Alternative Boundary Condition Implementations for Crank Nicolson Solution to the Heat Equation

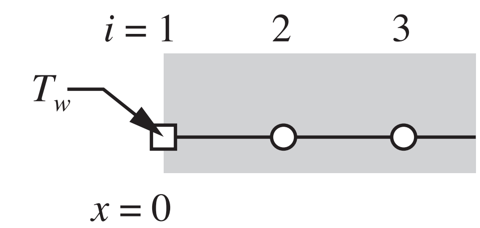
ME 448/548 Notes

Gerald Recktenwald
Portland State University
Department of Mechanical Engineering
gerry@pdx.edu

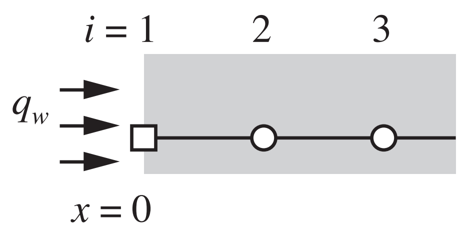
Overview

1. Goal is to allow Dirichlet, Neumann and mixed boundary conditions
2. Use ghost node formulation
   - Preserve spatial accuracy of $\mathcal{O}(\Delta x^2)$
   - Preserve tridiagonal structure to the coefficient matrix
3. Implement in a code that uses the Crank-Nicolson scheme.
4. Demonstrate the technique on sample problems

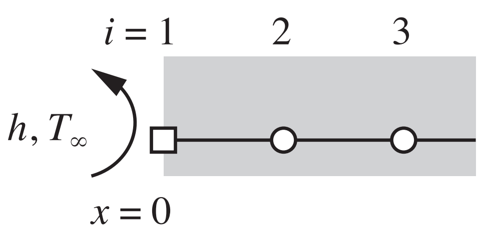
Heat Transfer Boundary Conditions



1. Prescribe T_w, a know wall temperature. Maybe $T_w = f(t)$.
2. Solve internal $T(x,t)$ field
3. Compute the wall heat flux, q_w.



1. Prescribe q_w, a know wall heat flux. Maybe $q_w(t) = f(t)$.
2. Solve internal $T(x,t)$ field
3. Compute the wall temperature, T_w.



1. Prescribe T_∞ and h. Maybe $T_w(t) = f(t)$ and $h = f(t)$.
2. Solve internal $T(x,t)$ field
3. Compute the wall heat flux, q_w and wall temperature, T_w.

Convective Boundary Condition

The general form of a convective boundary condition is

$$\left.\frac{\partial u}{\partial x}\right|_{x=0} = g_0 + h_0 u \tag{1}$$

This is also known as a *Robin* boundary condition or a *boundary condition of the third kind*.

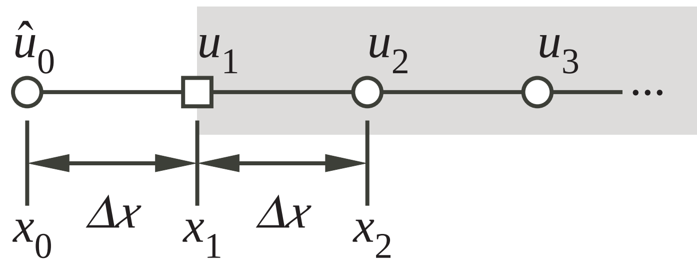
The simplistic implementation is to replace the derivative in Equation (1) with a one-sided difference

$$\frac{u_2^{k+1} - u_1^{k+1}}{\Delta x} = g_0 + h_0 u_1^{k+1} \tag{2}$$

Don't do that! The one-sided difference approximation has a spatial accuracy of $\mathcal{O}(\Delta x)$.

Introduce a Ghost Node

Imagine that there is a node $\hat{u}_0$ that is *outside* of the domain



this node is used to enforce the boundary condition from Equation (1).

The value $\hat{u}_0$ does not explicitly appear in the numerical scheme. We introduce it as a device to introduce a higher order approximation to the gradient at the boundary. It turns out that with algebra, $\hat{u}_0$ disappears from the final formulation.

Use the BC to compute $\hat{u}_0$ by extrapolation

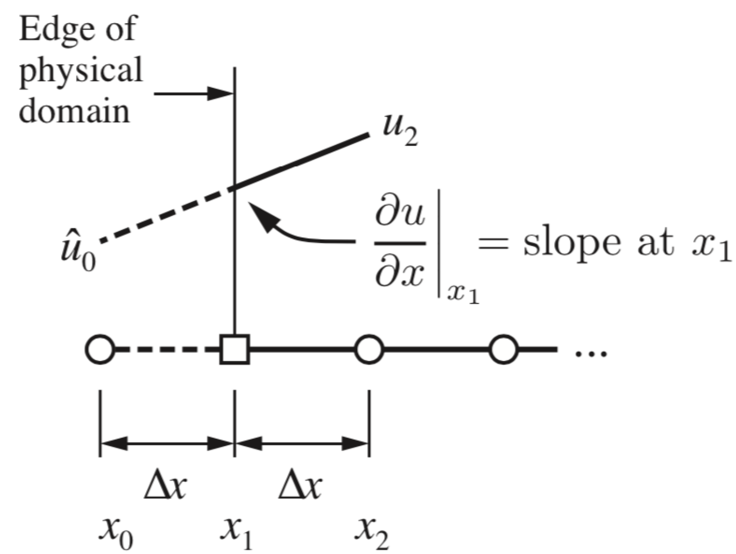
Use a *central* difference approximation at $x = 0$ $(x = x_1)$ to impose the boundary condition.

$$\frac{u_2 - \hat{u}_0}{2\Delta x} = g_0 + h_0 u_1. \tag{3}$$

The value of $\hat{u}_0$ consistent with the boundary condition is

$$\hat{u}_0 = u_2 - 2\Delta x(g_0 + h_0 u_1). \tag{4}$$

Equation (4) allows us to eliminate $\hat{u}_0$ at the boundary.



Equation for u_1

Evaluate the finite difference form of the heat equation at $x = x_1$.

$$\frac{u_1^{k+1} - u_1^k}{\Delta t} = \theta\alpha\left[\frac{\hat{u}_0^{k+1} - 2u_1^{k+1} + u_2^{k+1}}{\Delta x^2}\right] + (1-\theta)\alpha\left[\frac{\hat{u}_0^{k} - 2u_1^{k} + u_2^{k}}{\Delta x^2}\right]$$

Choose $\theta = 1/2$ and use the formulas for $\hat{u}_0$ at time step k and time step $k+1$

$$\begin{aligned}\frac{u_1^{k+1} - u_1^k}{\Delta t} = \frac{\theta\alpha}{\Delta x^2}&\left[\boxed{u_2^{k+1} - 2\Delta x(g_0^{k+1} + h_0^{k+1}u_1^{k+1})} - 2u_1^{k+1} + u_2^{k+1}\right] \\ &+ \frac{(1-\theta)\alpha}{\Delta x^2}\left[\boxed{u_2^{k} - 2\Delta x(g_0^{k} + h_0^{k}u_1^{k})} - 2u_1^{k} + u_2^{k}\right]\end{aligned}$$

The terms in boxes are from the boundary condition

Rearrange the Equation for u_1

Algebraically rearranging the preceding equation gives

$$a_1 u_1^{k+1} + b_1 u_2^{k+1} = d_1 \tag{5}$$

where

$$a_1 = \frac{1}{\Delta t} + \frac{2\theta\alpha}{\Delta x^2}(1 + \Delta x h_0^{k+1}) \tag{6}$$

$$b_1 = -\frac{2\theta\alpha}{\Delta x^2} \tag{7}$$

$$d_1 = \left[\frac{1}{\Delta t} - \frac{2(1-\theta)\alpha}{\Delta x^2}(1 + \Delta x h_0^k)\right] u_1^k + \frac{2(1-\theta)\alpha}{\Delta x^2} u_2^k - \frac{2\alpha}{\Delta x}\left[\theta g_0^{k+1} + (1-\theta) g_0^k\right] \tag{8}$$

These equations define the terms for the first row in the system of equations

Data structure for implementing alternative BC in the MATLAB code

Store the data defining the boundary condition for both boundaries in a 2×3 matrix.

The first row has data for $x = 0$
The second row has data for $x = L$.

$$
\texttt{ubc} \;=\; \begin{array}{c} \left[\begin{array}{ccc} \boxed{\text{Type}} & \boxed{\text{Value 1}} & \boxed{\text{Value 2}} \\ \boxed{\text{Type}} & \boxed{\text{Value 1}} & \boxed{\text{Value 2}} \end{array}\right] \end{array} \begin{array}{l} x = 0 \\ x = L \end{array}
$$

$\texttt{Type}$ is a flag with the boundary condition type.

if $\texttt{ubc}(\texttt{b}, 1) = 1$, then

$$u(x_b) = \text{value}$$
$$\texttt{ubc}(\texttt{b}, 2) = \text{value of } u \text{ at boundary}$$
$$\texttt{ubc}(\texttt{b}, 3) = \text{not used}$$

if $\texttt{ubc}(\texttt{b}, 1) = 2$, then

$$\partial u / \partial x \big|_{x_b} = g + h u(x_b)$$
$$\texttt{ubc}(\texttt{b}, 2) = g$$
$$\texttt{ubc}(\texttt{b}, 3) = h$$

$\texttt{b} = 1$ for $x = 0$

$\texttt{b} = 2$ for $x = L$

Verification: Solve the toy problem on half of the domain

The toy problem used to test the codes

$$\frac{\partial u}{\partial t} = \alpha \frac{\partial^2 u}{\partial x^2} \qquad t > 0, \quad 0 \le x \le L$$
$$u(0,t) = u(L,t) = 0;$$
$$u(x,0) = \sin(\pi x/L)$$

only needs to be solved on one half of the domain

$$\frac{\partial u}{\partial t} = \alpha \frac{\partial^2 u}{\partial x^2} \quad t > 0,\ 0 \le x \le L/2$$
$$u(0,t) = 0; \quad \left.\frac{\partial u}{\partial x}\right|_{L/2} = 0$$
$$u(x,0) = \sin(\pi x/L)$$

$$\frac{\partial u}{\partial t} = \alpha \frac{\partial^2 u}{\partial x^2} \quad t > 0,\ L/2 \le x \le L$$
$$\left.\frac{\partial u}{\partial x}\right|_{L/2} = 0 \quad u(L,t) = 0;$$
$$u(x,0) = \sin(\pi x/L)$$

Verification: Solve the toy problem on half of the domain

Use `ubc` matrix to specify boundary conditions.

For the half-problem on $0 \le x \le L/2$:

$$
u(0,t) = 0 \quad \Longrightarrow \quad
\begin{aligned}
&\texttt{ubc(1,1)} = 1, && \text{boundary type} \\
&\texttt{ubc(1,2)} = 0, && \text{value of } u \text{ at } x = 0 \\
&\texttt{ubc(1,3)} = 0, && \text{not used}
\end{aligned}
$$

$$
\left.\frac{\partial u}{\partial x}\right|_{L/2} = 0 \quad \Longrightarrow \quad
\begin{aligned}
&\texttt{ubc(2,1)} = 2, && \text{boundary type} \\
&\texttt{ubc(2,2)} = 0, && \text{value of } g_{L/2} \\
&\texttt{ubc(2,3)} = 0, && \text{value of } h_{L/2}
\end{aligned}
$$

Verification: Solve the toy problem on half of the domain

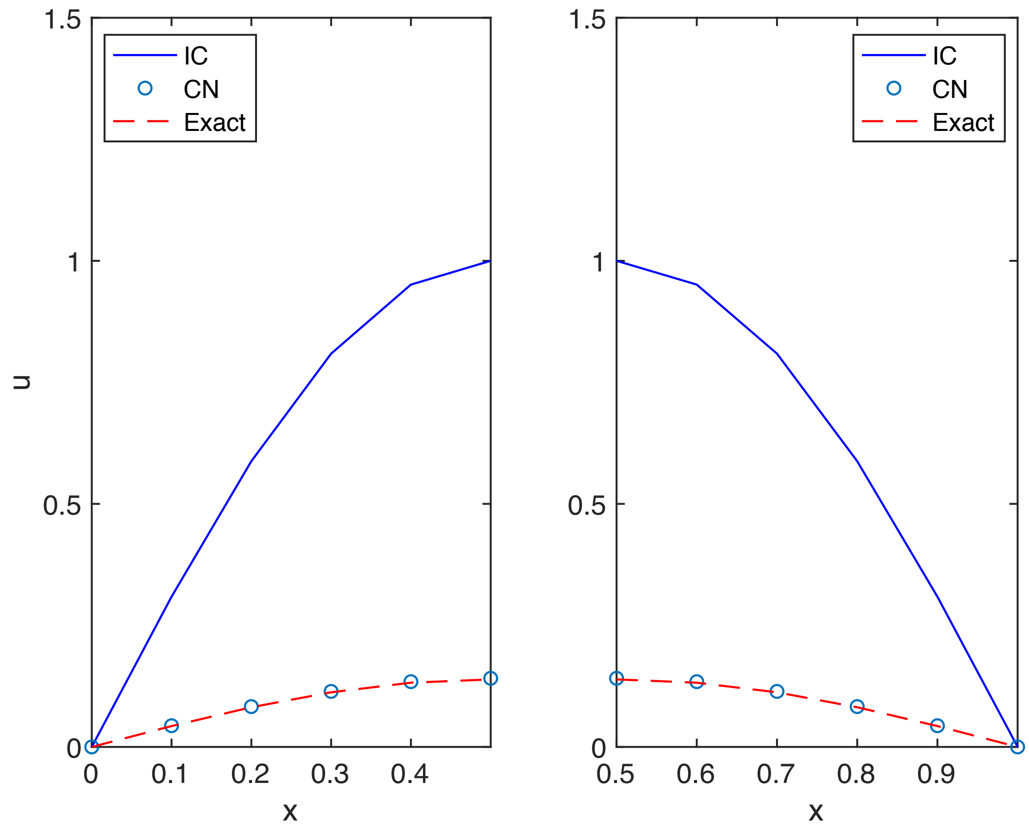
For the half-problem on $L/2 \le x \le L$:

$$\left.\frac{\partial u}{\partial x}\right|_{L/2} = 0 \quad \Longrightarrow \quad \begin{aligned} &\texttt{ubc(1,1)} = 2, && \text{boundary type} \\ &\texttt{ubc(1,2)} = 0, && \text{value of } g_0 \\ &\texttt{ubc(1,3)} = 0, && \text{value of } h_0 \end{aligned}$$

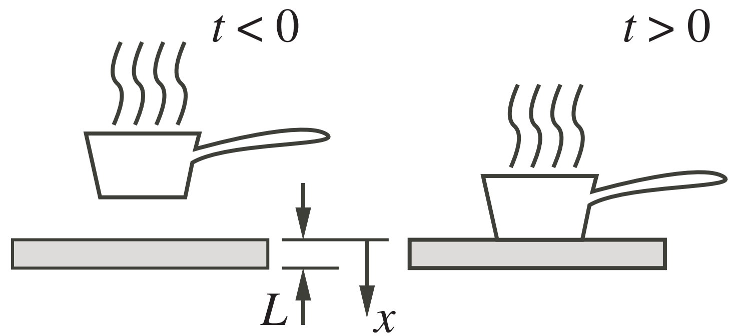
$$u(L,t) = 0 \quad \Longrightarrow \quad \begin{aligned} &\texttt{ubc(2,1)} = 1, && \text{boundary type} \\ &\texttt{ubc(2,2)} = 0, && \text{value of } u \text{ at } x = L \\ &\texttt{ubc(2,3)} = 0, && \text{not used} \end{aligned}$$

Verification: Solve the toy problem on half of the domain

Output of `demoCNBC`



Solve the hot pot problem



Contact resistance at $x = 0$

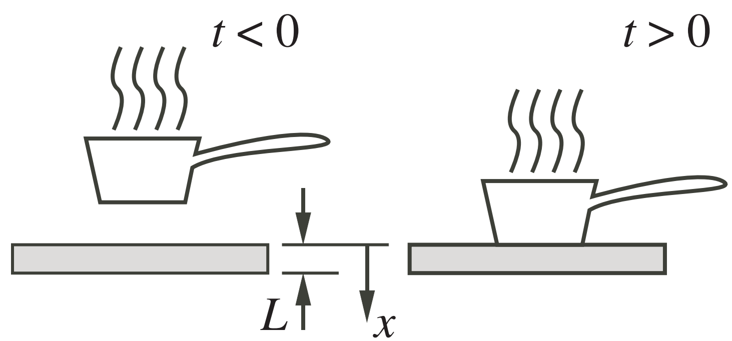
$$-k \left.\frac{\partial T}{\partial x}\right|_{x=0} = q_t = h_t(T_p - T_0)$$

$$\implies \left.\frac{\partial T}{\partial x}\right|_{x=0} = -\frac{h_t T_p}{k} + \frac{h_t}{k} T_0$$

MATLAB boundary matrix for $x = 0$

`ubc(1,1)` $= 2,$	boundary type
`ubc(1,2)` $= -h_t T_p / k,$	value of g_0
`ubc(1,3)` $= h_t / k,$	value of h_0

Solve the hot pot problem



Convective resistance at $x = L$

$$-k \left.\frac{\partial T}{\partial x}\right|_{x=L} = q_b = h_b(T_L - T_{\text{air}})$$

$$\Longrightarrow \left.\frac{\partial T}{\partial x}\right|_{x=L} = \frac{h_b T_{\text{air}}}{k} - \frac{h_b}{k} T_L$$

MATLAB boundary matrix for $x = L$

`ubc(2,1)`	$= 2,$	boundary type
`ubc(2,2)`	$= h_b T_{\text{air}}/k,$	value of g_L
`ubc(2,3)`	$= -h_b/k,$	value of h_L

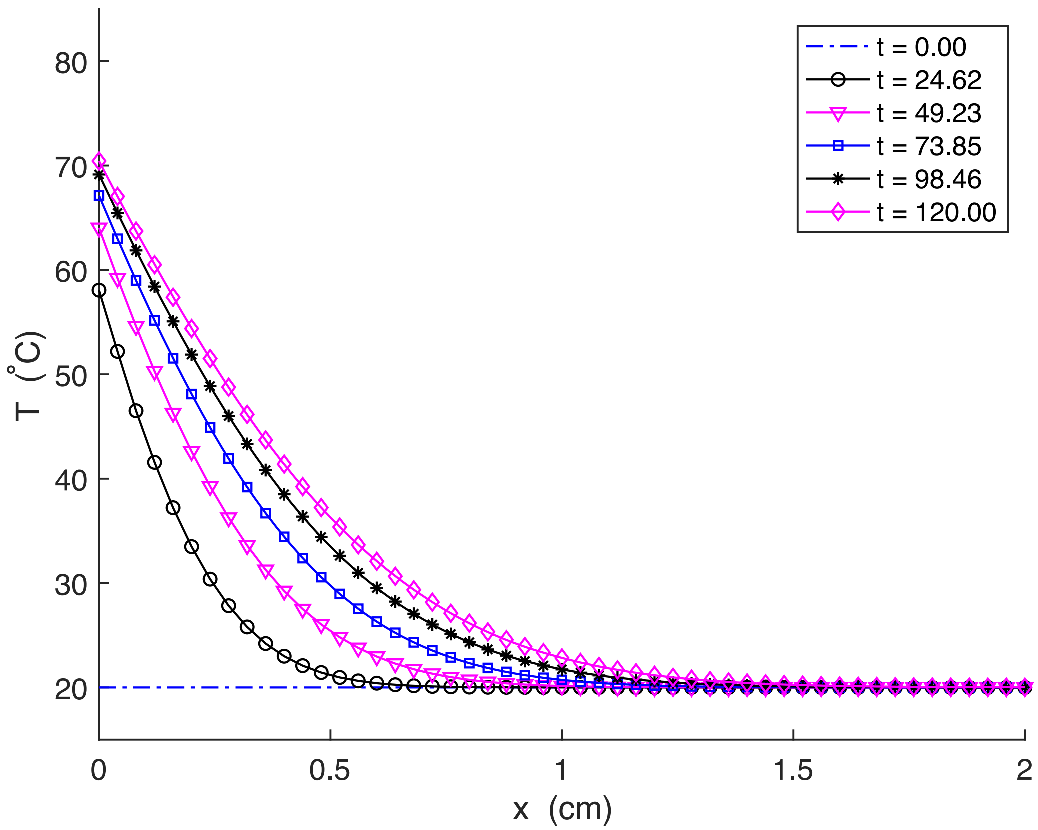
Solve the hot pot problem

Core of `demoHotPot.m`

```
% --- Define physical properties for table and boundary conditions
rho = 545;        %  density of oak (kg/m^3)
k = 0.17;         %  thermal conductivity of oak, across the grain (W/m/C)
c = 2385;         %  specific heat capacity of oak (J/kg/K)
alfa = k/rho/c;   %  thermal diffusivity (m^2/s)
L = 2e-2;         %  Table thickness (m)

% --- Use relaxation time of table material to specify time step size
tau = L^2/alfa;      %  Relaxation time for the heat condution (s)
dt = tau/1000;       %  Time step (s)
nt = ceil(tmax/dt);  %  Number of time steps

% --- Specify initial and boundary conditions
u0 = Tair*ones(nx,1);
ubc = [2 (-htop*Tp/k) (htop/k);  2  (hbot*Tair/k)  (-hbot/k)];

% --- Solve the heat equation and plot the results
[U,x,t] = heatCNBC(nx,nt,ubc,u0,L,tmax,alfa);
plotHeat(U,100*x,t,floor(nt/5))
xlabel('x  (cm)');   ylabel('T  ({}^\circ}C)');  ylim([Tair-5, Tp])
```

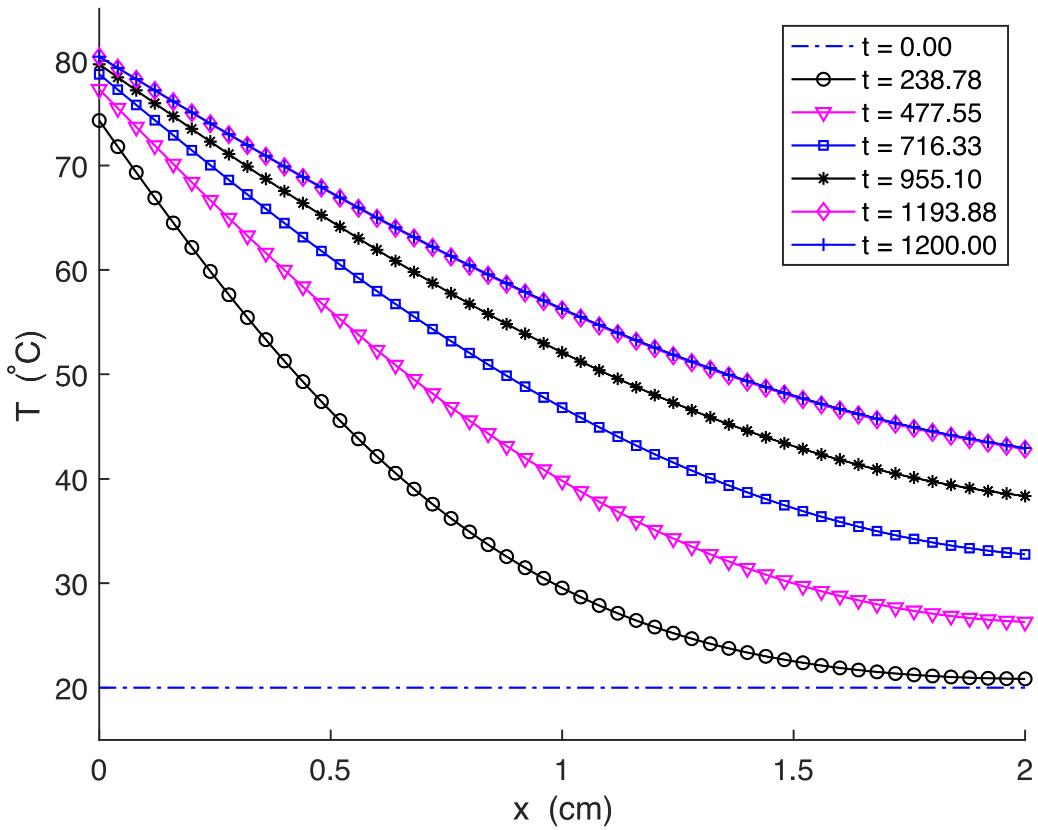
Solve the hot pot problem

```
>> demoHotPot
```



Solve the hot pot problem

```
>> demoHotPot(1200)
```



Solve the hot pot problem

```
>> demoHotPotQw([],[],[],[],1200)
```