Applying Finite-Difference Equations to find Nodal Temperatures

Gregory Dsouza

Embry-Riddle Aeronautical University

ES 403: Heat Transfer

Prof. Yang Chao

October 28, 2023

Applying Finite-Difference Equations to find Nodal Temperatures

Contents

Sets of Equations **3**

Interior Equations 3

Exterior Equations with Convection 3

Exterior Equations with Insulation 4

Tabulated Temperatures and Plots **4**

Explanation of Software Code **5**

Heat Conduction Analysis Code **7**

Sets of Equations

Interior Equations

$$T_b + T_{10} - 4.0T_9 + T_1 + T_{17} = 0$$

$$-4.0T_{10} + T_{11} + T_9 + T_{18} + T_2 = 0.0$$

$$T_{10} - 4.0T_{11} + T_{12} + T_{19} + T_3 = 0.0$$

$$T_{11} - 4.0T_{12} + T_{13} + T_{20} + T_4 = 0.0$$

$$T_{12} - 4.0T_{13} + T_{14} + T_{21} + T_5 = 0.0$$

$$T_{13} - 4.0T_{14} + T_{15} + T_{22} + T_6 = 0.0$$

Exterior Equations with Convection

$$T_b + 2.0T_9 + T_2 + \frac{2.0T_\infty \Delta x h}{k} = 2.0T_1 \left(\frac{\Delta x h}{k} + 2.0 \right)$$

$$T_b + 2.0T_9 + T_{18} + \frac{2.0T_\infty \Delta x h}{k} = 2.0T_{17} \left(\frac{\Delta x h}{k} + 2.0 \right)$$

$$2.0T_{10} + T_1 + T_3 + \frac{2.0T_\infty \Delta x h}{k} = 2.0T_2 \left(\frac{\Delta x h}{k} + 2.0 \right)$$

$$2.0T_{10} + T_{17} + T_{19} + \frac{2.0T_\infty \Delta x h}{k} = 2.0T_{18} \left(\frac{\Delta x h}{k} + 2.0 \right)$$

$$2.0T_{11} + T_2 + T_4 + \frac{2.0T_\infty \Delta x h}{k} = 2.0T_3 \left(\frac{\Delta x h}{k} + 2.0 \right)$$

$$2.0T_{11} + T_{18} + T_{20} + \frac{2.0T_\infty \Delta x h}{k} = 2.0T_{19} \left(\frac{\Delta x h}{k} + 2.0 \right)$$

$$2.0T_{12} + T_3 + T_5 + \frac{2.0T_\infty \Delta x h}{k} = 2.0T_4 \left(\frac{\Delta x h}{k} + 2.0 \right)$$

$$2.0T_{12} + T_{19} + T_{21} + \frac{2.0T_\infty \Delta x h}{k} = 2.0T_{20} \left(\frac{\Delta x h}{k} + 2.0 \right)$$

$$2.0T_{13} + T_4 + T_6 + \frac{2.0T_\infty \Delta x h}{k} = 2.0T_5 \left(\frac{\Delta x h}{k} + 2.0 \right)$$

$$2.0T_{13} + T_{20} + T_{22} + \frac{2.0T_\infty \Delta x h}{k} = 2.0T_{21} \left(\frac{\Delta x h}{k} + 2.0 \right)$$

$$2.0T_{14} + T_5 + T_7 + \frac{2.0T_\infty \Delta x h}{k} = 2.0T_6 \left(\frac{\Delta x h}{k} + 2.0 \right)$$

$$2.0T_{14} + T_{21} + T_{23} + \frac{2.0T_\infty \Delta x h}{k} = 2.0T_{22} \left(\frac{\Delta x h}{k} + 2.0 \right)$$

$$2.0T_{15} + T_6 + T_8 + \frac{2.0T_\infty \Delta x h}{k} = 2.0T_7 \left(\frac{\Delta x h}{k} + 2.0 \right)$$

$$2.0T_{15} + T_{22} + T_{24} + \frac{2.0T_\infty \Delta x h}{k} = 2.0T_{23} \left(\frac{\Delta x h}{k} + 2.0 \right)$$

$$2.0T_{16} + 2.0T_7 + \frac{2.0T_\infty \Delta x h}{k} = 2.0T_8 \left(\frac{\Delta x h}{k} + 2.0 \right)$$

$$2.0T_{16} + 2.0T_{23} + \frac{2.0T_\infty \Delta x h}{k} = 2.0T_{24} \left(\frac{\Delta x h}{k} + 2.0 \right)$$

Exterior Equations with Insulation

$$T_{14} - 4.0T_{15} + T_{16} + T_{23} + T_7 = 0.0$$

$$2.0T_{15} - 4.0T_{16} + T_{24} + T_8 = 0$$

Tabulated Temperatures and Plots

Table 1

Final Nodal Temperatures

Temperature (°F)							
113.598	91.279	84.111	81.546	80.589	80.229	80.098	80.065
131.955	100.622	87.976	83.060	81.174	80.457	80.196	80.130
113.598	91.279	84.111	81.546	80.589	80.229	80.098	80.065

113.598	91.279	84.111	81.546	80.589	80.229	80.098	80.065
131.955	100.622	87.976	83.060	81.174	80.457	80.196	80.130
113.598	91.279	84.111	81.546	80.589	80.229	80.098	80.065

Figure 1

Colored Distribution of Nodal Temperatures

Explanation of Software Code

All software code used can be seen in Appendix A. I chose to use Python to prepare a script to calculate the nodal temperatures. The script relies of 3 external dependencies:

- sympy
- xlsxwriter
- pandas

Two of these dependencies are commonly used well known packages. “Sympy” is a python module that is used to define and solve systems of equations, “xlsxwriter” is used to write out python data to an excel file, and “Pandas” is used to print the data of the excel file to the console window. These dependencies can be installed with the following commands on a Windows python distribution:

```
pip install sympy
pip install pandas
pip install xlsxwriter
```

For this reason, I have also included a message at the beginning of the script to warn about the excel file that is generated when the script is run. We start by defining the properties of the system:

- k, the thermal conductivity
- h, the convection heat transfer coefficient
- T_∞, the temperature of the surroundings
- T_b, the base temperature
- Δx, the horizontal distance between each node
- Δy, the vertical distance between each node

I then created a generalized “TNode” class that holds all the information and attributes of a single temperature node, including its neighbors and its own nodal temperature.

We then create a grid of these “TNode” objects and iterate over the grid after it has been created to find and assign all the neighbors. After doing this, we define variables for each temperature node in the grid using Sympy.

Using this along with our new grid of “TNodes”, we can apply our finite-difference equations by iterating over the grid and setting up a system of equations using variables from Sympy, and appending them to a list that we can pass as a linear system of equations to solve later.

This system of equations is then solved by Sympy and the solution temperatures are tabulated and output to an excel file for any further calculations that the user may wish to do with these nodal temperatures.

Lastly, the nodal temperatures are also printed to the console window using “Pandas” so that they can be quickly viewed.

Appendix

Heat Conduction Analysis Code

```
import sympy
import xlsxwriter
import pandas as pd

print("Running this script will output an excel file into the directory it
                                        has been downloaded into.")
input("Please press Enter if you wish to proceed.")

# Properties
k = 8.0 # Btu / hr*ft*degF
h = 96.0 # Btu / hr*ft*ft*degF
T_inf = 80.0 # degF
T_b = 200.0 # degF
dx = 1.0 / 8.0 # inch
dy = dx # inch

# Create a class to generalize each node
class TNode:
  # Neighbors
  bn = None # Bottom neighbor
  tn = None # Top neighbor
  ln = None # Left neighbor
  rn = None # Right neighbor

  tl = None # Top-left neighbor
  tr = None # Top-right neighbor
  bl = None # Bottom-left neighbor
  br = None # Bottom-right neighbor

  T = None # Node Temperature (Symbolic variable)
```

```
  def __init__(self, idx) -> None:
    self.index = idx

# Initialize grid of nodes
columns = 8
rows = 3
grid = []
idx = 1
for j in range(rows):
  r = []
  for i in range(idx, idx + columns):
    node = TNode(i)
    r.append(node)
    idx += 1
  grid.append(r)

# Now that we have initialized a grid, we find all the neighbors
for i in range(columns):
  for j in range(rows):
    node  = grid[j][i]
    # Bottom & Top neighbors
    if node.index + columns <= rows * columns:
      # Find bottom and top neighbor pairs
      # Since these pairs are each others vertical neighbors,
      # we only need to scan once to locate both the top and bottom
                                    neighbors
      bottom_neighbor = grid[j+1][i]
      node.bn = bottom_neighbor
      bottom_neighbor.tn = node

    # Left & Right Neighbors
    if i + 1 <= 7:
      # Find right and left neighbor pairs
```

```
      # Since these pairs are each others horizontal neighbors,
      # we only need to scan once to locate both the right and left
                                        neighbors
      right_neighbor = grid[j][i+1]
      node.rn = right_neighbor
      right_neighbor.ln = node

    if j-1 >= 0 and i-1 >= 0:
      top_left = grid[j-1][i-1]
      node.tl = top_left
      top_left.br = node

    if j+1 <=2 and i-1 >= 0:
      bottom_left = grid[j+1][i-1]
      node.bl = bottom_left
      bottom_left.tr = node

# Create list of symbolic variables
# Axis of symmetry runs down the middle row
# Therefore, per column there are only 2 variables:
#   "Inner" temperature and "Outer" temperature
# Then we only need to label these 2 variables per column

variable_set = []
for j in range(rows):
  for i in range(columns):
    node = grid[j][i]
    if j == 1:
      node.T = sympy.symbols("T_i_" + str(node.index))
    else:
      node.T = sympy.symbols("T_o_" + str(node.index))
    variable_set.append(node.T)
```

```
# Create the system of nodal finite-difference equations

equation_set = []
for i in range(columns):
  for j in range(rows):
    node  = grid[j][i]

    neighbors = [
      node.tn,
      node.bn,
      node.ln,
      node.rn,
      node.tr,
      node.tl,
      node.br,
      node.bl
    ]
    num_neighbors = 0
    for value in neighbors:
      num_neighbors += int(value != None)

    if num_neighbors == 8:
      # This is an interior node
      eq = sympy.Eq(node.tn.T + node.bn.T + node.rn.T + node.ln.T - 4.0*
                                        node.T, 0.0)
      equation_set.append(eq)

    if num_neighbors == 5 and not node.index in [9, 16]:
      # This is a node at plane surface with convection
      # Find out if the convection surface is above or below
      if node.tn == None:
        # Convection surface is above
        SigmaTn = 2.0*node.bn.T + node.ln.T + node.rn.T
```

```
      # We precalculate a constant A to make our equations simpler
      A = 2.0*h*dx / k
      eq = sympy.Eq(SigmaTn + A*T_inf - A*node.T - 4.0*node.T, 0)
      equation_set.append(eq)
    else:
      # Convection surface is below
      SigmaTn = 2.0*node.tn.T + node.ln.T + node.rn.T
      # We precalculate a constant A to make our equations simpler
      A = 2.0*h*dx / k
      eq = sympy.Eq(SigmaTn + A*T_inf - A*node.T - 4.0*node.T, 0)
      equation_set.append(eq)
  else:
    # Handle edge cases
    if node.index == 16:
      eq = sympy.Eq(2.0*node.ln.T + node.tn.T + node.bn.T - 4.0*node.T,
                                    0)
      equation_set.append(eq)
    elif node.index == 9:
      eq = sympy.Eq(node.tn.T + node.bn.T + node.rn.T + T_b - 4.0*node.T
                                    , 0)
      equation_set.append(eq)

  if num_neighbors == 3:
    # Adiabatic surface can be treated as axis of symmetry
    # This becomes similar to the nodes with 5 neighbors and convection
    # Determine if the convection surface is above or below

    # We precalculate a constant A to make our equations simpler
    A = 2.0*h*dx / k
    if node.tn == None:
      # Convection surface is above
      if node.index == 1:
        SigmaTn = 2.0*node.bn.T + T_b + node.rn.T
```

```
            eq = sympy.Eq(SigmaTn + A*T_inf - A*node.T - 4.0*node.T, 0)
            equation_set.append(eq)
          elif node.index == 8:
            SigmaTn = 2.0*node.bn.T + 2.0*node.ln.T
            eq = sympy.Eq(SigmaTn + A*T_inf - A*node.T - 4.0*node.T, 0)
            equation_set.append(eq)
        else:
          # Convection surface is below
          if node.index == 17:
            SigmaTn = 2.0*node.tn.T + T_b + node.rn.T
            eq = sympy.Eq(SigmaTn + A*T_inf - A*node.T - 4.0*node.T, 0)
            equation_set.append(eq)
          elif node.index == 24:
            SigmaTn = 2.0*node.tn.T + 2.0*node.ln.T
            eq = sympy.Eq(SigmaTn + A*T_inf - A*node.T - 4.0*node.T, 0)
            equation_set.append(eq)

# Solve the system and sort the node temperatures
# Then write them to an excel file

solution = sympy.solve(equation_set, variable_set)

workbook = xlsxwriter.Workbook("NodeTemperatures.xlsx")
worksheet = workbook.add_worksheet()

for i in range(columns):
  for j in range(rows):
    node  = grid[j][i]
    worksheet.write(j,i,solution[node.T])

workbook.close()

print("\nFinal Nodal Temperatures:")
```

```
df = pd.read_excel("NodeTemperatures.xlsx")
print(df)


input("\n\nPress Enter to close program")
```