**Transient 1D Conductive Heat Transfer**

Wayne Pafko  
11/19/01

**Background**

This Mathcad document shows how to use an finite difference algorithm to solve an initial value transient heat transfer problem involving conduction in a slab. In this problem, the temperature of the slab is initially uniform (Initial Condition). The edges are then instantly changed to a constant temperature boundary condition (Dirichlet BC). The finite difference algorithm then calculates how the temperature profile in the slab changes over time.

**Physical Model**

Conduction of heat in a slab is usually described using a parabolic partial differential equation.

\[
\frac{dT}{dt} = \lambda \frac{d^2 T}{dx^2} \quad \text{Transient conduction of heat in a slab.}
\]

This partial differential equation can be approximated using finite differences...

\[
\frac{\delta T}{\Delta t} = \lambda \frac{\Delta (\Delta T)}{(\Delta x)^2} \quad \text{Where...} \quad \Delta (\Delta T) = (T_{i+1} - T_i) - (T_i - T_{i-1}) = T_{i+1} - 2T_i + T_{i-1}
\]

\[
\delta T = T_{j+1} - T_j \quad \text{...with i indexing across the node and j indexing over time.}
\]

If the time step and nodal spacing is sufficiently small the finite difference solution will approach the analytical solution and we will find the numerical solution to our partial differential equation.

**Problem Setup**

1) Input the physical properties the slab (density, heat capacity, & thermal conductivity)...

\[
\rho := 1 \cdot \frac{\text{gm}}{\text{m}^3} \quad \text{Cp} := 5 \cdot \frac{\text{J}}{\text{gm} \cdot \text{K}} \quad \text{k} := 2 \cdot 10^{-5} \cdot \frac{\text{W}}{\text{m} \cdot \text{K}}
\]

\[
\lambda := \frac{k}{\rho \cdot \text{Cp}} \quad \lambda = 4 \times 10^{-6} \frac{\text{m}^2}{\text{s}} \quad \text{Thermal diffusivity.}
\]

2) Input finite difference parameters (time step, node spacing, # nodes, # time steps)...

\[
\Delta \text{time} := 0.1 \cdot \text{sec} \quad \Delta x := 0.1 \cdot \text{cm} \quad \lambda \cdot \frac{\Delta \text{time}}{\Delta x^2} = 0.4
\]

Note: This term should be less than 0.5 to ensure stability, less than 0.25 to ensure convergence, and about 0.166 to minimize truncation error (similar to a Courant-Friedrichs-Lewy stability criterion).
timesteps \cdot \Delta t = 60 \text{ s} \quad \text{Total simulation time}

\text{nodes} \cdot \Delta x = 3 \text{ cm} \quad \text{Total simulation width}
i := 0 \ldots (\text{nodes} - 1)

3) Input initial conditions for the slab.

\[ T_{\text{initial}} = 300 \cdot \text{K} \]

\[ T_{\text{initial}} = \begin{bmatrix} 0 & 300 & 300 & 300 & 300 & 300 & 300 & 300 & 300 \end{bmatrix} \text{K} \]

4) Input left & right boundary conditions.

\[ \text{BC}_{\text{left}} := 350 \cdot \text{K} \quad \text{BC}_{\text{right}} := 440 \cdot \text{K} \]

**Problem Solution**

5) Solve the problem using a finite difference algorithm.

When you pass a vector containing the temperature at each node to the "updateTemps" function calculates what the temperature at each node will be at the next time step and returns that vector updated temperatures. Note that the temperature at the edges are always just set to equal the left and right boundary conditions (Dirichlet BC).

updateTemps(T) :=

\[ \begin{align*}
T_{\text{0}} & \leftarrow \text{BC}_{\text{left}} \\
T_{\text{nodes-1}} & \leftarrow \text{BC}_{\text{right}} \\
\text{for} \quad i & \in 1 \ldots \text{nodes} - 2 \\
T_i & \leftarrow T_i + \lambda \cdot \frac{\Delta t}{\Delta x^2} \left( T_{i-1} - 2 \cdot T_i + T_{i+1} \right) \\
\text{Temp} & \end{align*} \]

The "SolveFiniteDifference" function first sets the temperature at each node to be equal to the initial temperature. It then loops over the specified number of time steps and calculates what the temperature profile will be at each step using the "updateTemps" function. This is stored in a 2D array. Each row contains the temperature profile for that time step.

SolveFiniteDifference(steps) :=

\[ \begin{align*}
T^{(0)} & \leftarrow T_{\text{initial}} \\
\text{for} \quad i & \in 1 \ldots \text{floor}(\text{steps}) \\
T^{(i)} & \leftarrow \text{updateTemps}(T^{(i-1)}) \\
T & \end{align*} \]
Tanswer := \textbf{SolveFiniteDifference}(\textit{timesteps})

The "Tanswer" array contains solution. Each row contains a temperature profile for a given time step. There are as many rows as time steps and as many columns as nodes (array is only partially shown, scroll around to see entire solution).

\[
\begin{array}{cccccccc}
0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\
0 & 300.0 & 300.0 & 300.0 & 300.0 & 300.0 & 300.0 & 300.0 \\
1 & 350.0 & 300.0 & 300.0 & 300.0 & 300.0 & 300.0 & 300.0 \\
2 & 350.0 & 320.0 & 300.0 & 300.0 & 300.0 & 300.0 & 300.0 \\
3 & 350.0 & 324.0 & 308.0 & 300.0 & 300.0 & 300.0 & 300.0 \\
4 & 350.0 & 328.0 & 311.2 & 303.2 & 300.0 & 300.0 & 300.0 \\
5 & 350.0 & 330.1 & 314.7 & 305.1 & 301.3 & 300.0 & 300.0 \\
6 & 350.0 & 331.9 & 317.0 & 307.4 & 302.3 & 300.5 & 300.0 \\
7 & 350.0 & 333.2 & 319.1 & 309.2 & 303.6 & 301.0 & 300.2 \\
8 & 350.0 & 334.3 & 320.8 & 311.0 & 304.8 & 301.7 & 300.5 \\
\end{array}
\]

The last row contains the temperature profile at the end of the simulation. This may or may not be the steady-state solution depending on how much simulation time elapses.

\[
\begin{array}{cccccccc}
0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\
0 & 350.0 & 352.3 & 354.6 & 357.0 & 359.4 & 361.8 & 364.2 \\
\end{array}
\]

6) View the results

\[x_i := i \cdot \Delta x \quad \text{Position of each node in slab}\]
We can also animate the results...

\[
\text{time} := 10 \cdot \text{FRAME} \cdot \Delta \text{time} \quad \text{Calculate time at each FRAME}
\]

\[
\frac{\text{timesteps}}{10} = 60 \quad \text{Animate FRAME up to this number.}
\]