20.         20.         20.         20.         20.
 20.         20.         20.         20.         20.         20.
 20.         20.         20.         20.         20.         20.
 20.        ]]
```

REFERENCES

Book : Fundamentals of Heat and Mass Transfer by Incropera and Dewitt