MEEN 644 - Homework 2

Logan Harbour
February 17, 2019

Problem statement

Consider one-dimensional heat conduction in a cylindrical copper rod of length 1.0 m long. The diameter of the rod is 0.05 m. The left end of the rod is held at 100 °C and the ambient temperature is 25 °C. Heat is transported from the surface of the rod and the right end of the rod through natural convection to the ambient. The natural convection heat transfer coefficient is 0.5 W/m^2 °C. Write a finite volume code to predict temperature distribution as a function of length. Use TDMA to solve a set of discretization equations. Make calculations using ITMAX: 6, 11, 21, 41, and 81 nodes. Plot your results.

Preliminaries

One-dimensional heat conduction

With one-dimensional heat conduction with convection and constant material properties, we have the ODE

$$\begin{cases} \frac{\mathrm{d}^2T}{\mathrm{d}x^2} + \frac{h}{kd}(T - T_\infty) = 0\,, \\ T(0) = T_0\,, \\ \frac{\mathrm{d}T}{\mathrm{d}x}\Big|_{x=L} = -\frac{h}{k}(T - T_\infty)\,, \end{cases} \tag{1}$$

where

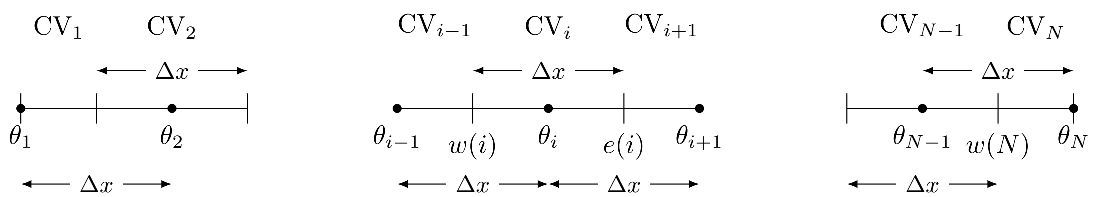
$$k \equiv 400 \text{ W/m °C}\,, \qquad h \equiv 0.5 \text{ W/m}^2 \text{ °C}\,, \qquad d \equiv 0.05 \text{ m}\,,$$
$$L \equiv 1.0 \text{ m}\,, \qquad T_0 \equiv 100 \text{ °C}\,, \qquad T_\infty \equiv 25 \text{ °C}\,.$$

We then make the substitutions $\theta(x) = T(x) - T_\infty$ and $m = 4h/kd$ to obtain the simplification

$$\begin{cases} \frac{\mathrm{d}^2\theta}{\mathrm{d}x^2} + m\theta = 0\,, \\ \theta(0) = T_0 - T_\infty\,, \\ \frac{\mathrm{d}\theta}{\mathrm{d}x}\Big|_{x=L} = -\frac{h}{k}\theta\,. \end{cases} \tag{2}$$

Grid generation

We discretize the region on $x = [0, L]$ by N (also defined as ITMAX) nodes and N control volumes, as follows in Figure 1 with $\Delta x = L/(N-1)$.



(a) Left CV (b) Interior CV ($1 < i < N$) (c) Right CV

Figure 1: The control volumes defined for discretization of the problem.

Equation discretization

Internal control volume equation

We start with the integration over an interior control volume, as

$$\int_{\mathrm{CV}_i} \left[-\frac{\mathrm{d}^2\theta}{\mathrm{d}x^2} + m\theta \right] dx = 0\,, \quad 1 < i < N\,,$$

in which we know that the material properties are independent and we assume θ_i to be constant over the cell for the second term to obtain

$$-\left(\left.\frac{\mathrm{d}\theta}{\mathrm{d}x}\right|_{e(i)} - \left.\frac{\mathrm{d}\theta}{\mathrm{d}x}\right|_{w(i)} \right) + m\Delta x\theta_i = 0\,, \quad 1 < i < N\,.$$

Use the two node formulation for the derivative terms and simplify as

$$-\left(\frac{\theta_{i+1} - \theta_i}{\Delta x} - \frac{\theta_i - \theta_{i-1}}{\Delta x} \right) + m\Delta x\theta_i = 0\,, \quad 1 < i < N\,,$$
$$-\frac{1}{\Delta x}\theta_{i-1} + \left(m\Delta x + \frac{2}{\Delta x} \right)\theta_i - \frac{1}{\Delta x}\theta_{i+1} = 0\,, \quad 1 < i < N\,.$$

Take note that at the $i = 2$ equation, θ_1 is known therefore we have

$$\boxed{\left(m\Delta x + \frac{2}{\Delta x} \right)\theta_2 - \frac{1}{\Delta x}\theta_3 = \frac{T_0 - T_\infty}{\Delta x}\,,} \tag{3}$$
$$\boxed{-\frac{1}{\Delta x}\theta_{i-1} + \left(m\Delta x + \frac{2}{\Delta x} \right)\theta_i - \frac{1}{\Delta x}\theta_{i+1} = 0\,, \quad 2 < i < N\,.} \tag{4}$$

Right control volume equation

We start with the integration over the right control volume, CV_N, as

$$\int_{\mathrm{CV}_N} \left[-\frac{\mathrm{d}^2\theta}{\mathrm{d}x^2} + m\theta \right] dx = 0\,,$$

in which for the second term we will assume θ_N to be constant over CV_N to obtain

$$-\left(\left.\frac{\mathrm{d}\theta}{\mathrm{d}x}\right|_{x=L} - \left.\frac{\mathrm{d}\theta}{\mathrm{d}x}\right|_{w(N)} \right) + \frac{1}{2}m\Delta x\theta_N = 0\,.$$

Use the two node formulation for the derivative term at $w(N)$ and the right boundary condition for the derivative term at $x = L$ m to obtain

$$\frac{h}{k}\theta_N + \frac{\theta_N - \theta_{N-1}}{\Delta x} + \frac{1}{2}m\Delta x\theta_N = 0\,,$$
$$\boxed{-\frac{1}{\Delta x}\theta_{N-1} + \left(\frac{1}{2}m\Delta x + \frac{h}{k} + \frac{1}{\Delta x}\right)\theta_N = 0\,.} \tag{5}$$

Summary

Putting the above into the a_p, a_e, a_w formulation we have discussed in class, we have

$$\begin{cases} a_p = m\Delta x + \frac{2}{\Delta x}\,, \quad a_e = -\frac{1}{\Delta x}\,, & \text{left boundary} \\ a_w = -\frac{1}{\Delta x}\,, \quad a_p = m\Delta x + \frac{2}{\Delta x}\,, \quad a_e = -\frac{1}{\Delta x} & \text{interior} \\ a_w = -\frac{1}{\Delta x}\,, \quad a_p = \frac{1}{2}m\Delta x + \frac{h}{k} + \frac{1}{\Delta x}\,, & \text{right boundary} \end{cases}$$

TDMA

The system we are solving is of the form

$$\begin{bmatrix} b_1 & c_1 & & & 0 \\ a_2 & b_2 & c_2 & & \\ & a_3 & b_3 & \ddots & \\ & & \ddots & \ddots & c_{n-1} \\ 0 & & & a_n & b_n \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} d_1 \\ d_2 \\ d_3 \\ \vdots \\ d_n \end{bmatrix}.$$

We will solve this system using the tridiagonal matrix algorithm (TDMA) given the fact that our system is positive definite. With our matrix in the form above, the algorithm follows in Algorithm 1. Note that for the algorithm below, we are solving the solution vector $\vec{x}$ in-place with the vector $\vec{d}$. This algorithm is in the member function `solveTDMA()` in the `TriDiagonal` class, which stores the left hand side system in the vectors $\vec{a}, \vec{b}$, and $\vec{c}$ (denoted `A`, `B`, and `C` in the code, respectively).

for $i = 2, 3, \ldots, n$ **do**
  $w = a_i/b_{i-1}$;
  $b_i = b_i - wc_{i-1}$;
  $d_i = d_i - wd_{i-1}$;
end
$d_n = d_n/b_n$;
for $i = n-1, n-2, \ldots, 1$ **do**
  $d_i = d_i - c_i d_{i+1}$;
  $d_i = d_i/b_i$;
end

Algorithm 1: The tridiagonal matrix algorithm (TDMA).

Results

The plotted results as requested follow below in Figure 2. In addition, the solution for ITMAX = 6 is tabulated in Table 1.

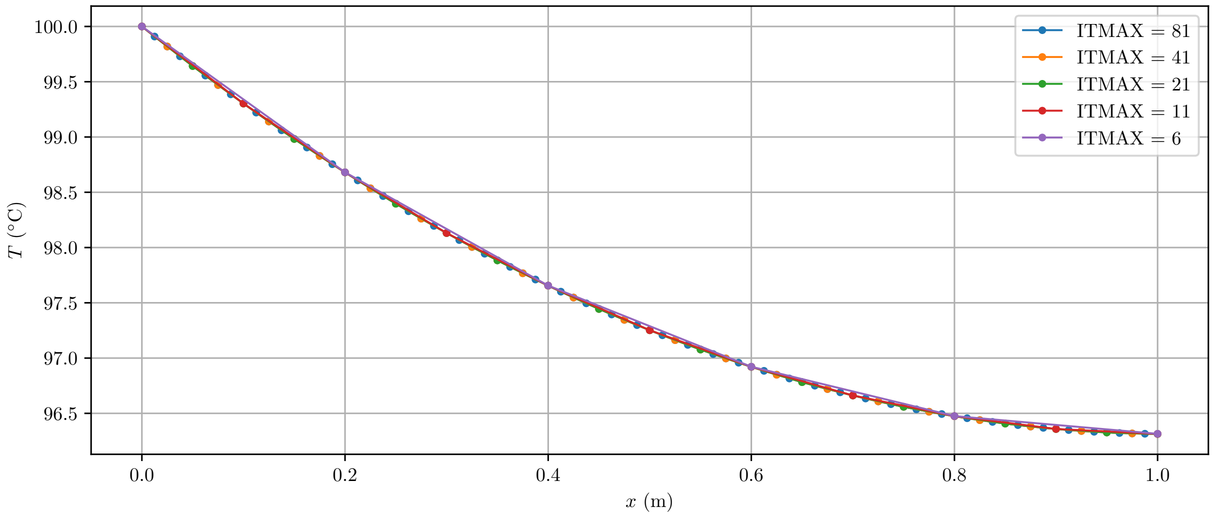


Figure 2: The plotted solution.

Table 1: The solution for ITMAX = 6.

x (m)	θ (°C)	T (°C)
0.0	75.00000	100.00000
0.2	73.68060	98.68060
0.4	72.65593	97.65593
0.6	71.92188	96.92188
0.8	71.47552	96.47552
1.0	71.31506	96.31506

Code listing

For the implementation, we have the following files:

- `Makefile` – Allows for compiling the c++ project with `make`.
- `hwk2.cpp` – Contains the `main()` function that is required by C and contains the `solveRod()` function which solves the 1-D heat conduction problem.
- `TriDiagonal.h` – Contains the `TriDiagonal` class which provides storage for a tri-diagonal matrix including the TDMA solver found in the member function `solveTDMA()`.
- `plots.py` - Produces the plot in this report.

Makefile

```
all: hwk2

hwk2: TriDiagonal.h hwk2.cpp
	clang++ -std=c++14 hwk2.cpp -o $@
```

hwk2.cpp

```
#include <fstream>
#include <iomanip>
#include <string>

#include "TriDiagonal.h"

/**
 * Saves a vector to a csv.
 */
void saveVectorCsv(const std::vector<double> x, const std::string filename) {
  std::ofstream f;
  f.open(filename);
  for (unsigned int i = 0; i < x.size(); ++i)
    f << std::setprecision(12) << x[i] << std::endl;
  f.close();
}

/**
 * Solves the 1D heat-conduction (with convection) problem with N nodes.
 */
void solveRod(unsigned int N) {
  // Initialize geometry, material properties, and constant coefficients
  double L = 1,          // [m]
      dx = L / (N - 1), // [m]
      d = 0.05,         // [m]
      k = 400.0,        // [W/m-C]
      h = 0.5,          // [W/m^2-C]
      T0 = 100.0,       // [C]
      Tinf = 25.0,      // [C]
      m = 4.0 * h / (k * d),
      a_p = m * dx + 2.0 / dx,
      a_w = 1.0 / dx,
      a_e = 1.0 / dx;

  // Initialize system A theta = b
  TriDiagonal A(N - 1);
  std::vector<double> b(N - 1);

  // Fill system
  A.addTopRow(a_p, -a_e);
  b[0] = a_w * (T0 - Tinf);
  for (unsigned int i = 1; i < N - 2; ++i)
    A.addMiddleRow(i, -a_w, a_p, -a_e);
  A.addBottomRow(-a_w, m * dx / 2.0 + h / k + a_e);

  // Solve system in place and save
  A.solveTDMA(b);
  saveVectorCsv(b, "results/theta_" + std::to_string(N) + ".csv");
}

int main() {
  // Run each requested case
  for (unsigned int N : {6, 11, 21, 41, 81})
    solveRod(N);

  return 0;
```

```
}
```

TriDiagonal.h

```cpp
#define NDEBUG
#include <cassert>
#include <vector>

/**
 * Class that holds a tri-diagonal matrix and is able to perform TDMA in place
 * with a given RHS.
 */
class TriDiagonal {
public:
  TriDiagonal(unsigned int N) : N(N), A(N), B(N), C(N - 1) {}

  // Adders for the top, interior, and bottom rows
  void addTopRow(double b, double c) {
    B[0] += b;
    C[0] += c;
  }
  void addMiddleRow(unsigned int i, double a, double b, double c) {
    assert(i < N - 1 && i != 0);
    A[i] += a;
    B[i] += b;
    C[i] += c;
  }
  void addBottomRow(double a, double b) {
    A[N - 1] += a;
    B[N - 1] += b;
  }

  // Solves the system Ax = d in place where d eventually stores the solution
  void solveTDMA(std::vector<double> &d) {
    // Forward sweep
    double tmp = 0;
    for (unsigned int i = 1; i < N; ++i) {
      tmp = A[i] / B[i - 1];
      B[i] -= tmp * C[i - 1];
      d[i] -= tmp * d[i - 1];
    }

    // Backward sweep
    d[N - 1] /= B[N - 1];
    for (unsigned int i = N - 2; i != std::numeric_limits<unsigned int>::max();
         --i) {
      d[i] -= C[i] * d[i + 1];
      d[i] /= B[i];
    }
  }

protected:
  // Matrix size (N x N)
  unsigned int N;
  // Left/main/right diagonal storage
  std::vector<double> A, B, C;
};
```

plot.py

```python
import numpy as np
import matplotlib.pyplot as plt
```

```python
plt.rc('text', usetex=True)
plt.rc('font', family='serif')

# Load results
T = {}
Tinf = 25
T0 = 100
nodes = [81, 41, 21, 11, 6]
for N in nodes:
    # Remember to add Tinf to all theta and append T0 to the first
    T[N] = np.loadtxt('results/theta_{}.csv'.format(N)) + Tinf
    T[N] = np.insert(T[N], 0, T0)

# Plot results
fig = plt.figure()
fig.set_figwidth(9)
fig.set_figheight(4)
for N in nodes:
    x = np.linspace(0, 1, num=N)
    plt.plot(x, T[N], '.-', linewidth=1, label='ITMAX = {}'.format(N))
plt.xlabel(r'$x$ (m)')
plt.ylabel(r'$T$ ($^\circ$C)')
plt.tight_layout()
plt.grid()
plt.legend()
fig.savefig('results/result.pdf')
```