```
function [x, t, w] = heatEquation1DBD(D, f, l, r, xdom, tdom, M, N)
% Solves the 1D heat equation in a bar using the Backward Difference Method
%   D: Material diffusion coefficient (aka diffusivity)
%   f: function (of x) describing the initial state of the bar
%   l: function (of time) describing the change in temperature at the left
 boundary of the bar
%   r: function (of time) describing the change in temperature at the right
 boundary of the bar
%   xdom: 1x2 array giving the locations of the boundaries
%   tdom: 1x2 array giving the time interval of interest
%   M: number of grid points along the bar (i.e. along x-coordinate)
%   N: number of grid points along time coordinate
% Discretize domain in time and space
x = linspace(xdom(1), xdom(2), M)';
t = linspace(tdom(1), tdom(2), N)';
%Check that initial and boundary conditions are compatible
TOL = 10^(-9);
if abs(f(x(1)) - l(t(1))) > TOL
    f(x(1))
    l(t(1))
    error('Initial condition does not agree with left boundary condition')
elseif abs(f(x(M)) - r(t(1))) > TOL
    f(x(M))
    r(t(1))
    error('Initial condition does not agree with right boundary condition')
end
% Define step sizes in space and time
h = (xdom(2) - xdom(1))/(M-1);
k = (tdom(2) - tdom(1))/(N-1);
sigma = D*k/(h*h);
% Initialize temperature matrix
w = zeros(N, M);
%Set the initial condition
w(1,:) = f(x);
%Set left and right boundary conditions
w(:,1) = l(t);
w(:,M) = r(t);
%Construct the matrix A
m = M - 2; %number of inner grid points along the bar
% A = diag((1+2*sigma)*ones(m,1));
A = (1+2*sigma)*eye(m);
for i=1:m-1
    A(i,i+1) = -sigma;
    A(i+1,i) = -sigma;
end
sj = zeros(m,1);
% compute temperature at inner nodes at each time step
for j=1:N-1
    wj = w(j,2:m+1)';
    sj(1) = sigma*w(j,1);
    sj(m) = sigma*w(j,M);
```

```
    w(j+1,2:m+1) = A\(wj + sj); %Can also use Jacobi or Gauss-Seidel
 interation since A is strictly row diagonally dominant
end
end
```

```
Not enough input arguments.

Error in heatEquation1DBD (line 13)
x = linspace(xdom(1), xdom(2), M)';
```

Published with MATLAB® R2022a