Derivation of a Numerical Method for solving a single parabolic PDE in two spatial dimensions

David Keffer
Department of Chemical Engineering
University of Tennessee, Knoxville
May-September 1999

Table of Contents

I. FORMULATION.

Linear parabolic partial differential equations are, in their most general form, given by:

$$d\frac{\partial T}{\partial t} = \nabla \cdot [c(\nabla T)] - aT - \underline{b} \cdot \nabla T + f \tag{I.1}$$

where the functions, $a, \underline{b}, c, d, f$ are known functions of t, x, y, z and the temperature, T, is an unknown function of t, x, y, z. In order to approximate this creature using a finite difference scheme we recognize that

$$\nabla \cdot [c(\nabla T)] = c\nabla \cdot \nabla T + \nabla T \cdot \nabla c = c\nabla^2 T + \nabla T \cdot \nabla c = c\nabla^2 T + \nabla c \cdot \nabla T \tag{I.2}$$

and rewrite this as:

$$d\frac{\partial T}{\partial t} = c\nabla^2 T + \nabla c \cdot \nabla T - aT - \underline{b} \cdot \nabla T + f \tag{I.3}$$

$$d\frac{\partial T}{\partial t} = c\nabla^2 T - aT + (\nabla c - \underline{b}) \cdot \nabla T + f \tag{I.4}$$

We will consider the case with variation in two spatial dimensions. The extension to three dimensions is straightforward. Our most general parabolic PDE becomes in two spatial dimension

$$d\frac{\partial T}{\partial t} = c\left(\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2}\right) - aT + \left(\frac{\partial c}{\partial x} - b_x\right)\frac{\partial T}{\partial x} + \left(\frac{\partial c}{\partial y} - b_y\right)\frac{\partial T}{\partial y} + f \tag{I.5}$$

We now need to know the functional forms of $a, \underline{b}, c, d, f, \nabla c$, which must be given. In many problems, most of these functions are constants and, often the constants are unity or zero. However, in order to write a code that solves any parabolic PDE, it is for this general formulation that we derive a finite-difference method.

Our plan is to divide our x-spatial dimension into m_x spatial increments, each of width $\frac{L_x}{m_x}$.

Similarly, we divide our x-spatial dimension into m_y spatial increments, each of width $\frac{L_y}{m_y}$. 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. Let's first consider the case where we have a rectangular area with 4 Dirichlet boundary conditions. In that case, we also know the values (temperatures, if we assume we are solving the heat equation) at the edges of the area for all time. Then what we next want is the temperatures for all interior nodes (all nodes but the nodes with temperatures defined by the boundary conditions at the first time increment, t_1. If we can get $T(t_1,\{x,y\})$ from $T(t_o,\{x,y\})$ and the boundary conditions: $T(t,x_o,y)$, $T(t,x_{m_x},y)$, $T(t,x,y_o)$, $T(t,x,y_{m_y})$, then we have a formulation which will allow us to incrementally solve the P.D.E through time. Where we could then obtain $T(t_2,\{x,y\})$ from $T(t_1,\{x,y\})$ and the boundary conditions. In general we want to obtain $T(t_{j+1},\{x\})$ from $T(t_j,\{x\})$ and $T(t,x_o)$ and $T(t,x_m)$.

We will derive one such method, a direct two-dimensional analog of the one-dimensional method known as the Crank-Nicolson method.

A comment on notation: we will write $T(t_j,x_i,y_k)$ as $T^j_{i,k}$ so that
j superscripts designate temporal increments
i subscripts designate spatial increments along the x coordinate
k subscripts designate spatial increments along the y coordinate

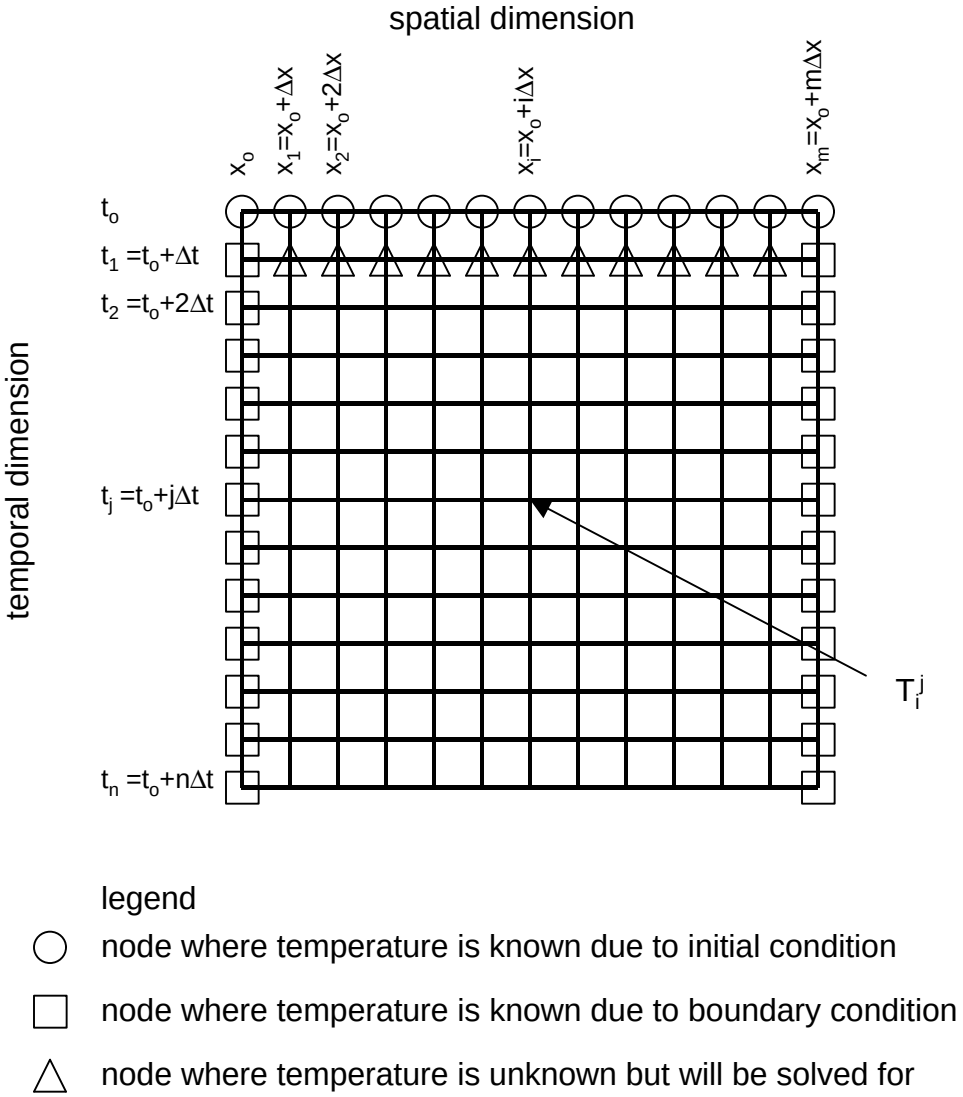


Figure One. Schematic of the spatial and temporal discretization. Case I. Two Dirichlet Boundary Conditions.

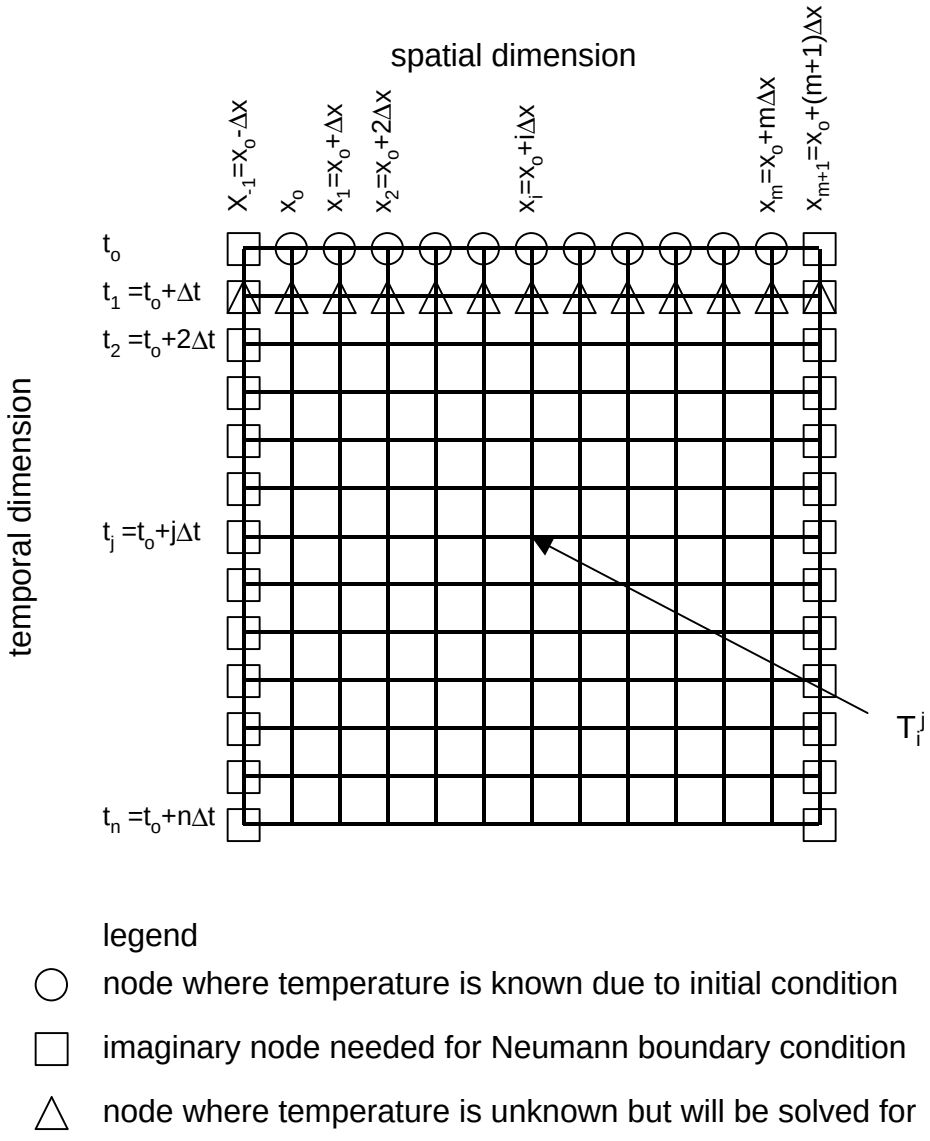


Figure Two. Schematic of the spatial and temporal discretization. Case II. Two Neumann Boundary Conditions.

II. DISCRETIZATION.

Derivation of the Crank-Nicolson finite difference equations

A. The Parabolic partial differential equation.

The Crank-Nicolson finite difference equations provide estimates that are second order in space and time. The Crank-Nicolson is an implicit method.

Let j superscripts designate temporal increments and let i subscripts designate spatial increments. For purposes of brevity only, we will consider the case with variation only in one spatial dimension. The extension to three dimensions is straightforward. Our most general parabolic PDE becomes in one spatial dimension

$$d\frac{\partial T}{\partial t} = c\left(\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2}\right) - aT + \left(\frac{\partial c}{\partial x} - b_x\right)\frac{\partial T}{\partial x} + \left(\frac{\partial c}{\partial y} - b_y\right)\frac{\partial T}{\partial y} + f \qquad \text{(II.1)}$$

In order to get an approximation that is second order in time, we recast this equation as

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

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

$$\left(\frac{\partial T}{\partial t}\right)_{i,k} \approx \frac{T_{i,k}^{j+1} - T_{i,k}^{j}}{t_{j+1} - t_j} = \frac{T_{i,k}^{j+1} - T_{i,k}^{j}}{\Delta t} \qquad \text{(II.3)}$$

This statement is true at any given point i in space. It is a forward finite difference formula of the partial derivative with respect to time. Now what is also true by the second-order Runge-Kutta method is that:

$$\left(\frac{\partial T}{\partial t}\right)_{i,k} \approx \frac{1}{2}\left[K(x_i, y_k, t_{j+1}, T_i^{j+1}) + K(x_i, y_k, t_{j+1}, T_i^{j+1})\right] = \frac{1}{2}\left[K_{i,k}^{j+1} + K_{i,k}^{j}\right] \qquad \text{(II.4)}$$

so that:

$$T_{i,k}^{j+1} = T_{i,ki}^{j} + \frac{\Delta t}{2}\left[K_{i,ki}^{j+1} + K_{i,k}^{j}\right] \qquad \text{(II.5)}$$

This is the formula for the second order Runge-Kutta. Ordinarily, we wouldn't know the temperature needed to evalute $K_{i,k}^{j+1}$ (and we would be forced to approximate it) but we shall see

that we can formulate the problem in a linear fashion so that we can implicitly solve the right and left-hand sides of this equation simultaneously, without further approximation.

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).

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

$$\left(\frac{\partial T}{\partial y}\right)_{i,k}^{j} \approx \frac{T_{i,k+1}^{j} - T_{i,k-1}^{j}}{y_{k+1} - y_{k-1}} = \frac{T_{i,k+1}^{j} - T_{i,k-1}^{j}}{2\Delta y} \tag{II.6y}$$

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.)

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

$$\left(\frac{\partial^2 T}{\partial y^2}\right)_{i,k}^{j} \approx \frac{\left(\frac{\partial T}{\partial y}\right)_{i,k+1}^{j} - \left(\frac{\partial T}{\partial y}\right)_{i,k}^{j}}{y_{k+1} - y_{i}} = \frac{\left(\frac{\partial T}{\partial y}\right)_{i,k+1}^{j} - \left(\frac{\partial T}{\partial y}\right)_{i,k}^{j}}{\Delta y} \tag{II.7y}$$

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

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

$$\left(\frac{\partial^2 T}{\partial y^2}\right)_{i,k}^{j} \approx \frac{\left(\frac{T_{i,k+1}^{j} - T_{i,k}^{j}}{\Delta y}\right) - \left(\frac{T_{i,k}^{j} - T_{i,k-1}^{j}}{\Delta y}\right)}{\Delta y} = \frac{T_{i,k+1}^{j} - 2T_{i,k}^{j} + T_{i,k-1}^{j}}{\Delta y^2} \tag{II.8y}$$

This gives the second spatial derivatives at time j. We can then obtain the two functions, $K_{i,k}^{j+1}$ and $K_{i,k}^{j}$ which we require to obtain the temperatures at the new time.

$$K_{i,k}^{j}=\frac{1}{d_{i,k}^{j}}\left[\begin{array}{l} c_{i,k}^{j}\left(\frac{T_{i+1,k}^{j}-2T_{i,k}^{j}+T_{i-1,k}^{j}}{\Delta x^{2}}+\frac{T_{i,k+1}^{j}-2T_{i,k}^{j}+T_{i,k-1}^{j}}{\Delta y^{2}}\right)-a_{i,k}^{j}T_{i,k}^{j} \\ +\left(\left(\frac{\partial c}{\partial x}\right)_{i,k}^{j}-b_{x_{i,k}}^{\ j}\right)\left(\frac{T_{i+1,k}^{j}-T_{i-1,k}^{j}}{2\Delta x}\right)+\left(\left(\frac{\partial c}{\partial y}\right)_{i,k}^{j}-b_{y_{i,k}}^{\ j}\right)\left(\frac{T_{i,k+1}^{j}-T_{i,k-1}^{j}}{2\Delta y}\right)+f_{i,k}^{j} \end{array}\right] \quad \text{(II.9)}$$

$$K_{i,k}^{j+1}=\frac{1}{d_{i,k}^{j}}\left[\begin{array}{l} c_{i,k}^{j+1}\left(\frac{T_{i+1,k}^{j+1}-2T_{i,k}^{j+1}+T_{i-1,k}^{j+1}}{\Delta x^{2}}+\frac{T_{i,k+1}^{j+1}-2T_{i,k}^{j+1}+T_{i,k-1}^{j+1}}{\Delta y^{2}}\right)-a_{i,k}^{j+1}T_{i,k}^{j+1} \\ +\left(\left(\frac{\partial c}{\partial x}\right)_{i,k}^{j+1}-b_{x_{i,k}}^{\ j+1}\right)\left(\frac{T_{i+1,k}^{j+1}-T_{i-1,k}^{j+1}}{2\Delta x}\right)+\left(\left(\frac{\partial c}{\partial y}\right)_{i,k}^{j+1}-b_{y_{i,k}}^{\ j+1}\right)\left(\frac{T_{i,k+1}^{j+1}-T_{i,k-1}^{j+1}}{2\Delta y}\right)+f_{i,k}^{j+1} \end{array}\right] \quad \text{(II.10)}$$

Define

$$A=\frac{\Delta t a}{2d},\ B_x=\frac{\Delta t}{4d\Delta x}\left(\frac{\partial c}{\partial x}-b_x\right),\ B_y=\frac{\Delta t}{4d\Delta y}\left(\frac{\partial c}{\partial y}-b_y\right),C_x=\frac{\Delta t}{2d\Delta x^{2}}c,$$

$$C_y=\frac{\Delta t}{2d\Delta y^{2}}c,\ D=1,\text{ and }F=\frac{\Delta t f}{2d}$$

so that we can rewrite equations (II.9) and (II.10) as

$$K_{i,k}^{j}=\frac{2}{\Delta t}\left[\begin{array}{l} C_{x_{i,k}}^{\ j}\left(T_{i+1,k}^{j}-2T_{i,k}^{j}+T_{i-1,k}^{j}\right)+C_{y_{i,k}}^{\ j}\left(T_{i,k+1}^{j}-2T_{i,k}^{j}+T_{i,k-1}^{j}\right) \\ -A_{i,k}^{j}T_{i,k}^{j}+B_{x_{i,k}}^{\ j}\left(T_{i+1,k}^{j}-T_{i-1,k}^{j}\right)+B_{y_{i,k}}^{\ j}\left(T_{i,k+1}^{j}-T_{i,k-1}^{j}\right)+F_{i,k}^{j} \end{array}\right] \quad \text{(II.9b)}$$

$$K_{i,k}^{j+1}=\frac{2}{\Delta t}\left[\begin{array}{l} C_{x_{i,k}}^{\ j+1}\left(T_{i+1,k}^{j+1}-2T_{i,k}^{j+1}+T_{i-1,k}^{j+1}\right)+C_{y_{i,k}}^{\ j+1}\left(T_{i,k+1}^{j+1}-2T_{i,k}^{j+1}+T_{i,k-1}^{j+1}\right) \\ -A_{i,k}^{j+1}T_{i,k}^{j+1}+B_{x_{i,k}}^{\ j+1}\left(T_{i+1,k}^{j+1}-T_{i-1,k}^{j+1}\right)+B_{y_{i,k}}^{\ j+1}\left(T_{i,k+1}^{j+1}-T_{i,k-1}^{j+1}\right)+F_{i,k}^{j+1} \end{array}\right] \quad \text{(II.10b)}$$

Group like temperatures in equations (II.9b) and (II.10b)

$$K_{i,k}^{j}=\frac{2}{\Delta t}\left[\begin{array}{l} \left(C_{x_{i,k}}^{\ j}-B_{x_{i,k}}^{\ j}\right)T_{i-1,k}^{j}+\left(C_{y_{i,k}}^{\ j}-B_{y_{i,k}}^{\ j}\right)T_{i,k-1}^{j}+\left(-2C_{x_{i,k}}^{\ j}-2C_{y_{i,k}}^{\ j}-A_{i,k}^{j}\right)T_{i,k}^{j} \\ +\left(C_{x_{i,k}}^{\ j}+B_{x_{i,k}}^{\ j}\right)T_{i+1,k}^{j}+\left(C_{y_{i,k}}^{\ j}+B_{y_{i,k}}^{\ j}\right)T_{i,k+1}^{j}+F_{i,k}^{j} \end{array}\right] \quad \text{(II.9c)}$$

$$K_{i,k}^{j+1} = \frac{2}{\Delta t}\left[\begin{array}{l}\left(C_{xi,k}^{\ j+1} - B_{xi,k}^{\ j+1}\right) T_{i-1,k}^{j+1} + \left(C_{y_{i,k}}^{\ j+1} - B_{y_{i,k}}^{\ j+1}\right) T_{i,k-1}^{j+1} + \left(-2C_{xi,k}^{\ j+1} - 2C_{y_{i,k}}^{\ j+1} - A_{i,k}^{j+1}\right) T_{i,k}^{j+1} \\ + \left(C_{xi,k}^{\ j+1} + B_{xi,k}^{\ j+1}\right) T_{i+1,k}^{j+1} + \left(C_{y_{i,k}}^{\ j+1} + B_{y_{i,k}}^{\ j+1}\right) T_{i,k+1}^{j+1} + F_{i,k}^{j+1}\end{array}\right]$$

(II.10c)

Substitute equations (II.9c) and (II.10c) into (II.5) to obtain

$$T_{i,k}^{j+1} = T_{i,k}^{j} + \left[\begin{array}{l}\left(C_{xi,k}^{\ j} - B_{xi,k}^{\ j}\right) T_{i-1,k}^{j} + \left(C_{y_{i,k}}^{\ j} - B_{y_{i,k}}^{\ j}\right) T_{i,k-1}^{j} + \left(-2C_{xi,k}^{\ j} - 2C_{y_{i,k}}^{\ j} - A_{i,k}^{j}\right) T_{i,k}^{j} \\ + \left(C_{xi,k}^{\ j} + B_{xi,k}^{\ j}\right) T_{i+1,k}^{j} + \left(C_{y_{i,k}}^{\ j} + B_{y_{i,k}}^{\ j}\right) T_{i,k+1}^{j} + F_{i,k}^{j} \ + \\ \left(C_{xi,k}^{\ j+1} - B_{xi,k}^{\ j+1}\right) T_{i-1,k}^{j+1} + \left(C_{y_{i,k}}^{\ j+1} - B_{y_{i,k}}^{\ j+1}\right) T_{i,k-1}^{j+1} + \left(-2C_{xi,k}^{\ j+1} - 2C_{y_{i,k}}^{\ j+1} - A_{i,k}^{j+1}\right) T_{i,k}^{j+1} \\ + \left(C_{xi,k}^{\ j+1} + B_{xi,k}^{\ j+1}\right) T_{i+1,k}^{j+1} + \left(C_{y_{i,k}}^{\ j+1} + B_{y_{i,k}}^{\ j+1}\right) T_{i,k+1}^{j+1} + F_{i,k}^{j+1}\end{array}\right] \tag{II.11}$$

Group like temperatures, placing all unknown temperatures (at time j+1) on the right-hand side and all known temperatures (at time j) on the left-hand side.

$$\begin{aligned}&\left[\begin{array}{l}-\left(C_{xi,k}^{\ j+1} - B_{xi,k}^{\ j+1}\right) T_{i-1,k}^{j+1} - \left(C_{y_{i,k}}^{\ j+1} - B_{y_{i,k}}^{\ j+1}\right) T_{i,k-1}^{j+1} + \left(1 + 2C_{xi,k}^{\ j+1} + 2C_{y_{i,k}}^{\ j+1} + A_{i,k}^{j+1}\right) T_{i,k}^{j+1} \\ - \left(C_{xi,k}^{\ j+1} + B_{xi,k}^{\ j+1}\right) T_{i+1,k}^{j+1} - \left(C_{y_{i,k}}^{\ j+1} + B_{y_{i,k}}^{\ j+1}\right) T_{i,k+1}^{j+1}\end{array}\right] \\ &\quad = \left[\begin{array}{l}\left(C_{xi,k}^{\ j} - B_{xi,k}^{\ j}\right) T_{i-1,k}^{j} + \left(C_{y_{i,k}}^{\ j} - B_{y_{i,k}}^{\ j}\right) T_{i,k-1}^{j} + \left(1 - 2C_{xi,k}^{\ j} - 2C_{y_{i,k}}^{\ j} - A_{i,k}^{j}\right) T_{i,k}^{j} \\ + \left(C_{xi,k}^{\ j} + B_{xi,k}^{\ j}\right) T_{i+1,k}^{j} + \left(C_{y_{i,k}}^{\ j} + B_{y_{i,k}}^{\ j}\right) T_{i,k+1}^{j} + F_{i,k}^{j} \ + F_{i,k}^{j+1}\end{array}\right]\end{aligned} \tag{II.12}$$

If we define $J_{diag} = (1 + 2C_x + 2C_y + A)$, $J_{hi,x} = -(C_x + B_x)$, $J_{hi,y} = -(C_y + B_y)$, $J_{lo,x} = -(C_x - B_x)$, $J_{lo,y} = -(C_y - B_y)$, and $R_{diag} = (1 - 2C_x - 2C_y - A)$ then we can rewrite equation (II.12) as

$$\begin{aligned}&\left[J_{lo,x_{i,k}}^{\ j+1} T_{i-1,k}^{j+1} + J_{lo,y_{i,k}}^{\ j+1} T_{i,k-1}^{j+1} + J_{diag_{i,k}}^{\ j+1} T_{i,k}^{j+1} + J_{hi,x_{i,k}}^{\ j+1} T_{i+1,k}^{j+1} + J_{hi,y_{i,k}}^{\ j+1} T_{i,k+1}^{j+1}\right] \\ &\quad = \left[-J_{lo,x_{i,k}}^{\ j} T_{i-1,k}^{j} - J_{lo,y_{i,k}}^{\ j} T_{i,k-1}^{j} + R_{diag_{i,k}}^{\ j} T_{i,k}^{j} - J_{hi,x_{i,k}}^{\ j} T_{i+1,k}^{j} - J_{hi,y_{i,k}}^{\ j} T_{i,k+1}^{j} + F_{i,k}^{j} \ + F_{i,k}^{j+1}\right]\end{aligned} \tag{II.13}$$

These equations hold for all interior nodes. We will deal with nodes affected by the boundary conditions shortly. When we have Dirichlet boundary conditions, an interior node is any node except those at the boundary. When we have Neumann boundary conditions, we are forced to add an imaginary node on each side of the system. Thus an interior node is any node

except those two imaginary nodes (but including the nodes that would be the boundary if we had Dirichlet boundary conditions).

This format is linear in unknowns, $\left\{ T^{j+1} \right\}$. We should take careful notice of this equation. (1) All our unknown temperatures (the temperatures at time j+1 are on the left hand side of the equation). (2) Moreover, they appear in a linear fashion on the LHS. (3) All the variables on the RHS are known quantities. Clearly this is going to give us a system of linear, algebraic equations. We solve this system of equations using the rules of linear algebra. In fact, we can write the above equation as:

$$\underline{\underline{J}}\,\underline{T}^{j+1} = \underline{R} \tag{II.14}$$

This is a system of equations of the standard form:

$$\underline{\underline{A}}\,\underline{x} = \underline{b} \tag{II.15}$$

with a solution

$$\underline{T}^{j+1} = \underline{\underline{J}}^{-1}\underline{R} \tag{II.16}$$

so long as the determinant of the J matrix is non-zero. We will call the matrix on the left-hand side of equation (II.14) the Jacobian and we will call the vector on the right-hand side of equation (II.14) the residual.

Size of the matrix:

If there are m_x spatial intervals in the x-direction, there are m_x+1 spatial nodes in the x-direction, numbered by the variable i from 0 to m_x. If there are m_y spatial intervals in the y-direction, there are m_y+1 spatial nodes in the y-direction, numbered by the variable k from 0 to m_y.

For 2 Dirichlet boundary conditions in each spatial dimension, if there are m_x spatial nodes, then there are m_x-1 interior nodes in the x-direction. If there are m_y spatial nodes, then there are m_y-1 interior nodes in the y-direction. Thus there are $m_u=(m_y-1)(m_x-1)$ unknown temperatures. The $\underline{\underline{J}}$ matrix is a matrix of dimension m_u by m_u, with an index κ bounded by $1 \le \kappa \le m_u$ corresponding to all of the spatial nodes.

The number of unknowns in the Jacobian for other boundary conditions is left as an exercise to the reader. See, the derivation of the same for the one-dimensional case, for a parallel example.

The right hands side of the equation (II.14) is the residual. The remaining problem lies in mapping the 2-D spatial coordinates of the nodes to the single dimensionality of the unknown vector and residual vector. Pay attention this is tricky.

When $m_x < m_y$, number the nodes along x-axis first, as shown in Figure Three. When $m_x > m_y$, number the nodes along y-axis first, as shown in Figure Four. This gives the banded matrix with the smallest bandwidth.

The book-keeping for this mapping is accomplished by the use of an index vector. We need to know how to do two things.

First, we need to know, given the spatial position (x and y positions or i and k coordinates) of the node, how do we get the matrix index.

This transformation is given by:

$$\kappa = (k-1)m_x + i \qquad \text{for } m_x < m_y, \text{ for i = 0 to } m_x, \text{ k = 0 to } m_y \qquad \text{(II.17)}$$

$$\kappa = (i-1)m_y + k \qquad \text{for } m_x > m_y, \text{ for i = 0 to } m_x, \text{ k = 0 to } m_y \qquad \text{(II.18)}$$

However, if you have numbered your nodes from i = 1 to $m_x + 1$ and k = 1 to $m_y + 1$ (which you would do if you were using MATLAB, which can only start vectors with index 1) rather than from i = 0 to m_x and k = 0 to m_y (which is allowed in the more general FORTRAN), then you have

$$\kappa = (k-2)m_x + i - 1 \qquad \text{for } m_x < m_y, \text{ for i = 1 to } m_x + 1, \text{ k = 1 to } m_y + 1 \qquad \text{(II.19)}$$

$$\kappa = (i-2)m_y + k - 1 \qquad \text{for } m_x < m_y, \text{ for i = 1 to } m_x + 1, \text{ k = 1 to } m_y + 1 \qquad \text{(II.20)}$$

The second transformation we need to accomplish is, given the matrix index, κ, how do we obtain the i and k coordinates.

$$\begin{aligned} i &= \left[\frac{\kappa-1}{m_x-1} - \text{int}\left(\frac{\kappa-1}{m_x-1}\right)\right]\cdot(m_x-1)+1 \\ k &= \text{int}\left(\frac{\kappa-1}{m_x-1}\right)+1 \end{aligned} \qquad \text{for} \begin{cases} m_x < m_y \\ i = 0 \text{ to } m_x \\ k = 0 \text{ to } m_y \end{cases} \qquad \text{(II.21)}$$

$$\begin{aligned} i &= \text{int}\left(\frac{\kappa-1}{m_y-1}\right)+1 \\ k &= \left[\frac{\kappa-1}{m_y-1} - \text{int}\left(\frac{\kappa-1}{m_y-1}\right)\right]\cdot(m_y-1)+1 \end{aligned} \qquad \text{for} \begin{cases} m_x > m_y \\ i = 0 \text{ to } m_x \\ k = 0 \text{ to } m_y \end{cases} \qquad \text{(II.22)}$$

However, if you have numbered your nodes from i = 1 to $m_x + 1$ and k = 1 to $m_y + 1$ rather than from i = 0 to m_x and k = 0 to m_y, then you have

$$\begin{aligned} i &= \left[\frac{\kappa-1}{m_x-1} - \text{int}\left(\frac{\kappa-1}{m_x-1}\right)\right]\cdot(m_x-1)+2 \\ k &= \text{int}\left(\frac{\kappa-1}{m_x-1}\right)+2 \end{aligned} \qquad \text{for} \begin{cases} m_x < m_y \\ i = 1 \text{ to } m_x+1 \\ k = 1 \text{ to } m_y+1 \end{cases} \tag{II.23}$$

$$\begin{aligned} i &= \text{int}\left(\frac{\kappa-1}{m_y-1}\right)+2 \\ k &= \left[\frac{\kappa-1}{m_y-1} - \text{int}\left(\frac{\kappa-1}{m_y-1}\right)\right]\cdot(m_y-1)+2 \end{aligned} \qquad \text{for} \begin{cases} m_x > m_y \\ i = 1 \text{ to } m_x+1 \\ k = 1 \text{ to } m_y+1 \end{cases} \tag{II.24}$$

These rules allow us to quickly perform the transformations without having to store the transformation matrix, which can be a memory limitation, if the problem gets large.

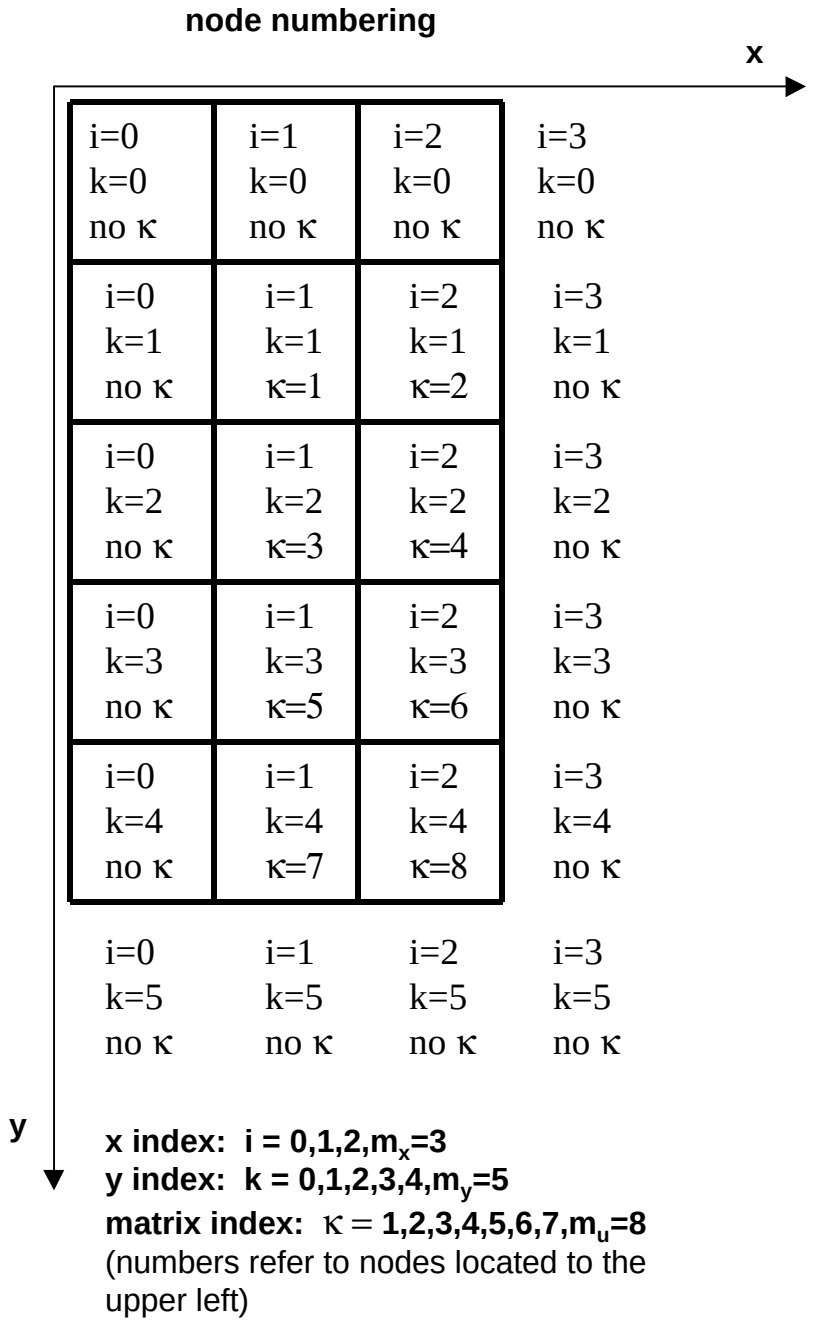
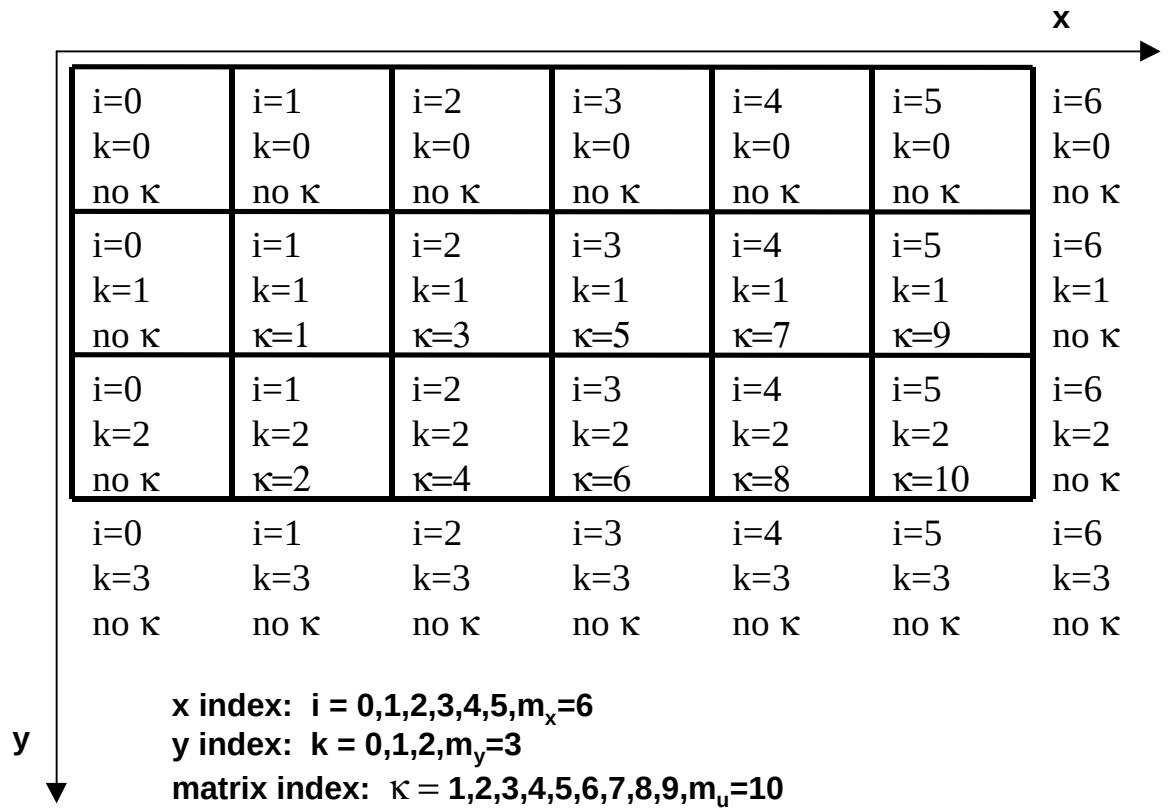


Figure Three. Node numbering example. $m_x < m_y$ Number along the x-axis first.
$m_u = (m_y - 1)(m_x - 1) = (5-1)(3-1) = 4 \cdot 2 = 8$

node numbering



x index: i = 0,1,2,3,4,5,m_x=6
y index: k = 0,1,2,m_y=3
matrix index: κ = 1,2,3,4,5,6,7,8,9,m_u=10

(numbers refer to nodes located to the upper left)

Figure Four. Node numbering example. $m_x > m_y$ Number along the y-axis first.
$m_u = (m_y - 1)(m_x - 1) = (3-1)(6-1) = 2 \cdot 5 = 10$

B. Dirichlet boundary conditions.

We will consider our Dirichlet Boundary conditions in their most general form as

$$T = h \tag{II.25}$$

where h is a known functions of t, x in one spatial dimension and t, x, y, z in three dimensions. In terms of an unknown temperature at a discretized node, we have:

$$T_{i,k}^{j+1} = h_{i,k}^{j+1} = h\left(x_i, y_k, t_{j+1}\right) \tag{II.26}$$

This equation will be used at the boundary nodes. This is in the same linear form as the finite-difference equations for the P.D.E., which is what we need since, ultimately, we need to solve them simultaneously.

III. JACOBIANS AND RESIDUALS.

Let's completely specify the initial Jacobian and Residual for a small problem with Dirichlet boundary conditions, as shown in Figure Five.

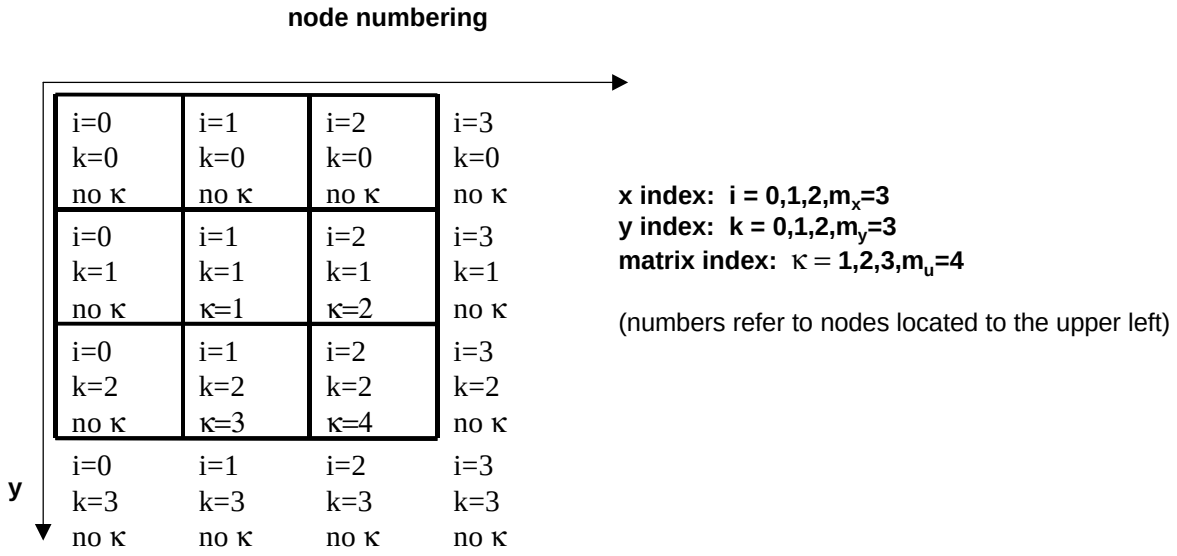


Figure Five. Example Problem. $m_x = m_y$ Number along the y-axis first.

$$m_u = (m_y - 1)(m_x - 1) = (3-1)(3-1) = 2 \cdot 2 = 4$$

To create the Jacobian, we explicity fill in the matrix as dictated by equation (II.13). Some of the terms on the left hand-side are not unknowns because they are defined by the Dirichlet boundary conditions. These terms get moved to the left-hand-side of equation (II.13) (which means they get moved into the residual vector).

$$\underline{\underline{J}} = \begin{bmatrix} J^{j+1}_{diag,1,1} & J^{j+1}_{hi,x,1,1} & J^{j+1}_{hi,y,1,1} & 0 \\ J^{j+1}_{lo,x,2,1} & J^{j+1}_{diag,2,1} & 0 & J^{j+1}_{hi,y,2,1} \\ J^{j+1}_{lo,y,1,2} & 0 & J^{j+1}_{diag,1,2} & J^{j+1}_{hi,x,1,2} \\ 0 & J^{j+1}_{lo,y,2,2} & J^{j+1}_{lo,x,2,2} & J^{j+1}_{diag,2,2} \end{bmatrix}$$

The residual is now everything else in equation (II.13) that didn't make it into the Jacobian.

$$\underline{R} = \begin{bmatrix} -J_{lo,x_{1,1}}^{j} T_{0,1}^{j} - J_{lo,y_{1,1}}^{j} T_{1,0}^{j} + R_{diag_{1,1}}^{j} T_{1,1}^{j} - J_{hi,x_{1,1}}^{j} T_{2,1}^{j} - J_{hi,y_{1,1}}^{j} T_{1,2}^{j} + F_{1,1}^{j} + F_{1,1}^{j+1} - J_{lo,x_{1,1}}^{j+1} T_{0,1}^{j+1} - J_{lo,y_{1,1}}^{j+1} T_{1,0}^{j+1} \\ -J_{lo,x_{2,1}}^{j} T_{1,1}^{j} - J_{lo,y_{2,1}}^{j} T_{2,0}^{j} + R_{diag_{2,1}}^{j} T_{2,1}^{j} - J_{hi,x_{2,1}}^{j} T_{3,1}^{j} - J_{hi,y_{2,1}}^{j} T_{2,2}^{j} + F_{2,1}^{j} + F_{2,1}^{j+1} - J_{lo,y_{2,1}}^{j+1} T_{2,0}^{j+1} - J_{hi,x_{2,1}}^{j+1} T_{3,1}^{j+1} \\ -J_{lo,x_{1,2}}^{j} T_{0,2}^{j} - J_{lo,y_{1,2}}^{j} T_{1,1}^{j} + R_{diag_{1,2}}^{j} T_{1,2}^{j} - J_{hi,x_{1,2}}^{j} T_{2,2}^{j} - J_{hi,y_{1,2}}^{j} T_{1,3}^{j} + F_{1,2}^{j} + F_{1,2}^{j+1} - J_{lo,x_{1,2}}^{j+1} T_{0,2}^{j+1} - J_{hi,y_{1,2}}^{j+1} T_{1,3}^{j+1} \\ -J_{lo,x_{2,2}}^{j} T_{1,2}^{j} - J_{lo,y_{2,2}}^{j} T_{2,1}^{j} + R_{diag_{2,2}}^{j} T_{2,2}^{j} - J_{hi,x_{2,2}}^{j} T_{3,2}^{j} - J_{hi,y_{2,2}}^{j} T_{2,3}^{j} + F_{2,2}^{j} + F_{2,2}^{j+1} - J_{hi,x_{2,2}}^{j+1} T_{3,2}^{j+1} - J_{hi,y_{2,2}}^{j+1} T_{2,3}^{j+1} \end{bmatrix}$$

which, when we use the Dirichlet boundary conditions becomes,

$$\underline{R} = \begin{bmatrix} -J_{lo,x_{1,1}}^{j} h_{0,1}^{j} - J_{lo,y_{1,1}}^{j} h_{1,0}^{j} + R_{diag_{1,1}}^{j} T_{1,1}^{j} - J_{hi,x_{1,1}}^{j} T_{2,1}^{j} - J_{hi,y_{1,1}}^{j} T_{1,2}^{j} + F_{1,1}^{j} + F_{1,1}^{j+1} - J_{lo,x_{1,1}}^{j+1} h_{0,1}^{j+1} - J_{lo,y_{1,1}}^{j+1} h_{1,0}^{j+1} \\ -J_{lo,x_{2,1}}^{j} T_{1,1}^{j} - J_{lo,y_{2,1}}^{j} h_{2,0}^{j} + R_{diag_{2,1}}^{j} T_{2,1}^{j} - J_{hi,x_{2,1}}^{j} h_{3,1}^{j} - J_{hi,y_{2,1}}^{j} T_{2,2}^{j} + F_{2,1}^{j} + F_{2,1}^{j+1} - J_{lo,y_{2,1}}^{j+1} h_{2,0}^{j+1} - J_{hi,x_{2,1}}^{j+1} h_{3,1}^{j+1} \\ -J_{lo,x_{1,2}}^{j} h_{0,2}^{j} - J_{lo,y_{1,2}}^{j} T_{1,1}^{j} + R_{diag_{1,2}}^{j} T_{1,2}^{j} - J_{hi,x_{1,2}}^{j} T_{2,2}^{j} - J_{hi,y_{1,2}}^{j} h_{1,3}^{j} + F_{1,2}^{j} + F_{1,2}^{j+1} - J_{lo,x_{1,2}}^{j+1} h_{0,2}^{j+1} - J_{hi,y_{1,2}}^{j+1} h_{1,3}^{j+1} \\ -J_{lo,x_{2,2}}^{j} T_{1,2}^{j} - J_{lo,y_{2,2}}^{j} T_{2,1}^{j} + R_{diag_{2,2}}^{j} T_{2,2}^{j} - J_{hi,x_{2,2}}^{j} h_{3,2}^{j} - J_{hi,y_{2,2}}^{j} h_{2,3}^{j} + F_{2,2}^{j} + F_{2,2}^{j+1} - J_{hi,x_{2,2}}^{j+1} h_{3,2}^{j+1} - J_{hi,y_{2,2}}^{j+1} h_{2,3}^{j+1} \end{bmatrix}$$

The residual is a vector of known quantities.

IV. CODIFICATION.

The only remaining trick is to come up with a code that will quickly fill the Jacobian matrix and the residual vector.

One can methodically write

Consider the case where the nodes are numbered from i = 0 to m_x and k = 0 to m_y. Then we can write:

```
% loop over i (only consider unknown nodes)
for i = 1:1:mx-1
        % loop over k (only consider unknown nodes)
        for k = 1:1:my-1
                if (mx < my)
                        kappa_row = f(i,k) from equation (II.17)
                else
                        kappa_row = f(i,k) from equation (II.18)
                end
                % loop over self and 4 adjacent nodes
                for jj = 1:1:5
                        % what you do depends on which node you have got
                        if (jj == 1)
                                % add low-x contribution
                                iu = i - 1
                                ku = k
                                % determine if this node is a variable or a known
                                if (iu > 0 & iu < mx & ku > 0 & ku < my)
                                        % you have got an unknown
                                        if (mx < my)
                                                kappa = f(iu,ku) from equation (II.17)
                                        else
                                                kappa = f(iu,ku) from equation (II.18)
                                        end
                                        J(kappa_row,kappa) = J^{j+1}_{lo,x,i,k}
                                        R(kappa_row) = R(kappa_row) - J^{j}_{lo,x,i,k} T^{j}_{lo,x,iu,ku}
                                else
                                        % you have got a known
                                        R(kappa_row) = R(kappa_row) - J^{j+1}_{lo,x,i,k} T^{j+1}_{lo,x,iu,ku}
                                        R(kappa_row) = R(kappa_row) - J^{j}_{lo,x,i,k} T^{j}_{lo,x,iu,ku}
                                end
                elseif (jj == 2)
                                % add low-y contribution, analogously to low-x
                        elseif (jj == 3)
                                % add high-x contribution, analogously to low-x
                        elseif (jj == 4)
                                % add high-y contribution, analogously to low-x
                        else
                                % add self contribution and F's to residual, analogously to low-x
```

```
                    end
                end     % this ends the jj-loop
            end         % this ends the k-loop
end                     % this ends the i-loop
```

This pseudocode algorithm will work. For the generalized case, it is as efficient a code as we can expect. However, if there are simple relationships between $J^{j+1}_{lo,x,i,k}$ and $J^{j+1}_{lo,y,i,k}$ and the others, more efficient schemes, which take advantage of any symmetries in the Jacobian can be derived.

V. SOLUTION.

With the Jacobian and Residual, we solve for the temperatures at the next time, j+1:

$$\underline{T}^{j+1} = \underline{\underline{J}}^{-1}\underline{R}$$

We can then repeat the calculation of the Jacobian and the residual and solve for the temperatures at time, j+2, and so on.

Since we have the reverse transformation from matrix coordinates to spatial coordinates in equations (II.21) and (II.22), it is no problem to find the temperatures at any given point.

VI. EXTENSION TO NONLINEAR EQUATIONS

The extension of the solution methodology from the 2D linear parabolic PDEs to 2D nonlinear parabolic PDEs is precisely analogous to the extension of the solution methodology from the 1D linear parabolic PDEs to 1D nonlinear parabolic PDEs.

You just write the PDE as ODE, use finite differences for the spatial derivatives and solve using a second-order Runge-Kutta method.