Derivation of a Numerical Method for solving a single linear parabolic PDE
The Most Elementary Method: The Euler Method

I. FORMULATION.

Consider a linear parabolic partial differential equation in one spatial dimension:

$$d\frac{\partial T}{\partial t} = c\frac{\partial^2 T}{\partial x^2} - aT + \left(\frac{\partial c}{\partial x} - b_x\right)\frac{\partial T}{\partial x} + f \tag{I.1}$$

where the functions, $a, \underline{b}, c, d, f$ are known functions of time, t, and position, x, but not of the temperature, T.

Because the equation is first order in time, this problem requires 1 initial condition of the form

$$T(x,t_o) = T_i(x)$$

Because the equation is second order in space, this problem requires 2 boundary conditions. Boundary conditions come in different forms, but for this problem, let's use the simplest form, called a Dirichlet Boundary Condition, where the temperature at the boundary node is explicitly given.

$$T(x_o,t) = T_o(t) \qquad \text{and} \qquad T(x_f,t) = T_f(t)$$

Our plan is to divide our space dimensions each into m spatial increments, each of width $\frac{L}{m}$ where $L = x_f - x_o$. If we are interested in observing the heat transfer from time t_o to t_f, then we can divide that time into n equal temporal increments, each of width $\frac{t_f - t_o}{n}$. See Figure One.

At the first step, you know all of the function values, T, at time=t_o, because these are given by the initial condition. We also know the values (temperatures, if we assume we are solving the heat equation) at the boundaries for all time. Then what we next want is the temperatures for all interior nodes (all nodes but the 2 nodes with temperatures defined by the boundary conditions at the first time increment, t_1. If we can get $T(t_1,\{x\})$ from $T(t_o,\{x\})$ and $T(t,x_o)$ and $T(t,x_m)$, then we have a formulation which will allow us to incrementally solve the P.D.E through time. Where we could then obtain $T(t_2,\{x\})$ from $T(t_1,\{x\})$ and

$T(t, x_o)$ and $T(t, x_m)$. In general we want to obtain $T(t_{j+1}, \{x\})$ from $T(t_j, \{x\})$ and $T(t, x_o)$ and $T(t, x_m)$.

In this lecture packet, we derive the simplest method for obtaining this solution, based on the Euler method. This is not a commonly used method because it is so inaccurate and unstable. However, it is relatively easy to derive and provides a path toward more sophisticated methods.

A comment on notation: we will write $T(t_j, x_i)$ as T_i^j so that

j superscripts designate temporal increments
i subscripts designate spatial increments

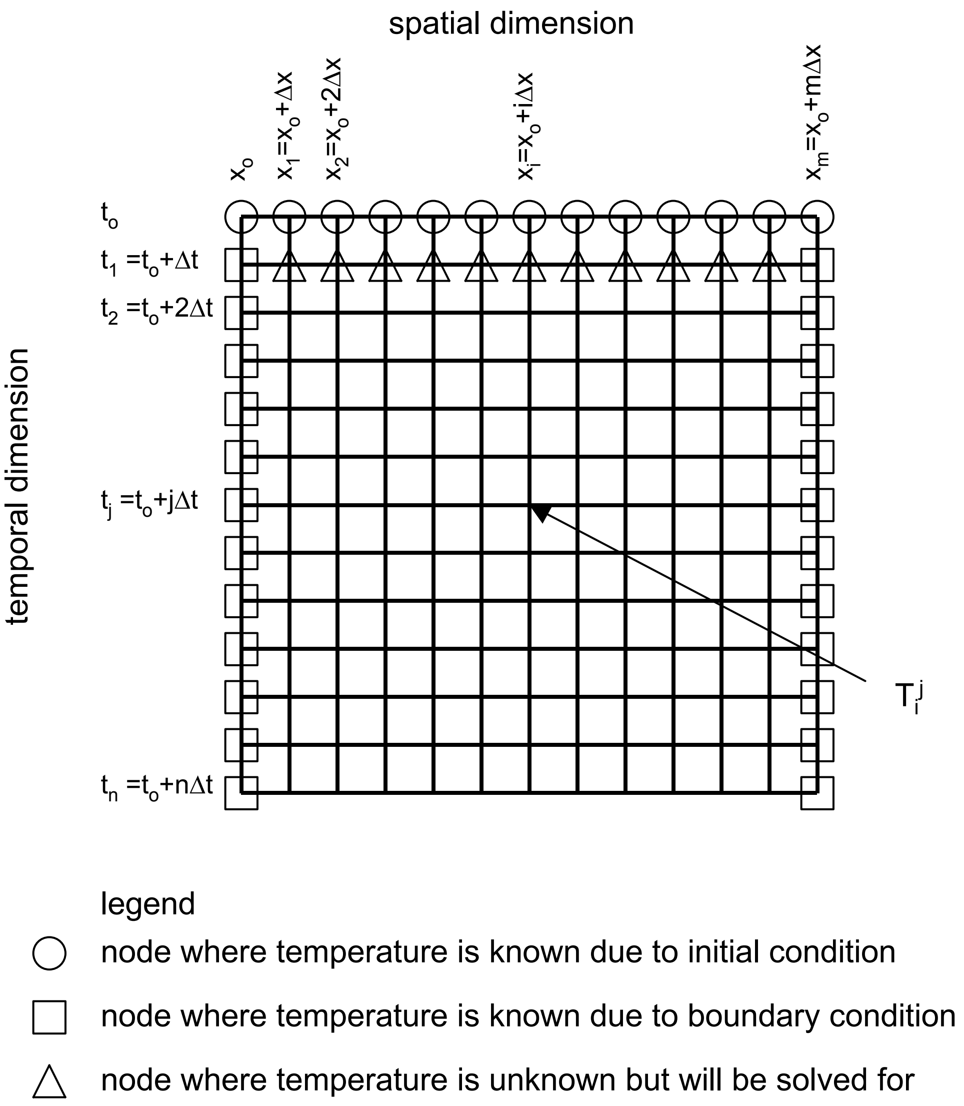


Figure One. Schematic of the spatial and temporal discretization. Case I. Two Dirichlet Boundary Conditions.

II. DISCRETIZATION.

A. The Parabolic partial differential equation.

We write the PDE as

$$\frac{\partial T}{\partial t} = \frac{1}{d}\left[c\frac{\partial^2 T}{\partial x^2} - aT + \left(\frac{\partial c}{\partial x} - b_x\right)\frac{\partial T}{\partial x} + f \right] = K(x,t,T) \tag{II.2}$$

Looking at it in this light, we can obtain a new estimate of the temperature, one increment ahead in time, namely

$$\left(\frac{\partial T}{\partial t}\right)_i \approx \frac{T_i^{j+1} - T_i^j}{t_{j+1} - t_j} = \frac{T_i^{j+1} - T_i^j}{\Delta t} \tag{II.3}$$

This statement is true at any given point i in space. It is a can make a forward finite difference formula of the partial derivative with respect to time, so that:

$$T_i^{j+1} = T_i^j + \Delta t K_i^j \tag{II.5}$$

This is the Euler Method. It is first order in time. This equation can be written in vector form

$$\underline{T}^{j+1} = \underline{T}^j + \Delta t\left(\underline{\underline{K}}^j \underline{T}^j + \underline{R}^j\right) \tag{II.5a}$$

where the vector of unknowns is a column vector of length (m-1) spanning over unknown spatial nodes:

$$\underline{T}^j = \begin{bmatrix} T_1^j \\ \vdots \\ T_{m-1}^j \end{bmatrix}$$

The first and last nodes, at i = 0 and i = m are not included in this vector because they are not unknowns. (See Figure One.) In equation (II.5a), the matrix K contains all terms which are coefficients to the temperature and the residual vector contains all constant terms. We can rewrite this as:

$$\underline{T}^{j+1} = \left(\underline{\underline{I}} + \underline{\underline{K}}^{*j} \right)\underline{T}^{j} + \underline{R}^{*j} \tag{II.5b}$$

where the factor of the time step has been absorbed into the K matrix and the R vector.

To continue, we need to evaluate the right hand side of equation (II.4). For any given point j in time, we can make a finite approximation of the partial derivative with respect to space (centered finite difference formula).

$$\left(\frac{\partial T}{\partial x}\right)_i^j \approx \frac{T_{i+1}^j - T_{i-1}^j}{x_{i+1} - x_{i-1}} = \frac{T_{i+1}^j - T_{i-1}^j}{2\Delta x} \tag{II.6}$$

Moreover, we can use that same formula, again to obtain the second derivative of the temperature with respect to space. (Forward finite difference formula for first derivative with backward finite difference formula for second derivative gives centered finite difference formula.)

$$\left(\frac{\partial^2 T}{\partial x^2}\right)_i^j \approx \frac{\left(\frac{\partial T}{\partial x}\right)_{i+1}^j - \left(\frac{\partial T}{\partial x}\right)_i^j}{x_{i+1} - x_i} = \frac{\left(\frac{\partial T}{\partial x}\right)_{i+1}^j - \left(\frac{\partial T}{\partial x}\right)_i^j}{\Delta x} \tag{II.7}$$

We can substitute our formula for the first spatial derivative into that for the second spatial derivative.

$$\left(\frac{\partial^2 T}{\partial x^2}\right)_i^j \approx \frac{\left(\frac{T_{i+1}^j - T_i^j}{\Delta x}\right) - \left(\frac{T_i^j - T_{i-1}^j}{\Delta x}\right)}{\Delta x} = \frac{T_{i+1}^j - 2T_i^j + T_{i-1}^j}{\Delta x^2} \tag{II.8}$$

This gives the second spatial derivative at time j. We can then obtain the two functions, K_i^{j+1} and c which we require to obtain the temperatures at the new time.

$$K_i^j = \frac{1}{d_i^j}\left[c_i^j\left(\frac{T_{i+1}^j - 2T_i^j + T_{i-1}^j}{\Delta x^2}\right) - a_i^j T_i^j + \left(\left(\frac{\partial c}{\partial x}\right)_i^j - b_{xi}^{\ j}\right)\left(\frac{T_{i+1}^j - T_{i-1}^j}{2\Delta x}\right) + f_i^j \right] \tag{II.9}$$

We now have enough information to fill the matrix and vector in the following equation

$$\underline{T}^{j+1} = \left(\underline{I} + \underline{\underline{K}}^{*j} \right) \underline{T}^{j} + \underline{R}^{*j} \tag{II.5b}$$

$$K^{*j}_{\ell,i} = \begin{cases} K_d = \dfrac{\Delta t}{d_i^j}\left(-\dfrac{2c_i^j}{\Delta x^2} - a_i^j\right) & \text{for } i = \ell, 1 \le i \le m-1 \\ K_{d+} = \dfrac{\Delta t}{d_i^j}\left(\dfrac{c_i^j}{\Delta x^2} + \dfrac{1}{2\Delta x}\left(\dfrac{dc_i^j}{dx} - b_i^j\right)\right) & \text{for } i = \ell+1, 1 \le i \le m-2 \\ K_{d-} = \dfrac{\Delta t}{d_i^j}\left(\dfrac{c_i^j}{\Delta x^2} - \dfrac{1}{2\Delta x}\left(\dfrac{dc_i^j}{dx} - b_i^j\right)\right) & \text{for } i = \ell-1, 2 \le i \le m-1 \\ 0 \quad \text{otherwise} \end{cases}$$

The matrix, $K^{*j}_{\ell,i}$, is tridiagonal. The associated vector of constants is

$$R^{*j}_{i} = \begin{cases} R_d = \dfrac{\Delta t}{d_i^j} f_i^j & \text{for } 2 \le i \le m-2 \\ R_1 = \dfrac{\Delta t}{d_i^j} f_i^j + K_{d-}T_o & \text{for } i = 1 \\ R_{m-1} = \dfrac{\Delta t}{d_i^j} f_i^j + K_{d+}T_f & \text{for } i = m-1 \end{cases}$$

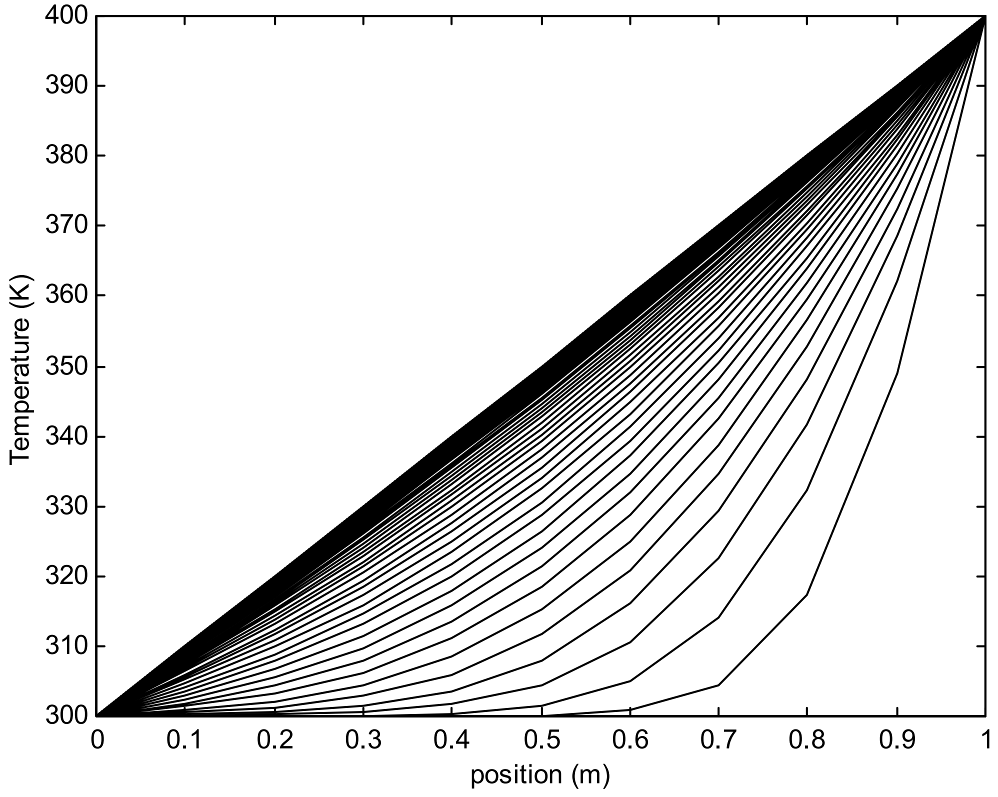
Thus the Euler method for solving a single linear parabolic partial differential equation in one spatial dimension with two Dirichlet boundary conditions is completely derived.

A short code which implements this routine in Matlab is included below:

```
function linparapde_euler
%
%  linparapde_euler
%
% single linear 1-D parabolic PDE with
%  2 Dirichlet Boundary Conditions
%
% d*dT/dt = div(c*grad(T)) - a*T -b*dTdx +f
%
clear all;
close all;
% discretize time
to = 0;
tf = 1.0e-0;
dt = 1.0e-3;
tvec = [to:dt:tf];
nt = length(tvec);

% discretize space
xo = 0;
xf = 1.0;
dx = 0.1;
xvec = [xo:dx:xf];
nx = length(xvec);

% dimension solution
Tmat = zeros(nx,nt);

% dimension temporary vectors
nvar = nx-2;
Told = zeros(nvar,1);
Tnew = zeros(nvar,1);

% apply initial conditions
for i = 1:1:nx
   x = xvec(i);
   Tmat(i,1) = icfunk(x);
end
% apply boundary conditions
for i = 2:1:nt
```

```
   t = tvec(i);
   Tmat(1,i) = bc1funk(t);
   Tmat(nx,i) = bc2funk(t);
end

% loop over times
Told = Tmat(2:1:nx-1,1);
for i = 2:1:nt
   t = tvec(i);
   Kmat = zeros(nvar,nvar);
   Rvec = zeros(nvar,1);
   % diagonal elements of matrix
   for j = 1:1:nvar
      x = xvec(j+1);
      Kmat(j,j) =  dt/dfunk(x,t)* (-2*cfunk(x,t)/dx^2 - afunk(x,t) );
   end
   % upper off-diagonal elements of matrix
   for j = 1:1:nvar-1
      x = xvec(j+1);
      Kmat(j,j+1) =  dt/dfunk(x,t)* (cfunk(x,t)/dx^2 + (dcdxfunk(x,t) - bfunk(x,t))/(2*dx) );
   end
   % lower off-diagonal elements of matrix
   for j = 2:1:nvar
      x = xvec(j+1);
      Kmat(j,j-1) =  dt/dfunk(x,t)* (cfunk(x,t)/dx^2 - (dcdxfunk(x,t) - bfunk(x,t))/(2*dx) );
   end
   % vector of constants
   for j = 1:1:nvar
      x = xvec(j+1);
      Rvec(j) =  dt/dfunk(x,t)*ffunk(x,t);
   end
   % incorporate BCs
   x = xvec(1);
   term = dt/dfunk(x,t)* (cfunk(x,t)/dx^2 - (dcdxfunk(x,t) - bfunk(x,t))/(2*dx) );
   Rvec(1) = Rvec(1) + term*bc1funk(t);
   x = xvec(nx);
   term = dt/dfunk(x,t)* (cfunk(x,t)/dx^2 + (dcdxfunk(x,t) - bfunk(x,t))/(2*dx) );
   Rvec(nvar) = Rvec(nvar) + term*bc2funk(t);
   Tnew = Told + Kmat*Told + Rvec;
   Tmat(2:1:nx-1,i) = Tnew;
   Told = Tnew;
end
% plot
figure(1);
nskip = 10;
for i = 1:nskip:nt
   plot(xvec,Tmat(:,i),'k-');
   %pause(1);
   hold on;
end
xlabel('position (m)')
ylabel('Temperature (K)');


function a = afunk(x,t);
a = 0;

function b = bfunk(x,t);
b = 0;

function c = cfunk(x,t);
c = 1;

function dcdx = dcdxfunk(x,t);
dcdx = 0;

function d = dfunk(x,t);
d = 1;

function f = ffunk(x,t);
f = 000;

function ic = icfunk(x);
ic = 300;

function bc1 = bc1funk(t);
bc1 = 300;

function bc2 = bc2funk(t);
bc2 = 400;
```

Solution: Euler method: $\Delta x = 0.1, \Delta t = 0.001$



Here we plot several temperature profiles for the partial differential equation and initial conditions given by:

$$(d=1)\frac{\partial T}{\partial t} = (c=1)\frac{\partial^2 T}{\partial x^2} - (a=0)T + \left((\frac{\partial c}{\partial x}=0) - (b=0)\right)\frac{\partial T}{\partial x} + (f=0)$$

$T(x,t_o) = T_i(x) = 300$ $\qquad$ $T(x_o,t) = T_o(t) = 300$ $\qquad$ $T(x_f,t) = T_f(t) = 400$

We used 10 spatial intervals and solved from to = 0 to tf = 1, with 1000 temporal intervals. In the plot, we show every tenth profile.

Derivation of a Numerical Method for solving a single linear parabolic PDE
The Second-Order Method: The Crank-Nicholson Method

The Crank-Nicholson Method is a second order Runge-Kutte type method. In this case, we evaluate the slope at the beginning and end of the interval (at time j and time j+1), and average the slopes.

$$T_i^{j+1} = T_i^j + \frac{\Delta t}{2}\left(K_i^{j+1} + K_i^j\right) \tag{III.1}$$

In matrix notation this becomes, where we have now split the function K up into a matrix of coefficients, $\underline{\underline{K}}^j$, and a vector of constants, $\underline{R}^j$.

$$\underline{T}^{j+1} = \underline{T}^j + \frac{\Delta t}{2}\left(\underline{\underline{K}}^{j+1}\underline{T}^{j+1} + \underline{R}^{j+1} + \underline{\underline{K}}^j\underline{T}^j + \underline{R}^j \right) \tag{III.2}$$

If we absorb the factor of half the time step into the matrix K and the vector R, then we can rearrange and write:

$$\left(\underline{\underline{I}} - \underline{\underline{K}}^{*\,j+1} \right)\underline{T}^{j+1} = \left(\underline{\underline{I}} + \underline{\underline{K}}^{*\,j} \right)\underline{T}^j + \underline{R}^{*\,j+1} + \underline{R}^{*\,j} \tag{III.3}$$

where the asterisk indicates that we have included the factor of half the time step in the matrix. We should note that in this manner the matrix of coefficients and the vector of constants have the same definition as that in the Euler method, except for an additional factor of ½.

Solving for the vector of unknowns, $\underline{T}^{j+1}$, we have

$$\underline{T}^{j+1} = \left(\underline{\underline{I}} - \underline{\underline{K}}^{*\,j+1} \right)^{-1}\left[\left(\underline{\underline{I}} + \underline{\underline{K}}^{*\,j} \right)\underline{T}^j + \underline{R}^{*\,j+1} + \underline{R}^{*\,j}\right] \tag{III.4}$$

The advantage of this method is that it is second order. We can take larger time steps, without the method blowing up.

Below, we present a MATLAB code to implement the Crank-Nicholson method. This code is 90% the same as the Euler code. We have moved the calculation of K and R into function, to conserve space.

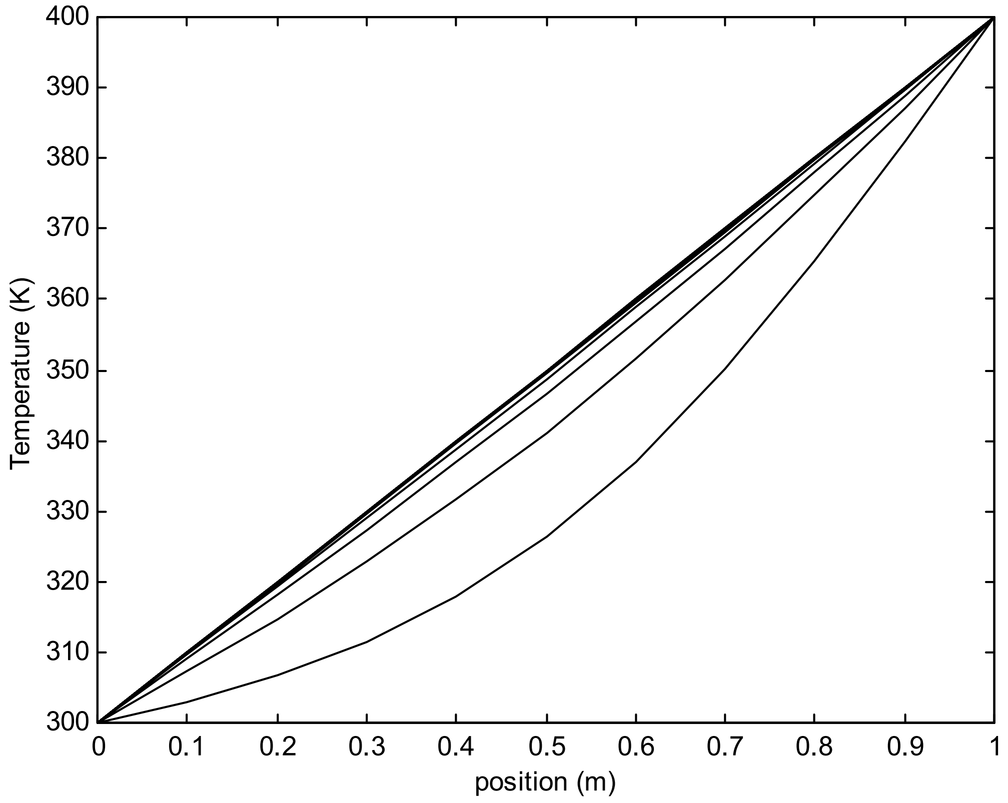
```
function linparapde_crank
%
%  linparapde_crank
%
% single linear 1-D parabolic PDE with
%  2 Dirichlet Boundary Conditions
%
% d*dT/dt = div(c*grad(T)) - a*T -b*dTdx +f
%
clear all;
close all;
% discretize time
to = 0;
tf = 1.0e-0;
dt = 1.0e-2;
tvec = [to:dt:tf];
nt = length(tvec);

% discretize space
xo = 0;
xf = 1.0;
dx = 0.1;
xvec = [xo:dx:xf];
nx = length(xvec);

% dimension solution
Tmat = zeros(nx,nt);

% dimension temporary vectors
nvar = nx-2;
Told = zeros(nvar,1);
Tnew = zeros(nvar,1);

% apply initial conditions
for i = 1:1:nx
   x = xvec(i);
   Tmat(i,1) = icfunk(x);
end

% apply boundary conditions
```

```
for i = 2:1:nt
   t = tvec(i);
   Tmat(1,i) = bc1funk(t);
   Tmat(nx,i) = bc2funk(t);
end

%
%  make an identity matrix for later use
%
Id = zeros(nvar,nvar);
for i = 1:1:nvar
   Id(i,i) =1.0;
end

% loop over times
i = 1;
t = tvec(i);
Told = Tmat(2:1:nx-1,i);
Kmatold = 0.5*getK(nvar,t,xvec,dt,dx);
Rvecold = 0.5*getR(nvar,t,xvec,dt,dx,nx);
for i = 2:1:nt
   t = tvec(i);
   Kmatnew = 0.5*getK(nvar,t,xvec,dt,dx);
   Rvecnew = 0.5*getR(nvar,t,xvec,dt,dx,nx);
   Ainv = inv(Id - Kmatnew);
   % vector of constants
   Tnew = Ainv*((Id + Kmatold)*Told + Rvecnew + Rvecold);
   Tmat(2:1:nx-1,i) = Tnew;
   Told = Tnew;
   Kmatold = Kmatnew;
   Rvecold = Rvecnew;
end
% plot
figure(1);
nskip = 10;
for i = 1:nskip:nt
   plot(xvec,Tmat(:,i),'k-');
   %pause(1);
   hold on;
end
```

```
xlabel('position (m)')
ylabel('Temperature (K)');

function Kmat = getK(nvar,t,xvec,dt,dx);
   Kmat = zeros(nvar,nvar);
   % diagonal elements of matrix
   for j = 1:1:nvar
      x = xvec(j+1);
      Kmat(j,j) =  dt/dfunk(x,t)* (-2*cfunk(x,t)/dx^2 - afunk(x,t) );
   end
   % upper off-diagonal elements of matrix
   for j = 1:1:nvar-1
      x = xvec(j+1);
      Kmat(j,j+1) =  dt/dfunk(x,t)* (cfunk(x,t)/dx^2 + (dcdxfunk(x,t) - bfunk(x,t))/(2*dx) );
   end
   % lower off-diagonal elements of matrix
   for j = 2:1:nvar
      x = xvec(j+1);
      Kmat(j,j-1) =  dt/dfunk(x,t)* (cfunk(x,t)/dx^2 - (dcdxfunk(x,t) - bfunk(x,t))/(2*dx) );
   end

   function Rvec = getR(nvar,t,xvec,dt,dx,nx);
   Rvec = zeros(nvar,1);
   for j = 1:1:nvar
      x = xvec(j+1);
      Rvec(j) =  dt/dfunk(x,t)*ffunk(x,t);
   end
   % incorporate BCs
   x = xvec(1);
   term = dt/dfunk(x,t)* (cfunk(x,t)/dx^2 - (dcdxfunk(x,t) - bfunk(x,t))/(2*dx) );
   Rvec(1) = Rvec(1) + term*bc1funk(t);
   x = xvec(nx);
   term = dt/dfunk(x,t)* (cfunk(x,t)/dx^2 + (dcdxfunk(x,t) - bfunk(x,t))/(2*dx) );
   Rvec(nvar) = Rvec(nvar) + term*bc2funk(t);

function a = afunk(x,t);
a = 0;

function b = bfunk(x,t);
b = 0;
```

```
function c = cfunk(x,t);
c = 1;

function dcdx = dcdxfunk(x,t);
dcdx = 0;

function d = dfunk(x,t);
d = 1;

function f = ffunk(x,t);
f = 000;

function ic = icfunk(x);
ic = 300;

function bc1 = bc1funk(t);
bc1 = 300;

function bc2 = bc2funk(t);
bc2 = 400;
```

Solution: Crank-Nicholson method: $\Delta x = 0.1, \Delta t = 0.01$



Here we plot several temperature profiles for the partial differential equation and initial conditions given by:

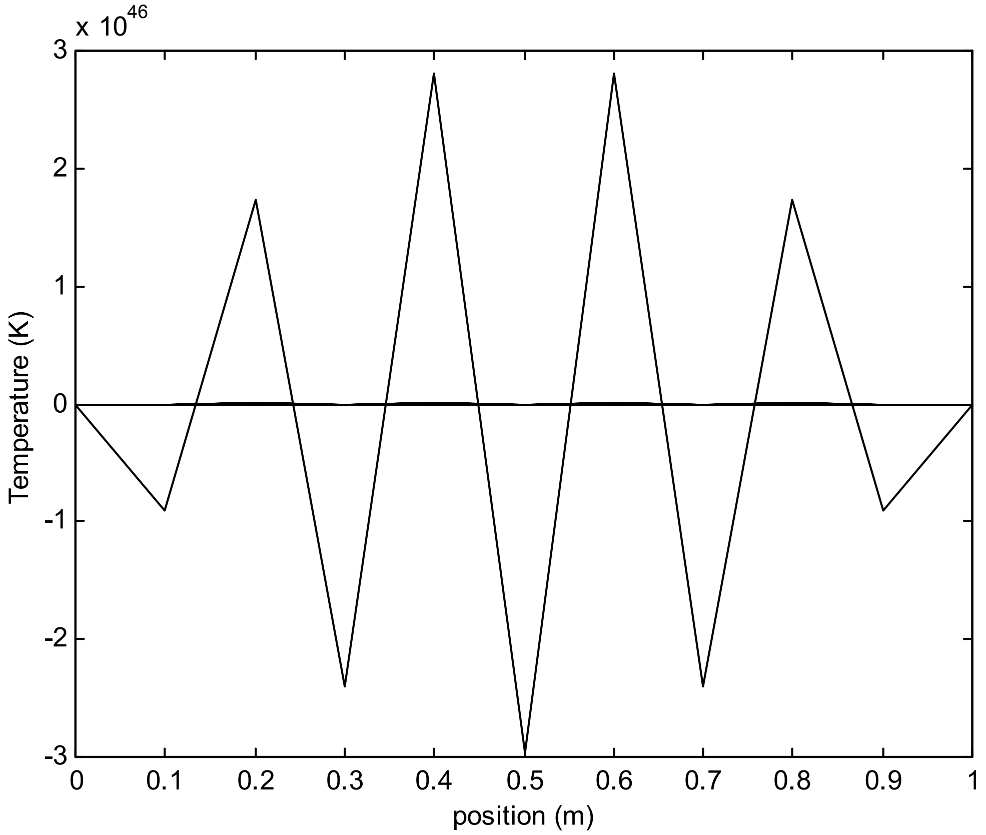
$$(d=1)\frac{\partial T}{\partial t} = (c=1)\frac{\partial^2 T}{\partial x^2} - (a=0)T + \left((\frac{\partial c}{\partial x} = 0) - (b=0) \right)\frac{\partial T}{\partial x} + (f=0)$$

$$T(x,t_o) = T_i(x) = 300 \qquad T(x_o,t) = T_o(t) = 300 \qquad T(x_f,t) = T_f(t) = 400$$

We used 10 spatial intervals and solved from to = 0 to tf = 1, with 100 temporal intervals. In the plot, we show every tenth profile.

As a comparison, if we try to solve the first order, Euler method with a similar spatial and temporal interval ($\Delta x = 0.1, \Delta t = 0.01$), then we will find that the method crashes. A plot of the crash is shown in the next figure.

Solution: Euler method: $\Delta x = 0.1, \Delta t = 0.01$



Here we plot several temperature profiles for the partial differential equation and initial conditions given by:

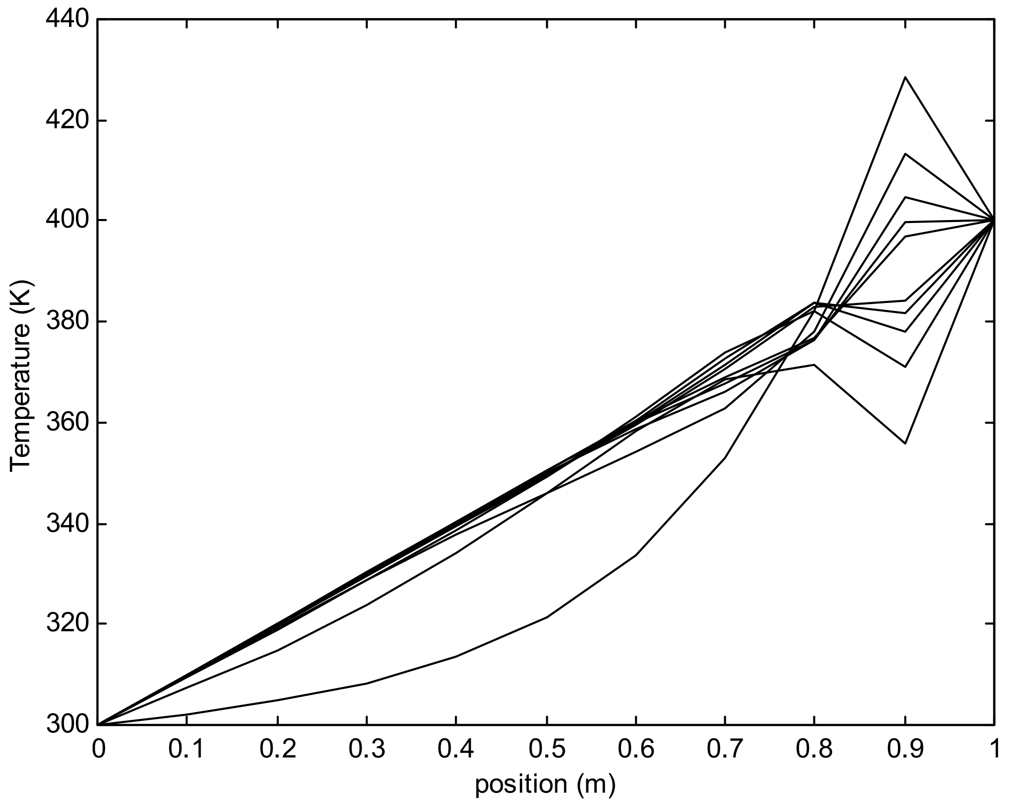
$$(d=1)\frac{\partial T}{\partial t} = (c=1)\frac{\partial^2 T}{\partial x^2} - (a=0)T + \left((\frac{\partial c}{\partial x}=0) - (b=0) \right)\frac{\partial T}{\partial x} + (f=0)$$

$T(x,t_o) = T_i(x) = 300$ $\qquad$ $T(x_o,t) = T_o(t) = 300$ $\qquad$ $T(x_f,t) = T_f(t) = 400$

We used 10 spatial intervals and solved from to = 0 to tf = 1, with 100 temporal intervals. In the plot, we show every tenth profile.

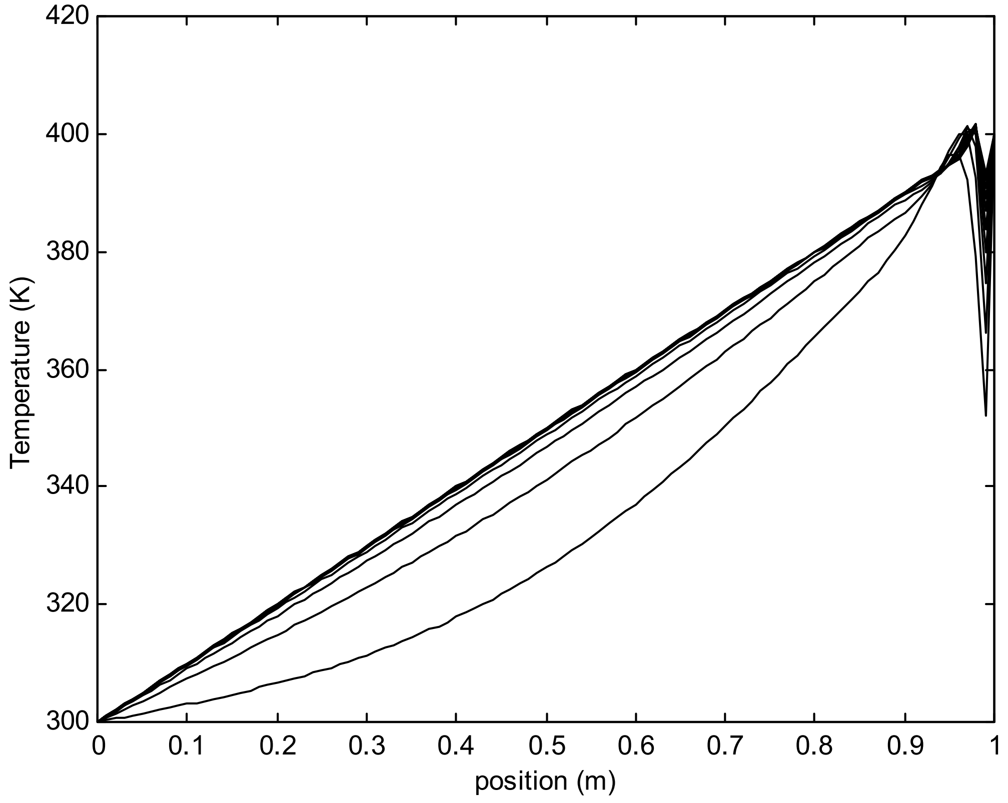
As is clearly evident, the method was unstable and the result is rubbish.

Solution: Crank-Nicholson method: $\Delta x = 0.1, \Delta t = 0.1$



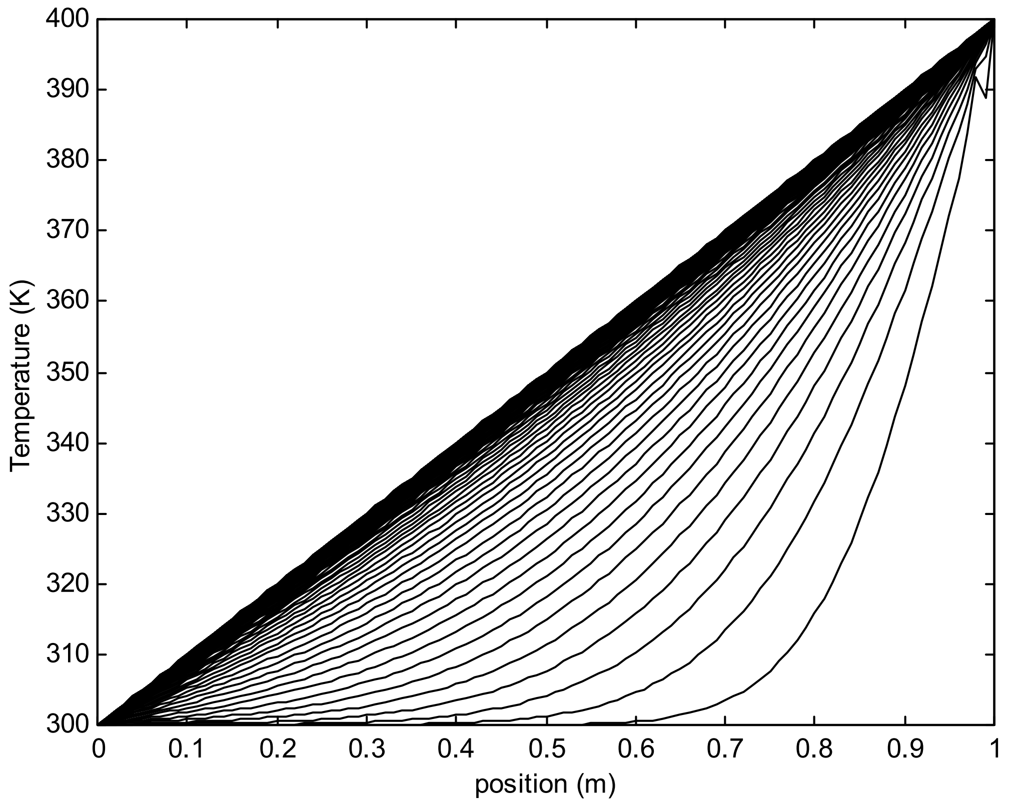
Clearly the time step of 0.1 sec is too small for the spatial interval of 0.1 meters.

Solution: Crank-Nicholson method: $\Delta x = 0.01, \Delta t = 0.01$



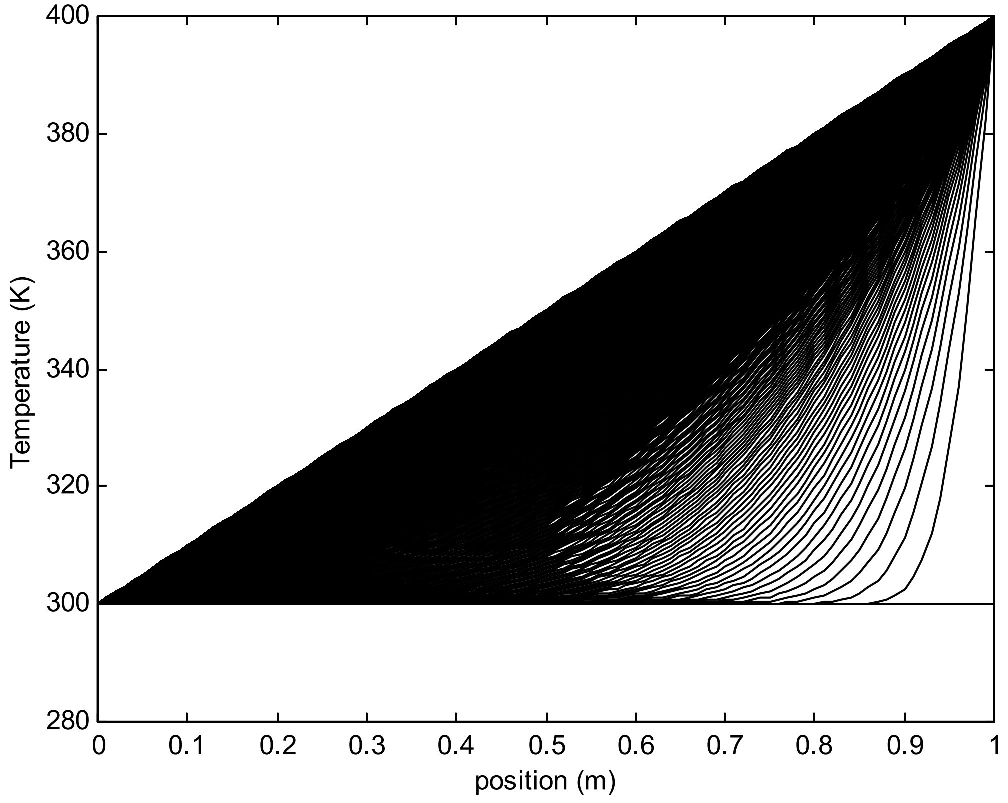
Finer meshes require finer time steps. If we drop the spatial interval to 0.01 meters, then the time step of 0.01 seconds, which previously was adequate for a spatial interval of 0.1 meters, is no longer sufficient.

Solution: Crank-Nicholson method: $\Delta x = 0.01, \Delta t = 0.001$



Finer meshes require finer time steps. If we drop the spatial interval to 0.01 meters, then the time step of 0.001 seconds is no longer sufficient, as can be seen from the spurious behavior at x=1.0 m.

Solution: Crank-Nicholson method: $\Delta x = 0.01, \Delta t = 0.0001$



Finer meshes require finer time steps. If we drop the spatial interval to 0.01 meters, then the time step of 0.0001 seconds yields a reasonable result. (More lines are shown here because the code was set to draw every tenth profile, and we have 10,000 steps in this particular case.)