Finite Difference Methods (FDMs) 2

Time-dependent PDEs

A partial differential equation of the form

$$Au_{xx} + Bu_{xy} + Cu_{yy} = f(x, y, u, u_x, u_y) \quad \textbf{(15.1)}$$

where A, B, and C are constants, is called quasilinear. There are three types of quasilinear equations:

Elliptic, if $B^2 - 4AC < 0$

Parabolic, if $B^2 - 4AC = 0$

Hyperbolic, if $B^2 - 4AC > 0$

Forward Time Central Space (FTCS)

Heat/diffusion equation is an example of parabolic differential equations. The general 1D form of heat equation is given by

$$u_t = D\,u_{xx} + f(u, x, t) \qquad \textbf{(15.2)}$$

which is accompanied by initial and boundary conditions in order for the equation to have a unique solution.

We approximate temporal- and spatial-derivatives separately. Using explicit or forward Euler method, the difference formula for time derivative is

$$u_t = \frac{u(x_i, t_{j+1}) - u(x_i, t_j)}{\Delta t} + \mathcal{O}(\Delta t) \qquad \textbf{(15.3)}$$

Forward Time Central Space (FTCS)

And the difference formula for spatial derivative is

$$u_{xx} = \frac{u(x_{i-1}, t_j) - 2u(x_i, t_j) + u(x_{i+1}, t_j)}{\Delta x^2} + \mathcal{O}(\Delta x^2) \qquad \textbf{(15.4)}$$

We consider a simple heat/diffusion equation of the form

$$u_t = D\, u_{xx} \qquad \textbf{(15.5)}$$

that we want to solve in a 1D domain $0 \leq x < L$ within time interval $0 < t < T$.

The initial and boundary conditions are given by

$$u(x, 0) = f(x) \qquad u(0, t) = g_1(t)$$

$$u(L, t) = g_2(t)$$

Forward Time Central Space (FTCS)

Employing the notation $U_{i,j} = u(x_i, t_j)$, using the difference formulas (15.3) and (15.4), the heat equation (15.5) becomes (dropping the terms $\mathcal{O}(\Delta t)$ and $\mathcal{O}(\Delta x^2)$)

$$\frac{U_{i,j+1} - U_{i,j}}{\Delta t} = D\,\frac{U_{i-1,j} - 2U_{i,j} + U_{i+1,j}}{\Delta x^2} \qquad \textbf{(15.6)}$$

for $i = 0, 1, 2, \ldots, M$ and $j = 0, 1, 2, \ldots, N$

Here, *M* and *N* are the number of grid points, Δt and Δx are grid sizes (length of subintervals) along the *t*-axis and *x*-axis, respectively.

Letting

$$r = \frac{D\,\Delta t}{\Delta x^2} \qquad \textbf{(15.7)}$$

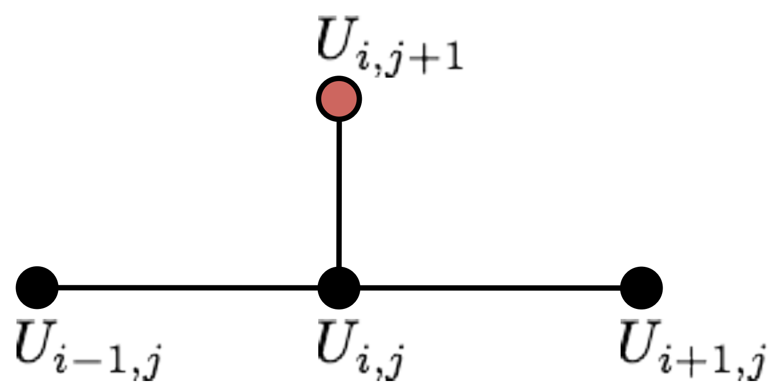
Forward Time Central Space (FTCS)

equation (15.6) can be rearranged into

$$U_{i,j+1} = \frac{\Delta t D}{\Delta x^2}\left(U_{i-1,j} - 2U_{i,j} + U_{i+1,j}\right) + U_{i,j}$$

$$U_{i,j+1} = rU_{i-1,j} + (1-2r)U_{i,j} + rU_{i+1,j} \quad \textbf{(15.8)}$$

The formula (15.8) explicitly gives the value $U_{i,j+1}$ in terms of $U_{i-1,j}, U_{i,j},$ and $U_{i+1,j}$. The computational stencil representing the situation in formula (15.8) is given



Forward Time Central Space (FTCS)

The explicit formula in (15.8) is stable if and only if

$$r = \frac{\Delta t D}{\Delta x^2} \leq \frac{1}{2} \qquad \textbf{(15.9)}$$

This means that the grid size Δt must satisfy

$$\Delta t \leq \frac{\Delta x^2}{2D} \qquad \textbf{(15.10)}$$

If this condition is not fulfilled, errors committed in one line $\{U_{i,j}\}$ may be magnified in subsequent lines $\{U_{i,p}\}$ for some $p > j$.

Forward Time Central Space (FTCS)

Step 1: Define problem parameters such as
- domain size
- number of grid points (or subintervals)
- grid size

Step 2: Define condition for stability, abort function if condition is not met

Step 3: Generate grids

Step 4: Initialize matrix for solution

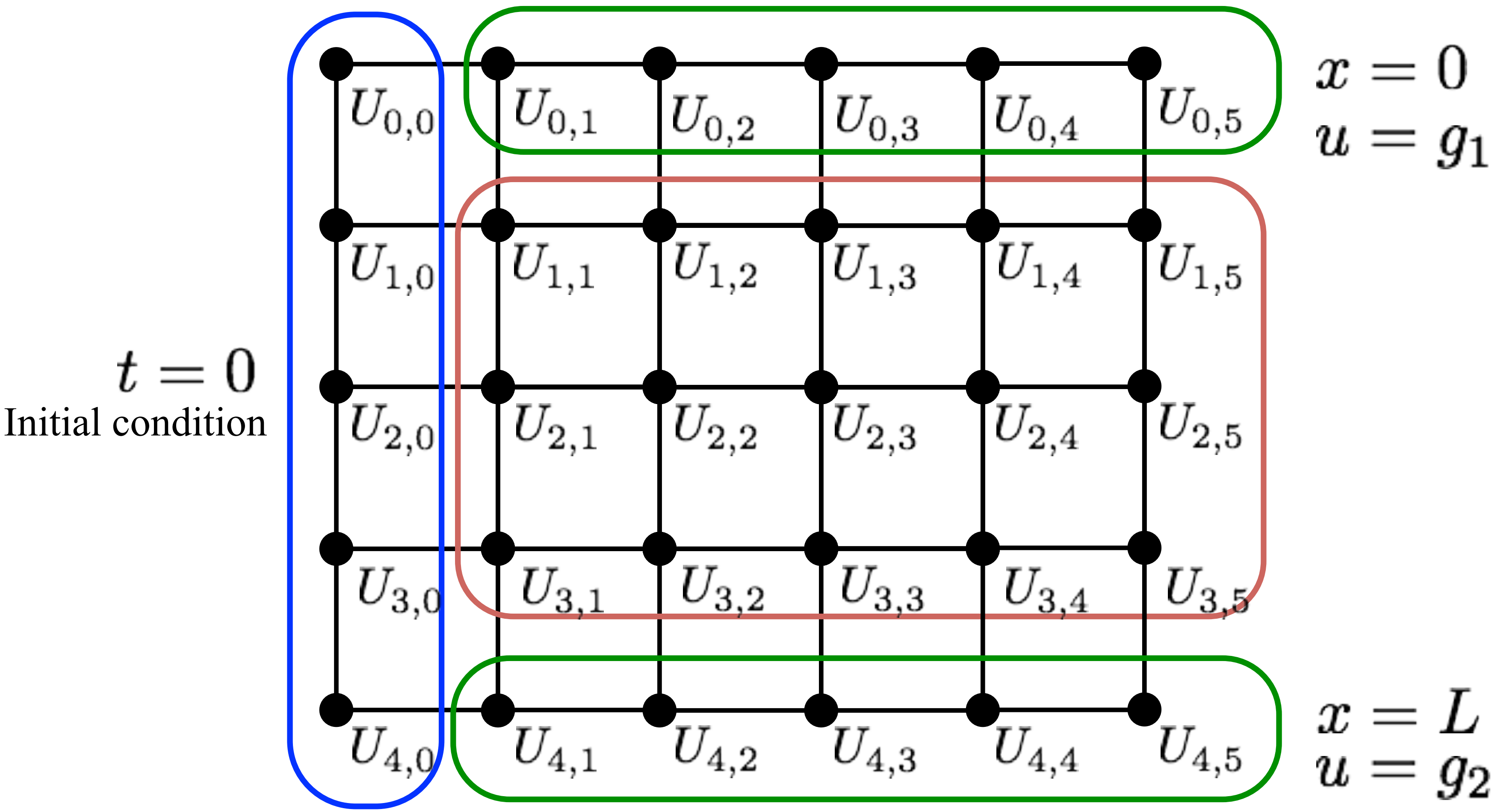
Step 5: Fill in initial and boundary conditions

Step 6: Iteration/solve the linear algebraic equations

Step 7: Visualization

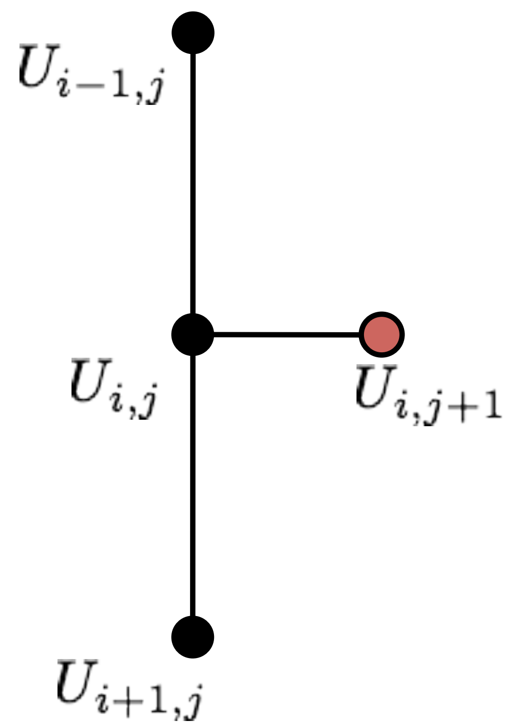
Forward Time Central Space (FTCS)

Grid to solve 1D heat equation with Dirichlet BC



Forward Time Central Space (FTCS)

If the arrangement of grid as in the image in the previous slide (slide 9) where space is discretized along *y*-axis with 5 grid points and time is discretized along *x*-axis with 6 grid points, then the computational stencil representing the situation is shown in this figure:



Example

Solve the following 1D heat/diffusion equation

$$u_t = u_{xx}$$

in a unit domain $0 \leq x \leq 1$ and time interval $0 \leq t \leq 0.2$ subject to:

initial condition $u(0,x) = 4x - 4x^2$

boundary condition: $u(t,0) = 0$ and $u(t,1) = 0$

Approximate with explicit/forward finite difference method and use the following:

$M = 12$ (number of grid points along x-axis)

$N = 100$ (number of grid points along t-axis)

Try other values of M and N to see if the stability condition works.

Example

Display error message and abort function if the stability condition is not fulfilled.

```
r = dt*D/dx^2;
s = 1 - 2*r;

condStab = dx^2/(2*D);

if dt > condStab
    error('Input parameters cause instability!');
end
```

Backward Time Central Space (BTCS)

The next method is called implicit or backward Euler method. In this method the formula for time derivative is given by

$$u_t = \frac{u(x_i, t_j) - u(x_i, t_{j-1})}{\Delta t} + \mathcal{O}(\Delta t) \qquad \textbf{(15.11)}$$

while the formula for spatial derivative may be similar to the formula in (15.4). The approximation of heat equation (15.5) becomes

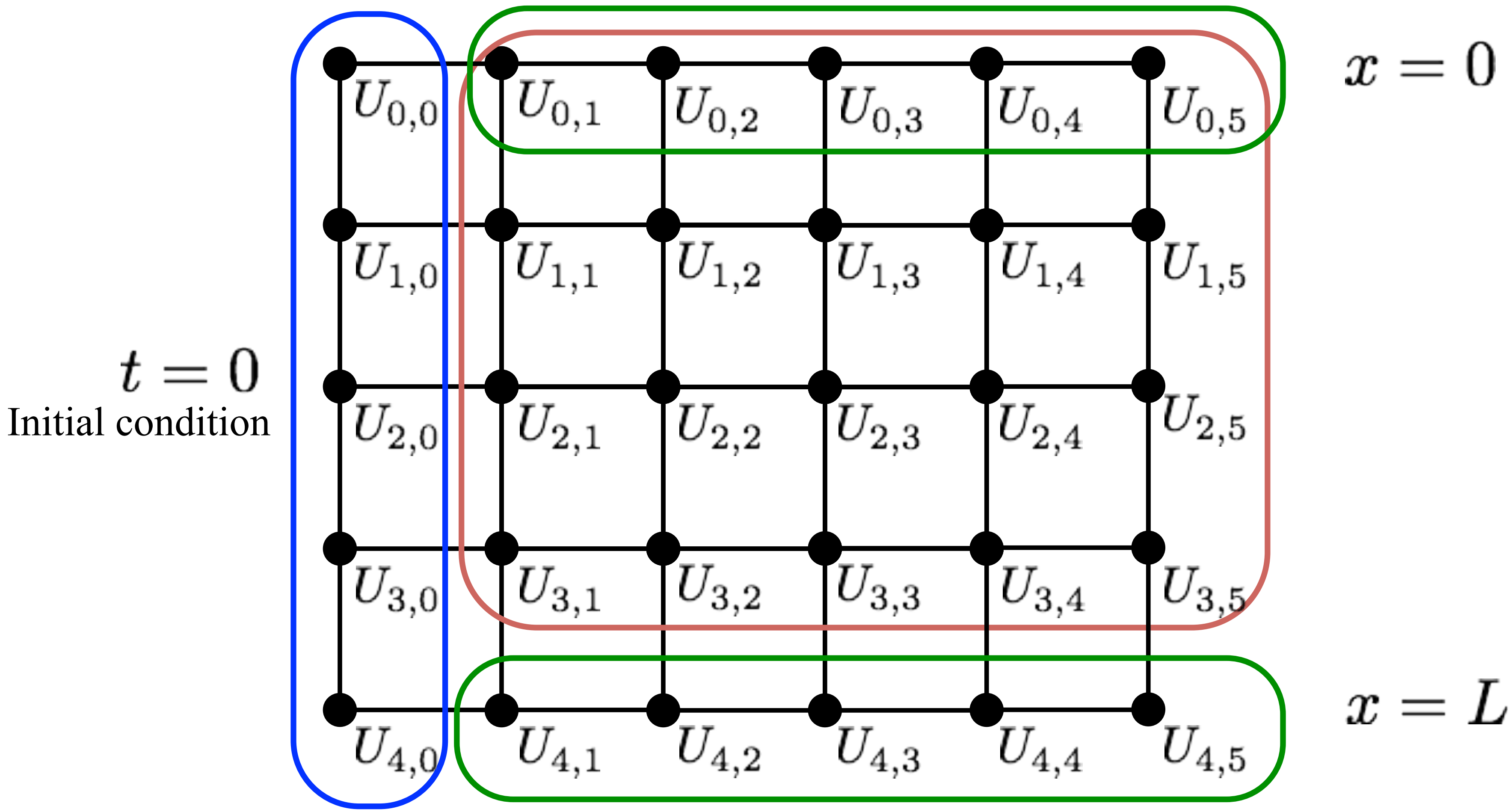
$$\frac{U_{i,j} - U_{i,j-1}}{\Delta t} = \frac{U_{i-1,j} - 2U_{i,j} + U_{i+1,j}}{\Delta x^2} \qquad \textbf{(15.12)}$$

Similarly, letting $r = \dfrac{D\,\Delta t}{\Delta x^2}$ and rearranging yields

$$U_{i,j-1} = -rU_{i-1,j} + (1+2r)U_{i,j} - rU_{i+1,j} \qquad \textbf{(15.13)}$$

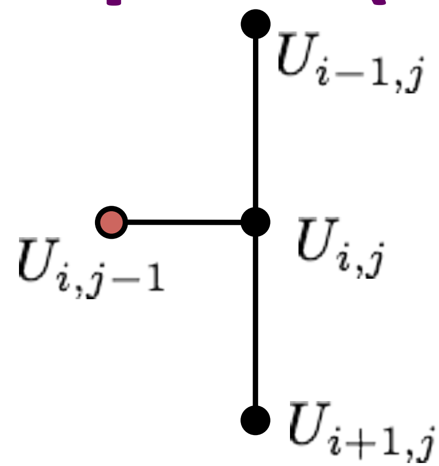
Backward Time Central Space (BTCS)

Grid for 1D heat equation



Backward Time Central Space (BTCS)

The computational stencil represents the implicit method is illustrated as in the right figure.



If the values at x end points are given from the Dirichlet type of boundary condition, equation (15.13) becomes a system of simultaneous equations, as in:

$$\begin{bmatrix} 1+2r & -r & & & & \\ -r & 1+2r & -r & & & \\ & \ddots & & & & \\ & & \ddots & & & \\ & -r & 1+2r & -r & & \\ & & -r & 1+2r & & \end{bmatrix} \begin{bmatrix} U_{1,j} \\ U_{2,j} \\ \vdots \\ \vdots \\ U_{M-3,j} \\ U_{M-2,j} \end{bmatrix} = \begin{bmatrix} rU_{0,j} & + & U_{1,j-1} \\ 0 & + & U_{2,j-1} \\ \vdots & & \vdots \\ \vdots & & \vdots \\ 0 & + & U_{M-3,j-1} \\ rU_{M-1,j} & + & U_{M-2,j-1} \end{bmatrix}$$

Backward Time Central Space (BTCS)

In MATLAB, the linear equation is solved by iterating over time discretization:

```
for k=2:N+1
    % Right hand side vector
    b = [r*U(1,k); zeros(M-3,1); r*U(M+1,k)] + U(2:M,k-1);
    % Solve for linear equation
    U(2:M,k) = A\b;
end
```

Backward Time Central Space (BTCS)

If Neumann boundary condition is applied, where $\partial_x u = g$ at $x = 0$, this type of boundary is approximated by

$$-\left(\frac{U_{i+1,j} - U_{i-1,j}}{2\Delta x}\right) = g_{i,j}$$

At $x = 0$, or $i = 0$ the formula is rearranged to get

$$U_{-1,j} - U_{1,j} = 2\Delta x \, g_{i,j}$$

Hence along the $x = 0$ axis, the approximation (15.13) becomes

$$U_{0,j-1} = (1 + 2r)U_{0,j} - 2rU_{1,j} - 2r\Delta x \, g_{0,j} \quad \textbf{(15.14)}$$

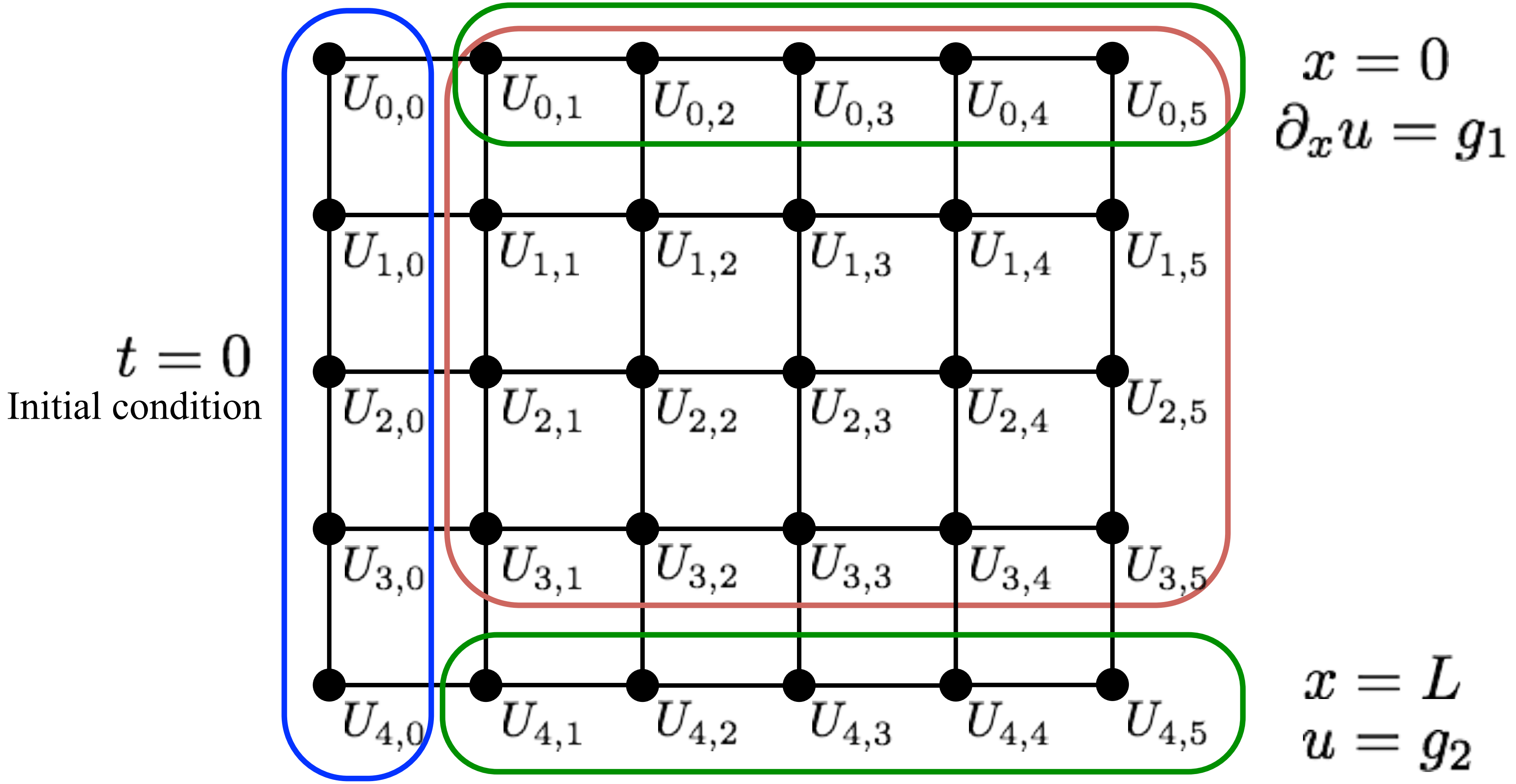
Backward Time Central Space (BTCS)

And the simultaneous equations in matrix-vector notation:

$$\begin{bmatrix} 1+2r & -2r & & & \\ -r & 1+2r & -r & & \\ & \ddots & & & \\ & & \ddots & & \\ & -r & 1+2r & -r & \\ & & -r & 1+2r & \end{bmatrix} \begin{bmatrix} U_{0,j} \\ U_{1,j} \\ \vdots \\ \vdots \\ U_{M-2,j} \\ U_{M-1,j} \end{bmatrix} = \begin{bmatrix} U_{0,j-1} + 2r\Delta x g_{0,j} \\ U_{1,j-1} \\ U_{2,j-1} \\ \vdots \\ U_{M-2,j-1} \\ U_{M-1,j-1} + rU_{M,j} \end{bmatrix}$$

Backward Time Central Space (BTCS)

Grid for 1D heat equation with Neumann + Dirichlet BCs



Backward Time Central Space (BTCS)

If both boundaries at $x = 0$ and $x = L$ are equipped with Neumann boundary conditions:

$$\partial_x u = g \quad \text{at} \quad x = 0$$
$$\partial_x u = f \quad \text{at} \quad x = L$$

the simultaneous equations in matrix-vector notation becomes:

$$\begin{bmatrix} 1+2r & -2r & & & \\ -r & 1+2r & -r & & \\ & \ddots & & & \\ & & \ddots & & \\ & -r & 1+2r & -r & \\ & & -2r & 1+2r \end{bmatrix} \begin{bmatrix} U_{0,j} \\ U_{1,j} \\ \vdots \\ \vdots \\ U_{M-2,j} \\ U_{M-1,j} \end{bmatrix} = \begin{bmatrix} 2r\Delta x g_{0,j} & + & U_{0,j-1} \\ 0 & + & U_{1,j-1} \\ \vdots & & \vdots \\ \vdots & & \vdots \\ 0 & + & U_{M-2,j-1} \\ 2r\Delta x f_{M-1,j} & + & U_{M-1,j-1} \end{bmatrix}$$

Crank-Nicolson Method

An implicit scheme, invented by John Crank and Phyllis Nicolson, is based on numerical approximations for solutions of differential equation (15.1) at the point $\left(x_i, t_{j+\frac{\Delta t}{2}}\right)$ that lies between the rows in the grid.

The approximation formula for time derivative is given by

$$u_t\left(x_i, t_{j+\frac{\Delta t}{2}}\right) = \frac{u(x_i, t_{j+1}) - u(x_i, t_j)}{\Delta t} + \mathcal{O}(\Delta t^2) \quad \textbf{(15.15)}$$

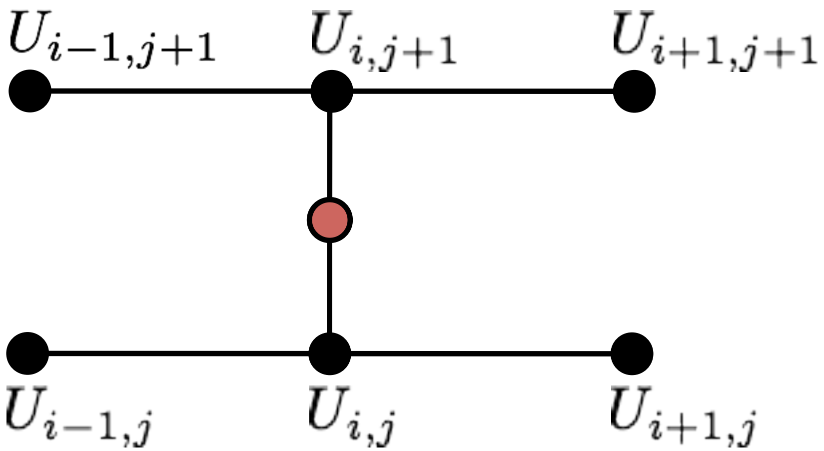
and for spatial derivative

$$u_{xx}\left(x_i, t_{j+\frac{\Delta t}{2}}\right) = \frac{1}{2\Delta x^2}\left(u(x_{i-1}, t_{j+1}) - 2u(x_i, t_{j+1}) + u(x_{i+1}, t_{j+1})\right.$$
$$\left. + u(x_{i-1}, t_j) - 2u(x_i, t_j) + u(x_{i+1}, t_j)\right) + \mathcal{O}(\Delta x^2)$$

(15.16)

Crank-Nicolson Method

The Crank-Nicolson method in numerical stencil is illustrated as in the right figure.



In a similar fashion to the previous derivation, the difference equation for Crank-Nicolson method is

$$\frac{U_{i,j+1} - U_{i,j}}{\Delta t} = \frac{D}{2\Delta x^2}\left(U_{i-1,j+1} - 2U_{i,j+1} + U_{i+1,j+1} + U_{i-1,j} - 2U_{i,j} + U_{i+1,j}\right)$$

(15.17)

Crank-Nicolson Method

Multiplying both sides with Δt and substituting

$$r = \frac{D\Delta t}{2\Delta x^2}$$

and rearranging gives

$$-rU_{i-1,j+1} + (1+2r)U_{i,j+1} - rU_{i+1,j+1} = rU_{i-1,j} + (1-2r)U_{i,j} + rU_{i+1,j} \qquad \textbf{(15.18)}$$

Crank-Nicolson Method

For diffusion equation with Dirichlet boundary conditions, using the grid as in slide 14, equation (15.18) can be represented in matrix-vector notation

$$
\begin{bmatrix}
1+2r & -r & & & \\
-r & 1+2r & -r & & \\
\ddots & \ddots & \ddots & & \\
& \ddots & \ddots & \ddots & \\
& -r & 1+2r & -r & \\
& & -r & 1+2r &
\end{bmatrix}
\begin{bmatrix}
U_{1,j+1} \\ U_{2,j+1} \\ \vdots \\ \vdots \\ U_{M-3,j+1} \\ U_{M-2,j+1}
\end{bmatrix}
=
\begin{bmatrix}
1-2r & r & & & \\
r & 1-2r & r & & \\
\ddots & \ddots & \ddots & & \\
& \ddots & \ddots & \ddots & \\
& r & 1-2r & r & \\
& & r & 1-2r &
\end{bmatrix}
\begin{bmatrix}
U_{1,j} \\ U_{2,j} \\ \vdots \\ \vdots \\ U_{M-3,j} \\ U_{M-2,j}
\end{bmatrix}
$$

$$
+ \begin{bmatrix}
rU_{0,j} + rU_{0,j+1} \\ 0 \\ \vdots \\ \vdots \\ 0 \\ rU_{M-1,j} + rU_{M-1,j+1}
\end{bmatrix}
$$