CHAPTER 5.

Multi-Dimensional Parabolic Problems.

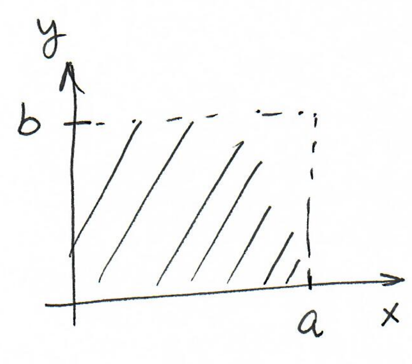
Heat Conduction (3-D)

$$
\begin{cases}
\rho c\, u_t = \nabla\cdot(K\nabla u), & (x,y,z)\in\Omega. \\
u(x,y,z,0) = \phi(x,y,z), & (x,y,z)\in\Omega\cup\partial\Omega. \\
\alpha u + \beta \dfrac{\partial u}{\partial n} = \gamma, & (x,y,z)\in\partial\Omega,\quad t>0.
\end{cases}
$$

We will restrict our attention to 2-D problems on a rectangular domain. $K \equiv \text{const}$.

$$
\begin{cases}
u_t = \sigma(u_{xx}+u_{yy}), & (x,y)\in\Omega \\
u(x,y) = \phi(x,y), & (x,y)\in\Omega\cup\partial\Omega \\
u(x,y,t) = \gamma(x,y,t), & (x,y)\in\partial\Omega,\quad t>0
\end{cases}
$$

$$\Omega = \{(x,y):\ 0\le x<a,\quad 0<y<b\}.$$



Introduce a uniform rectangular grid with space step sizes given by

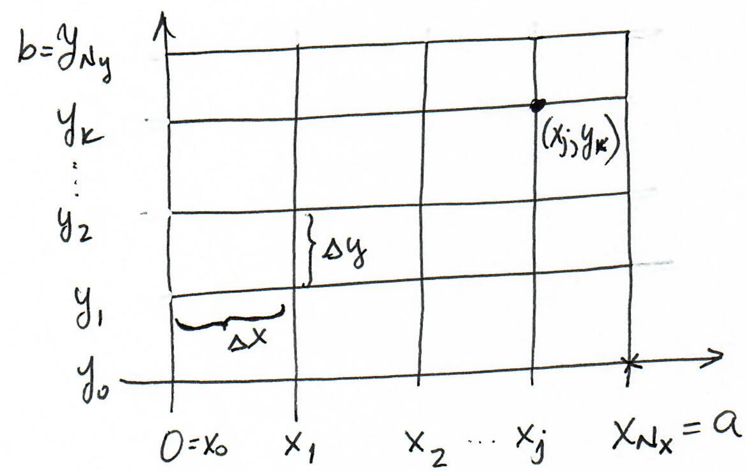
$$\Delta x = \frac{a}{N_x}, \qquad \Delta y = \frac{b}{N_y}.$$

and Δt as a time step size.

Let $U_{j,k}^n \simeq u(x_j, y_k, t_n)$ where

$$x_j = j * \Delta x$$
$$y_k = k * \Delta y$$
$$t_n = n * \Delta t.$$

$$j = 0, .., N_x$$
$$k = 0, .., N_y$$



Explicit scheme FT-CS

$$\frac{U_{j,k}^{n+1} - U_{j,k}^n}{\Delta t} = \sigma \left[\frac{U_{j-1,k}^n - 2U_{j,k}^n + U_{j+1,k}^n}{(\Delta x)^2} + \frac{U_{j,k-1}^n - 2U_{j,k}^n + U_{j,k+1}^n}{(\Delta y)^2} \right] \tag{1}$$

$j = 1, .., N_x - 1$
$k = 1, .., N_y - 1$

Equ. (1) can also be written as

$$U_{j,k}^{n+1} = U_{j,k}^n + r_x \left[U_{j-1,k}^n - 2U_{j,k}^n + U_{j+1,k}^n \right] + r_y \left[U_{j,k-1}^n - 2U_{j,k}^n + U_{j,k+1}^n \right] \tag{2}$$

or $\boxed{U_{j,k}^{n+1} = U_{j,k}^n + r_x \delta_x^2 U_{j,k}^n + r_y \delta_y^2 U_{j,k}^n}$

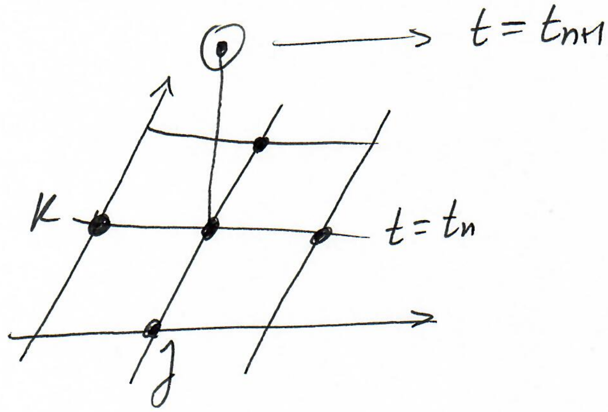
where $r_x = \dfrac{\sigma \Delta t}{\Delta x^2}$, and $r_y = \dfrac{\sigma \Delta t}{\Delta y^2}$

To study stability, it's convenient to write equ. (2) as

$$U_{j,k}^{n+1} = r_y U_{j,k-1}^{n} + r_x U_{j-1,k}^{n} + \underbrace{(1-2r_x-2r_y)}_{=\,(1-2(r_x+r_y))} U_{j,k}^{n} + r_x U_{j+1,k}^{n} + r_y U_{j,k+1}^{n} \quad (3)$$

$\Rightarrow$ $\vec{U}^{n+1} = [I - C]\vec{U}^{n}$, C is defined in page 8.

<u>Computational stencil</u>



<u>Stability</u>

If $r_x + r_y \le 1/2 \Rightarrow 1 - 2(r_x + r_y) \ge 0$

and $|1 - 2(r_x + r_y)| + 2r_x + 2r_y = 1$

Max. norm principle applies

In particular, if $\Delta x = \Delta y \Rightarrow r_x + r_y \le 1/2 \Leftrightarrow 2r_x \le 1/2$

$\Rightarrow \boxed{r_x \le 1/4}$ More restrictive than in 1-D.

Two-Dimensional Crank-Nicholson. Derivation. Matrix Representation

Approximation of Heat Equation at $(x_j, y_k, t_{n+1/2})$

$$(u_t)_{jk}^{n+1/2} = \sigma\left[(u_{xx})_{jk}^{n+1/2} + (u_{yy})_{jk}^{n+1/2} \right] \qquad (1)$$

CT-CS. Time step-size $\Delta t/2$.

$$\frac{v_{jk}^{n+1} - v_{jk}^{n}}{2(\Delta t/2)} = \sigma\left[\frac{v_{j-1,k}^{n+1/2} - 2v_{jk}^{n+1/2} + v_{j+1,k}^{n+1/2}}{(\Delta x)^2} \right] + \sigma\left[\frac{v_{j,k-1}^{n+1/2} - 2v_{j,k}^{n+1/2} + v_{j,k+1}^{n+1/2}}{(\Delta y)^2} \right]$$

or

$$\frac{v_{jk}^{n+1} - v_{jk}^{n}}{\Delta t} = \sigma\left[\frac{\delta_x^2}{(\Delta x)^2} v_{jk}^{n+1/2} + \frac{\delta_y^2}{(\Delta y)^2} v_{jk}^{n+1/2} \right] \qquad (2)$$

Approximate $v_{jk}^{n+1/2} = \dfrac{v_{jk}^{n+1} + v_{jk}^{n}}{2}$

Then,

$$\frac{v_{jk}^{n+1} - v_{jk}^{n}}{\Delta t} = \frac{\sigma}{2}\left[\frac{\delta_x^2}{\Delta x^2} v_{jk}^{n+1} + \frac{\delta_y^2}{(\Delta y)^2} v_{jk}^{n+1} \right] + \frac{\sigma}{2}\left[\frac{\delta_x^2}{\Delta x^2} v_{jk}^{n} + \frac{\delta_x^2}{(\Delta y)^2} v_{jk}^{n} \right]. \qquad (3)$$

or

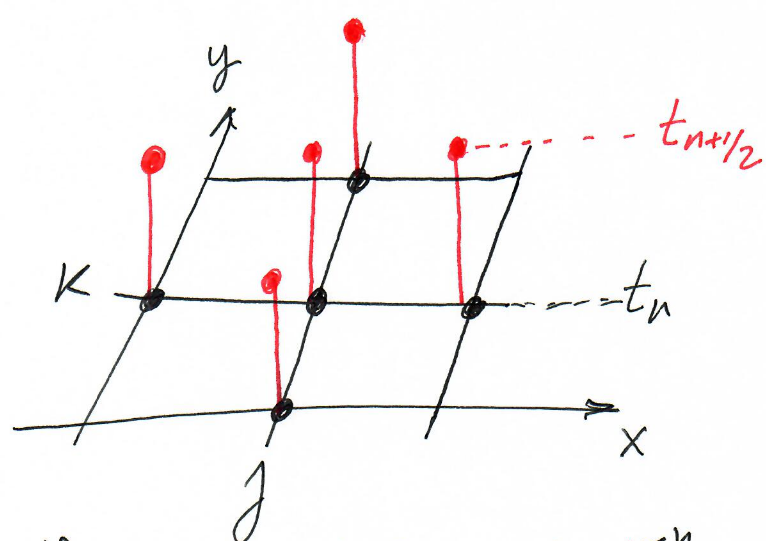
$$U_{jk}^{n+1} - \frac{\sigma \Delta t}{2\Delta x^2} \delta_x^2 U_{jk}^{n+1} - \frac{\sigma \Delta t}{2\Delta y^2} \delta_y^2 U_{jk}^{n+1}$$

$$= U_{jk}^{n} + \frac{\sigma \Delta t}{2\Delta x^2} \delta_x^2 U_{jk}^{n} + \frac{\sigma \Delta t}{2\Delta y^2} \delta_y^2 U_{jk}^{n}$$

or

$$\boxed{U_{jk}^{n+1} - \frac{1}{2} r_x \delta_x^2 U_{jk}^{n+1} - \frac{1}{2} r_y \delta_y^2 U_{jk}^{n+1} = U_{jk}^{n} + \frac{1}{2} r_x \delta_x^2 U_{jk}^{n} + \frac{1}{2} r_y \delta_y^2 U_{jk}^{n}} \tag{4}$$

Computational Stencil.



10-point stencil.

It means the value at (x_j, y_k, t_n), U_{jk}^{n}, depends on values at 10 other neighbors nodes including itself.

Matrix Representation

Equ. (4) can also be written as

$$U_{j,k}^{n+1} - \frac{r_x}{2}\delta_x^2 U_{j,k}^{n+1} - \frac{r_y}{2}\delta_y^2 U_{j,k}^{n+1} = \left[1 + \left(\frac{r_x\delta_x^2 + r_y\delta_y^2}{2}\right)\right] U_{jk}^{n} \equiv \text{rhs}$$

or expanding $\delta_x^2 U_{jk}^{n+1}$, $\delta_y^2 U_{jk}^{n+1}$ in the lhs.

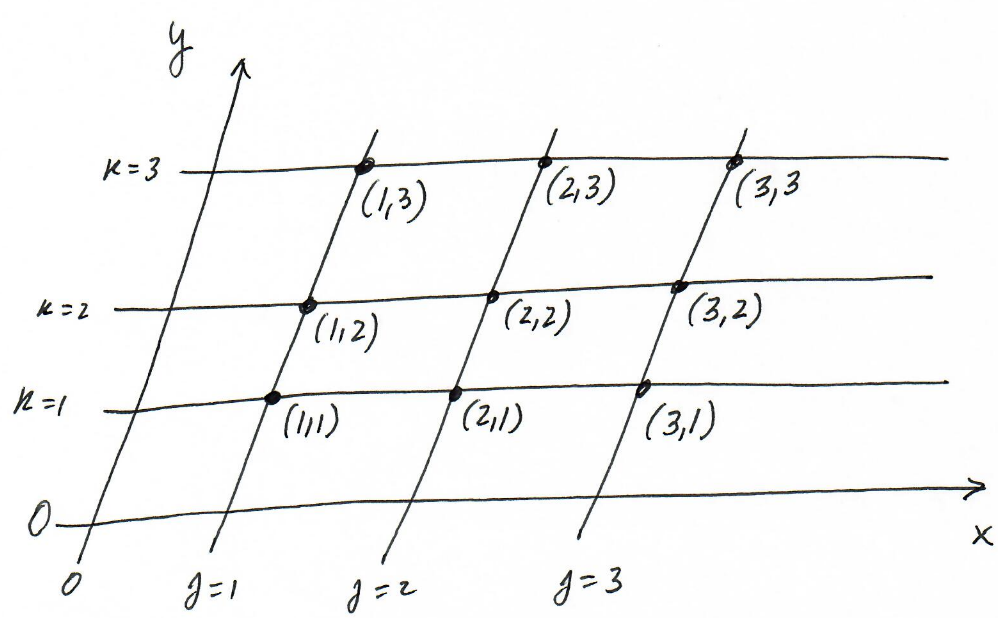
$$U_{j,k}^{n+1} + \frac{1}{2}\left[-r_x U_{j-1,k}^{n+1} + 2r_x U_{j,k}^{n+1} - r_x U_{j+1,k}^{n+1}\right] +$$

$$+ \frac{1}{2}\left[-r_y U_{j,k-1}^{n+1} + 2r_y U_{j,k}^{n+1} - r_y U_{j,k+1}^{n+1}\right] = \text{rhs}$$

or

$$U_{j,k}^{n+1} + \frac{1}{2}\left[-r_y U_{j,k-1}^{n+1} - r_x \boxed{U_{j-1,k}^{n+1}} + 2(r_x+r_y)\boxed{U_{jk}^{n+1}} - r_x \boxed{U_{j+1,k}^{n+1}} - r_y U_{j,k+1}^{n+1}\right] = \text{rhs}.$$

To construct the matrix consider



If BC's are homogeneous (DIRICHLET).

$\boxed{k=1}$

$$j=1:\ U_{11}^{n+1} + \frac{1}{2}\left[2(r_x+r_y)U_{11}^{n+1} - r_x U_{2,1}^{n+1} - r_y U_{1,2}^{n+1}\right] = rhs.$$

$$j=2:\ U_{2,1}^{n+1} + \frac{1}{2}\left[-r_x U_{11}^{n+1} + 2(r_x+r_y)U_{21}^{n+1} - r_x U_{3,1}^{n+1} - r_y U_{22}^{n+1}\right] = rhs.$$

$\vdots$

$$j=N_x-1:\ U_{N_x-1,1}^{n+1} + \frac{1}{2}\left[-r_x U_{N_x-2,1}^{n+1} + 2(r_x+r_y)U_{N_x-1,1}^{n+1} - r_y U_{N_x-1,2}^{n+1}\right] = rhs$$

$\boxed{k=2}$

$$j=1:\ U_{1,2}^{n+1} + \frac{1}{2}\left[-r_y U_{11}^{n+1} + 2(r_x+r_y)U_{1,2}^{n+1} - r_x U_{2,2}^{n+1} - r_y U_{1,3}^{n+1}\right] = rhs$$

$$j=2:\ U_{2,2}^{n+1} + \frac{1}{2}\left[-r_y U_{2,1}^{n+1} - r_x U_{1,2}^{n+1} + 2(r_x+r_y)U_{2,2}^{n+1} - r_x U_{3,2}^{n+1} - r_y U_{2,3}^{n+1}\right] = rhs$$

$\vdots$

$$j=N_x-1:\ U_{N_x-1,2}^{n+1} + \frac{1}{2}\left[-r_y U_{N_x-1,1}^{n+1} - r_x U_{N_x-2,2}^{n+1} + 2(r_x+r_y)U_{N_x-1,2}^{n+1} - r_y U_{N_x-1,3}^{n+1}\right] = rhs$$

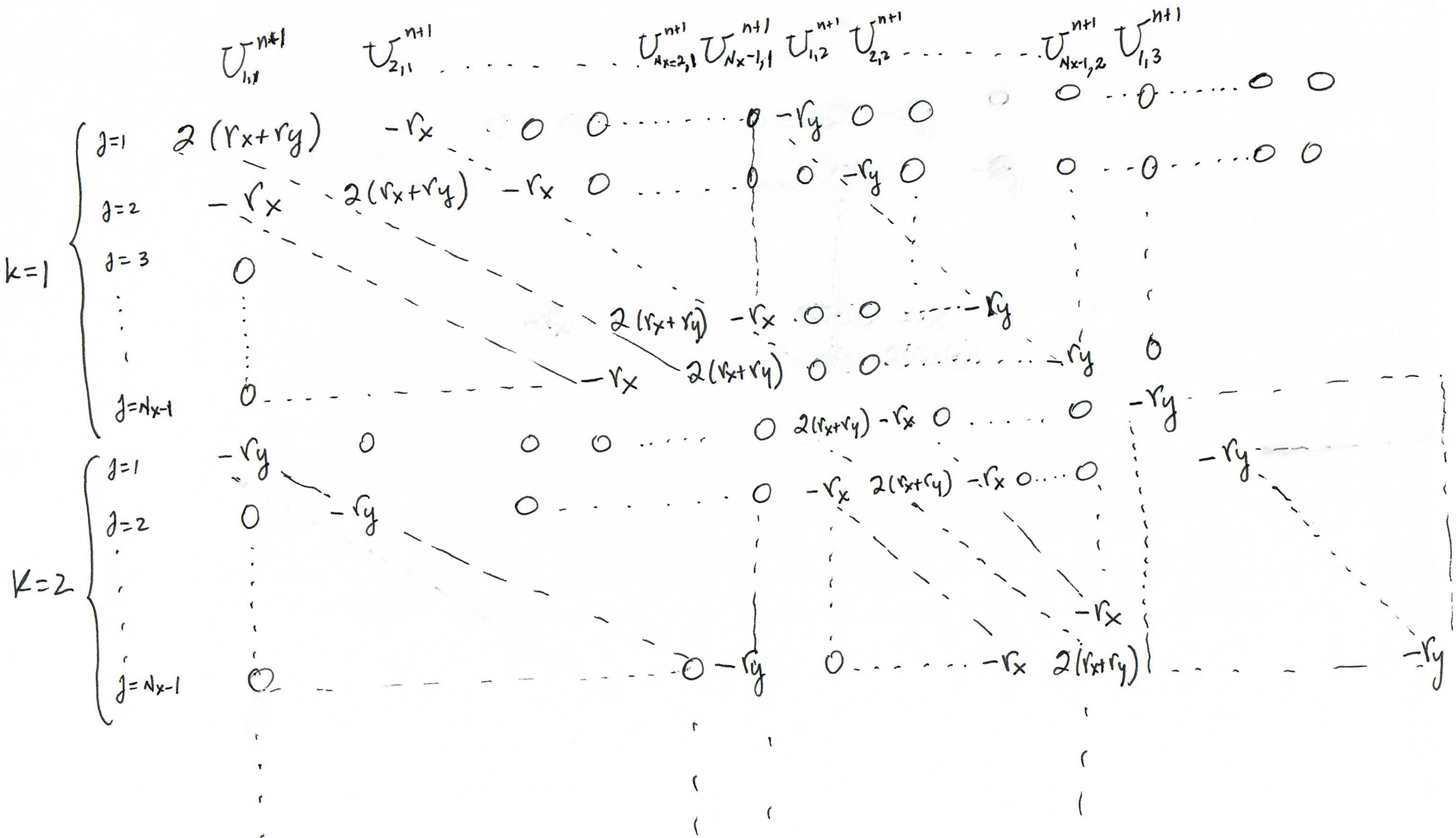
$\vdots$

$k=N_y-1$.

It's convenient to define the unknown vector $\vec{U}^n$ by rows.

$$\vec{U}^n = \begin{bmatrix} U_{11}^n \\ U_{21}^n \\ \vdots \\ U_{N_x-1,1}^n \\ U_{1,2}^n \\ U_{22}^n \\ \vdots \\ U_{N_x-1,2}^n \\ \vdots \end{bmatrix}\begin{array}{l} \left.\begin{matrix} \\ \\ \\ \\ \end{matrix}\right\} \text{1st row in x-direction} \\ \left.\begin{matrix} \\ \\ \\ \\ \end{matrix}\right\} \text{2nd row in x-direction} \\ \\ \end{array} \qquad \begin{bmatrix} \vdots \\ U_{1,N_y-1}^n \\ U_{2,N_y-1}^n \\ \vdots \\ U_{N_x-1,N_y-1}^n \end{bmatrix}\left.\begin{matrix} \\ \\ \\ \\ \\ \end{matrix}\right\} \text{Last row in x-direction.}$$

MATRIX REPRESENTATION. (only the expressions in brackets)



Size of matrix

$$[(N_x-1)(N_y-1)] \times [(N_x-1)(N_y-1)]$$

The matrix Representation of the previous page Can be written in compact form as

$$\boxed{\left[I+\frac{1}{2}G\right]\vec{U}^{n+1}=\left[I-\frac{1}{2}G\right]\vec{U}^{n}} \qquad (8.1)$$

where

$$G \equiv \begin{bmatrix} D_x & D_y & 0 & 0 & \cdots & 0 \\ D_y & D_x & D_y & 0 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & & \vdots \\ & & & & & 0 \\ & & & D_y & D_x & D_y \\ 0 & \cdots & & 0 & D_y & D_x \end{bmatrix}_{[(N_x-1)(N_y-1)]\times[(N_x-1)(N_y-1)]}$$

$$D_x \equiv \begin{bmatrix} 2(r_x+r_y) & -r_x & 0 & \cdots & 0 \\ -r_x & 2(r_x+r_y) & -r_x & \cdots & 0 \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & & -r_x & 2(r_x+r_y) & -r_x \\ 0 & \cdots & 0 & -r_x & 2(r_x+r_y) \end{bmatrix}_{(N_x-1)\times(N_x-1)}$$

$$D_y \equiv \begin{pmatrix} -r_y & 0 & \cdots & 0 \\ 0 & -r_y & \cdots & 0 \\ \vdots & & \ddots & 0 \\ 0 & 0 & \cdots & -r_y \end{pmatrix}$$

The linear system (8.1) can be solved by an extension of the tridiagonal algorithm to block systems.

It requires $\frac{5}{3} N_y (N_x)^3$ multiplications per time step.

Too expensive!

A tridiagonal algorithm will require only $\mathcal{O}(N_y \times N_x)$ operations.

Which is of the same order of the number of unknowns.

where

$$a = -\frac{\alpha\,\Delta t}{2(\Delta y)^2} = -\frac{1}{2}\,r_y$$

$$b = -\frac{\alpha\,\Delta t}{2(\Delta x)^2} = -\frac{1}{2}\,r_x$$

$$c = 1 + r_x + r_y$$

$$d_{i,j}^n = u_{i,j}^n + \frac{\alpha\,\Delta t}{2}\,(\hat{\delta}_x^2 + \hat{\delta}_y^2)u_{i,j}^n$$

If we apply Eq. (4-101) to the two-dimensional (6 X 6) computational mesh shown in Fig. 4-15, the following system of 16 linear algebraic equations must be solved at each $(n+1)$ time level.

$$\begin{bmatrix}
c & b & 0 & 0 & a & 0 & & & & & & & & & & 0 \\
b & c & b & & & a & & & & & & & & & & \\
0 & b & c & b & & & a & & & & & & & & & \\
0 & & b & c & 0 & & & a & & & & & & & & \\
a & & & 0 & c & b & & & a & & & & & & & \\
0 & a & & & b & c & b & & & a & & & & & & \\
& & a & & & b & c & b & & & a & & & & & \\
& & & a & & & b & c & 0 & & & a & & & & \\
& & & & a & & & 0 & c & b & & & a & & & \\
& & & & & a & & & b & c & b & & & a & & \\
& & & & & & a & & & b & c & b & & & a & 0 \\
& & & & & & & a & & & b & c & 0 & & & a \\
& & & & & & & & a & & & 0 & c & b & & 0 \\
& & & & & & & & & a & & & b & c & b & 0 \\
& & & & & & & & & & a & & & b & c & b \\
0 & & & & & & & & & & 0 & a & 0 & 0 & b & c
\end{bmatrix}
\begin{bmatrix}
u_{2,2}^{n+1} \\ u_{3,2}^{n+1} \\ u_{4,2}^{n+1} \\ u_{5,2}^{n+1} \\ u_{2,3}^{n+1} \\ u_{3,3}^{n+1} \\ u_{4,3}^{n+1} \\ u_{5,3}^{n+1} \\ u_{2,4}^{n+1} \\ u_{3,4}^{n+1} \\ u_{4,4}^{n+1} \\ u_{5,4}^{n+1} \\ u_{2,5}^{n+1} \\ u_{3,5}^{n+1} \\ u_{4,5}^{n+1} \\ u_{5,5}^{n+1}
\end{bmatrix}
=
\begin{bmatrix}
d_{2,2}''' \\ d_{3,2}' \\ d_{4,2}' \\ d_{5,2}''' \\ d_{2,3}'' \\ d_{3,3} \\ d_{4,3} \\ d_{5,3}'' \\ d_{2,4}'' \\ d_{3,4} \\ d_{4,4} \\ d_{5,4}'' \\ d_{2,5}''' \\ d_{3,5}' \\ d_{4,5}' \\ d_{5,5}'''
\end{bmatrix}
\qquad (4\text{-}102)$$

where $d' = d - au_0$

$d'' = d - bu_0$

$d''' = d - (a+b)u_0$

A system of equations, like Eq. (4-102), requires substantially more computer time to solve than does a tridiagonal which we substitute into (15.10) to find the amplification factor ρ. In this calculation, as well as in many other calculations in the remainder of this Lecture, we will use the following formulae:

$$\delta_x^2 \left[e^{i\beta mh}\, e^{i\gamma lh}\right] = -4\sin^2\left(\frac{\beta h}{2}\right) \left[e^{i\beta mh}\, e^{i\gamma lh}\right], \tag{15.12}$$

$$\delta_y^2 \left[e^{i\beta mh}\, e^{i\gamma lh}\right] = -4\sin^2\left(\frac{\gamma h}{2}\right) \left[e^{i\beta mh}\, e^{i\gamma lh}\right]. \tag{15.13}$$

(You will be asked to confirm the validity of these formulae in a homework problem.) Substituting (15.11) into (15.10) and using (15.12) and (15.13), one finds

$$\rho = 1 - 4r\left(\sin^2\frac{\beta h}{2} + \sin^2\frac{\gamma h}{2}\right). \tag{15.14}$$

The harmonics most prone to instability are, as for the one-dimensional Heat equation, those with the highest spatial frequency, and for which

$$\sin^2\frac{\beta h}{2} = \sin^2\frac{\gamma h}{2} = 1\,.$$

For these harmonics, the stability condition $|\rho| \le 1$ implies

$$r \le \frac{1}{4} \qquad \text{or, equivalently,} \qquad \kappa \le \frac{h^2}{4}. \tag{15.15}$$

Thus, in order to ensure the stability of the simple explicit scheme (15.10), one has to impose a restriction on the time step κ that is twice as strong as the analogous restriction in the case of the one-dimensional Heat equation. Therefore, the simple explicit scheme is computationally inefficient, and our next step is, of course, to look for a computationally efficient scheme. As the first candidate for that position, we will analyze the Crank-Nicolson scheme.

15.3 Naive generalization of Crank-Nicolson scheme for the 2D Heat equation

Our main finding in this subsection will be that a naive generalization of the CN method (13.6) is also computationally inefficient. The underlying analysis will allow us to formulate specific properties that a computationally efficient scheme must possess.

The naive generalization to two dimensions of the CN scheme, (13.5) or (13.6), is:

$$U_{ml}^{n+1} = U_{ml}^n + \frac{r}{2}\left(\delta_x^2 + \delta_y^2\right)\left(U_{ml}^n + U_{ml}^{n+1}\right), \tag{15.16}$$

or, equivalently,

$$\left(1 - \frac{r}{2}\delta_x^2 - \frac{r}{2}\delta_y^2\right) U_{ml}^{n+1} = \left(1 + \frac{r}{2}\delta_x^2 + \frac{r}{2}\delta_y^2\right) U_{ml}^n\,. \tag{15.17}$$

Following the lines of Lecture 13, one can show that the accuracy of this scheme is $O(\kappa^2 + h^2)$. Also, the von Neumann analysis yields the following expression for the error amplification factor:

$$\rho = \frac{1 - 2r\left(\sin^2\frac{\beta h}{2} + \sin^2\frac{\gamma h}{2}\right)}{1 + 2r\left(\sin^2\frac{\beta h}{2} + \sin^2\frac{\gamma h}{2}\right)}, \tag{15.18}$$

We will now demonstrate that scheme (15.16) / (15.17) is computationally inefficient. To that end, we need to exhibit the explicit matrix form of that scheme. We begin by rewriting (15.16) in the form[26]:

$$\begin{aligned}(1+2r)U^{n+1}_{m,l} - \frac{r}{2}\left(U^{n+1}_{m+1,l}+U^{n+1}_{m-1,l}\right) - \frac{r}{2}\left(U^{n+1}_{m,l+1}+U^{n+1}_{m,l-1}\right)\\ = (1-2r)U^{n}_{m,l} + \frac{r}{2}\left(U^{n}_{m+1,l}+U^{n}_{m-1,l}\right) + \frac{r}{2}\left(U^{n}_{m,l+1}+U^{n}_{m,l-1}\right).\end{aligned} \tag{15.19}$$

To write down Eqs. (15.19) for all m and l in a compact form, we will need the following notations:

$$A = \begin{pmatrix} 2r & -r/2 & 0 & \cdot & \cdot & 0 \\ -r/2 & 2r & -r/2 & 0 & \cdot & 0 \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ 0 & \cdot & 0 & -r/2 & 2r & -r/2 \\ 0 & \cdot & \cdot & 0 & -r/2 & 2r \end{pmatrix}, \qquad \vec{\mathbf{U}}_{;l} = \begin{pmatrix} U_{1,l} \\ U_{2,l} \\ \cdot \\ U_{M-2,l} \\ U_{M-1,l} \end{pmatrix}, \tag{15.20}$$

and

$$\vec{\mathbf{B}}_k = \begin{pmatrix} (g_k)_1 \\ (g_k)_2 \\ \cdot \\ \cdot \\ (g_k)_{M-1} \end{pmatrix}, \quad \text{for } k=2,3; \qquad \vec{\mathbf{b}}^n_l = \begin{pmatrix} (g_0)^n_l + (g_0)^{n+1}_l \\ 0 \\ \cdot \\ 0 \\ (g_1)^n_l + (g_1)^{n+1}_l \end{pmatrix}. \tag{15.21}$$

Using these notations, one can recast Eq. (15.19) in a matrix form. Namely, for $l = 2, \ldots, L-2$ (i.e. for layers with constant y and which are *not* adjacent to the boundaries), Eq. (15.19) becomes:

$$(I+A)\vec{\mathbf{U}}^{n+1}_{;l} - \frac{r}{2}I\,\vec{\mathbf{U}}^{n+1}_{;l+1} - \frac{r}{2}I\,\vec{\mathbf{U}}^{n+1}_{;l-1} = (I-A)\vec{\mathbf{U}}^{n}_{;l} + \frac{r}{2}I\,\vec{\mathbf{U}}^{n}_{;l+1} + \frac{r}{2}I\,\vec{\mathbf{U}}^{n}_{;l-1} + \frac{r}{2}\vec{\mathbf{b}}^n_l, \tag{15.22}$$

where I is the $(M-1)\times(M-1)$ identity matrix. Note that Eq. (15.22) is analogous to Eq. (13.9), although the meanings of notation A is *different* in these two equations. Continuing, for the layer with $l=1$ one obtains:

$$(I+A)\vec{\mathbf{U}}^{n+1}_{;l} - \frac{r}{2}I\,\vec{\mathbf{U}}^{n+1}_{;l+1} - \frac{r}{2}\vec{\mathbf{B}}^{n+1}_2 = (I-A)\vec{\mathbf{U}}^{n}_{;l} + \frac{r}{2}I\,\vec{\mathbf{U}}^{n}_{;l+1} + \frac{r}{2}\vec{\mathbf{B}}^n_2 + \frac{r}{2}\vec{\mathbf{b}}^n_l. \tag{15.23}$$

The equation for $l = L-1$ has a similar form. Combining now all these equations into one, we obtain:

$$(\mathcal{I}+\mathcal{A})\vec{\mathbf{U}}^{n+1} = (\mathcal{I}-\mathcal{A})\vec{\mathbf{U}}^{n} + \mathcal{B}^n, \tag{15.24}$$

where $\vec{\mathbf{U}}$ has been defined in (15.8), $\mathcal{I}$ is the $[(M-1)(L-1)]\times[(M-1)(L-1)]$ identity matrix, and

$$\mathcal{A} = \begin{pmatrix} A & -\frac{r}{2}I & O & \cdot & \cdot & O \\ -\frac{r}{2}I & A & -\frac{r}{2}I & O & \cdot & O \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ O & \cdot & O & -\frac{r}{2}I & A & -\frac{r}{2}I \\ O & \cdot & \cdot & O & -\frac{r}{2}I & A \end{pmatrix}, \qquad \mathcal{B}^n = \frac{r}{2}\begin{pmatrix} \vec{\mathbf{B}}^n_2 + \vec{\mathbf{B}}^{n+1}_2 + \vec{\mathbf{b}}^n_1 \\ \vec{\mathbf{b}}^n_2 \\ \cdot \\ \vec{\mathbf{b}}^n_{L-2} \\ \vec{\mathbf{B}}^n_3 + \vec{\mathbf{B}}^{n+1}_3 + \vec{\mathbf{b}}^n_{L-1} \end{pmatrix}. \tag{15.25}$$

In (15.25), O stands for the $(M-1) \times (M-1)$ zero matrix; hopefully, the use of the same character here and in the O-symbol (e.g., $O(h^2)$) will not cause any confusion.

Now, the $[(M-1)(L-1)] \times [(M-1)(L-1)]$ matrix $\mathcal{A}$ in (15.25) is *block*-tridiagonal, but *not* tridiagonal. Namely, it has only 5 nonzero diagonals or subdiagonals, but the outer subdiagonals are not located next to the inner subdiagonals but separated from them by a band of zeros, with the band's width being $(M-2)$. Thus, the total width of the central nonzero band in matrix $\mathcal{A}$ is $2(M-2)+3$. Inverting such a matrix is not a computationally efficient process in the sense that it will require not $O(ML)$, but $O(ML)^2$ or $O(ML)^3$ operations. In other words, the number of operations required to solve Eq. (15.25) is *much greater* than the number of unknowns.[27]

Let us summarize what we have established about the CN method (15.17) for the 2D Heat equation. The method: (i) has accuracy $O(\kappa^2+h^2)$, (ii) is unconditionally stable, but (iii) requires much more operations per time step than the number of unknown variables. We are satisfied with features (i) and (ii), but not with (iii). In the remainder of this Lecture, we will be concerned with constructing methods that do not have the deficiency stated in (iii). For reference purposes, we will now repeat the properties that we want our "dream scheme" to have.

In order to be considered computationally efficient, the scheme:

(i) must have accuracy $O(\kappa^2+h^2)$ (or better);

(ii) must be unconditionally stable; (15.26)

(iii) must require the number of operations per time step that is proportional to the number of the unknowns.

In the next subsection, we will set the ground for obtaining such schemes.

15.4 Derivation of a computationally efficient scheme

In this section, we will derive a scheme which we will use later on to obtain methods that satisfy all the three conditions (15.26). Specifically, we pose the problem as follows: *Find a scheme that (a) reduces to the Crank-Nicolson scheme (13.6) in the case of the one-dimensional Heat equation and (b) has the same order of truncation error, i.e. $O(\kappa^2+h^2)$; or, in other words, satisfies property (i) of (15.26).* Of course, there are many (probably, infinitely many) such schemes. A significant contribution by computational scientists in the 1950's was finding, among those schemes, the ones which are unconditionally stable (property (ii)) and could be implemented in a time-efficient manner (property (iii)). In the remainder of this section, we

[27] One might have reasoned that, since $\mathcal{A}$ in (15.25) is block-tridiagonal, then one could solve Eq. (15.24) by the block-Thomas algorithm. This well-known generalization of the Thomas algorithm presented in Lecture 8 assumes that the coefficients a_k, b_k, c_k and α_k, β_k in (8.18) and (8.19) are $(M-1)\times(M-1)$ square matrices. Then formulae (8.21)–(8.23) of the Thomas algorithm are straightforwardly generalized by assigning the matrix sense to all the operations in those formulae.

However, this naive idea of being able to solve (15.24) by the block-Thomas algorithm does not work. Indeed, consider the defining equation for α_2 in (8.21). It involves β_1^{-1}. While matrix $\beta_1 = b_1$ is tridiagonal, its inverse β_1^{-1} is full. Hence α_2 is also a full matrix. Then by the last equation in (8.21), all subsequent β_k's are also full matrices. But then finding the inverse of each β_k in (8.21)–(8.23) would require $O(M^3)$ operations, and this would have to be repeated $O(L)$ times. Thus, the total operation count in this naive approach is $O(M^3L)$,