Computational Finance with C++

Finite Difference Methods for Partial Differential Equations

Panos Parpas
Imperial College London
p.parpas@imperial.ac.uk

Finite Difference Methods for Partial Differential Equations

- From the Black and Scholes Equation to the Heat Equation
- Explicit finite difference methods
- Stability of the explicit finite difference scheme
- Implicit Schemes
- Boundary Conditions
- Finite Difference vs Monte Carlo

Model Problem: The BSE PDE

The Black and Scholes Equation:

$$\frac{\partial V}{\partial t} + \frac{\sigma^2}{2} S^2 \frac{\partial^2 V}{\partial S^2} + rS\frac{\partial V}{\partial S} - rV = 0$$

$$V(S,T) = \Phi(S,T)$$

where Φ is the pay-off function e.g. $\Phi(S,T) = \max[S-K,0]$

Risk Neutral Dynamics:

$$dS(t) = rS(t)dt + \sigma S(t)dW(t) \qquad S(0) = s$$

Model Problem: The BSE PDE

The Black and Scholes Equation:

$$\frac{\partial V}{\partial t} + \frac{\sigma^2}{2} S^2 \frac{\partial^2 V}{\partial S^2} + rS\frac{\partial V}{\partial S} - rV = 0$$

$$V(S,T) = \Phi(S,T)$$

where Φ is the pay-off function e.g. $\Phi(S,T) = \max[S-K,0]$

Risk Neutral Dynamics:

$$dS(t) = rS(t)dt + \sigma S(t)dW(t) \qquad S(0) = s$$

A good idea when dealing with PDE is to attempt to simplify it

BSE happens to be related to the Heat Equation with a change of variables

From the BSE PDE to the Heat Equation

The Black and Scholes Equation:

$$\frac{\partial V}{\partial t} + \frac{\sigma^2}{2} S^2 \frac{\partial^2 V}{\partial S^2} + rS\frac{\partial V}{\partial S} - rV = 0$$

$$V(S,T) = \Phi(S,T)$$

Following the change of variables (derived on the board), we have:

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2}$$

$$u(x,0) = e^{-\alpha x}\Phi(e^x, T)$$

← The Heat Equation

From the BSE PDE to the Heat Equation

The Black and Scholes Equation:

$$\frac{\partial V}{\partial t} + \frac{\sigma^2}{2} S^2 \frac{\partial^2 V}{\partial S^2} + rS\frac{\partial V}{\partial S} - rV = 0$$

$$V(S,T) = \Phi(S,T)$$

Following the change of variables (derived on the board), we have:

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2}$$

$$u(x,0) = e^{-\alpha x}\Phi(e^x, T)$$

← The Heat Equation

Why analyse the heat equation?

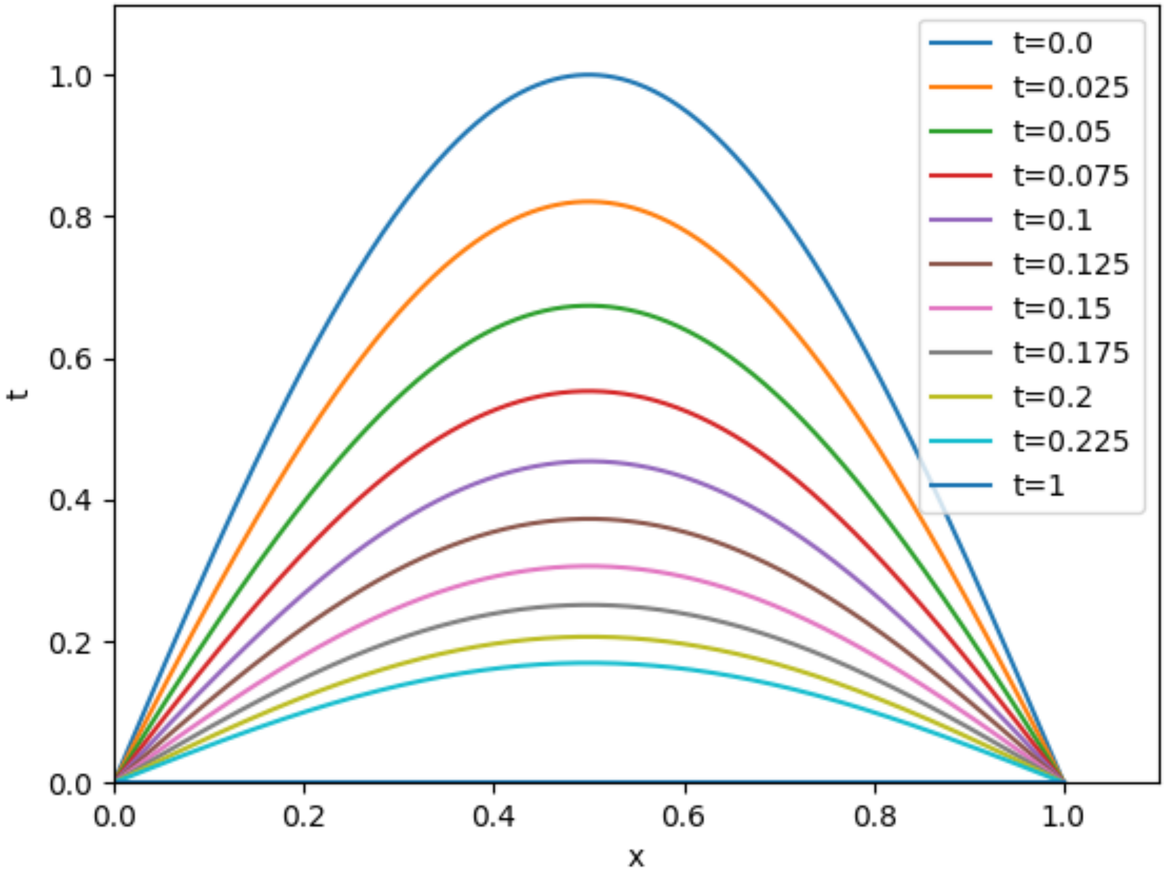
- The finite difference method can (in principle) be applied to the BSE directly
- The **two equations are equivalent**
- But **numerically they are not equivalent**, the heat equation behaves much better!
- **Theoretically it is easier to analyse the heat equation**

Example

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2}$$

Boundary conditions: $u(x,0) = \sin(\pi x), u(0,t) = 0,\ u(1,t) = 0$

Exact Solition: $u(x,t) = e^{-\pi^2 x} \sin(\pi x)$



Foundations of Finite Difference Methods

Mean Value Theorem: $f(x+h) = f(x) + f'(x)h + \frac{1}{2}f''(\xi)h^2,$ where $\xi \in [x, x+h]$

Rearrange: $f'(x) = \frac{f(x+h) - f(x)}{h} - \frac{1}{2}f''(\xi)h,$ where $\xi \in [x, x+h]$

Create equidistant grid: $\cdots < x_{i-1} < x_i < \cdots$

Constant mesh size: $h,$ use notation $f_i = f(x_i),$ and write

$$f'(x_i) = \frac{f_{i+1} - f_i}{h} + O(h)$$

Basic idea: Use two meshes sizes one for space and one for time

Δx Δt

Foundations of Finite Difference Methods

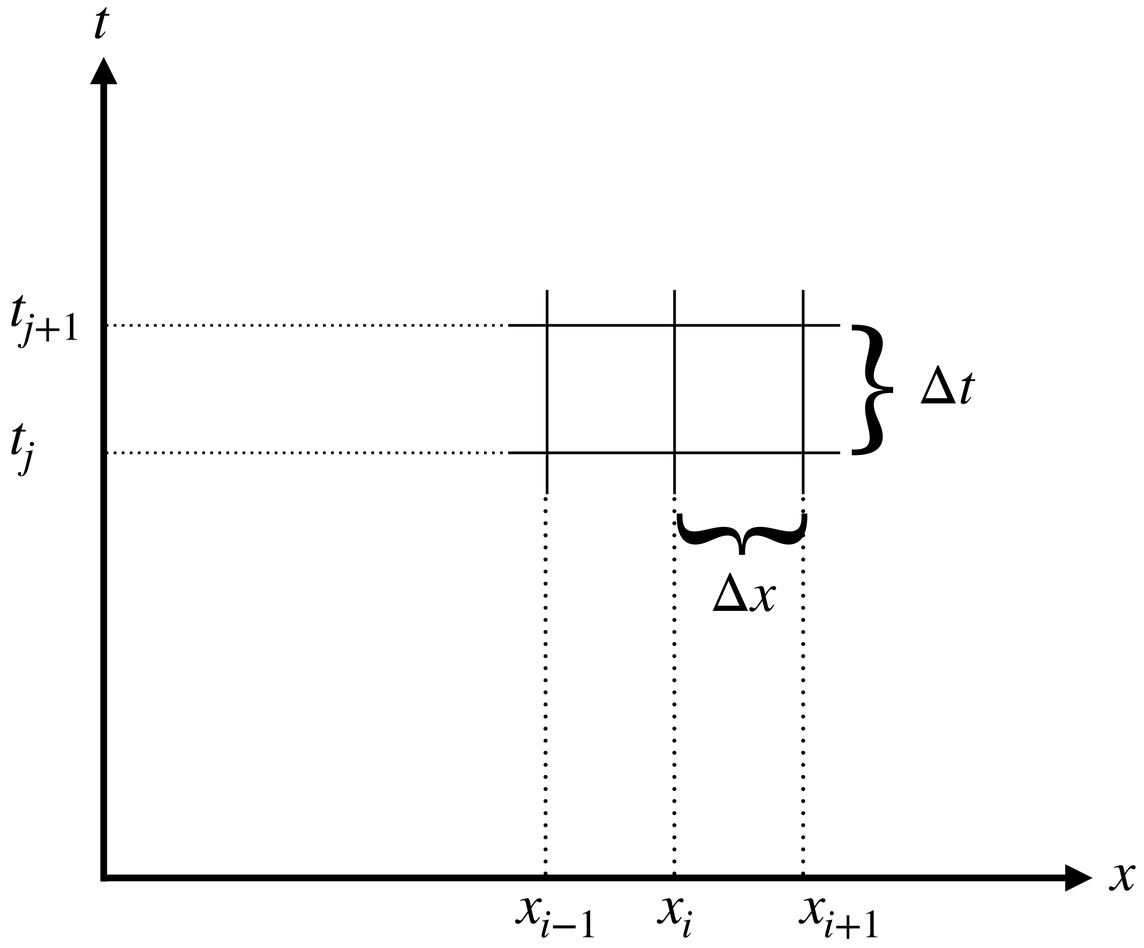
Forward Difference: $f'(x_i) = \frac{f_{i+1} - f_{i-1}}{2h} + O(h)$

Central Difference: $f'(x_i) = \frac{f_{i+1} - f_{i-1}}{2h} + O(h)$

Central Difference:
(second derivative) $f''(x_i) = \frac{f_{i+1} - 2f_i + f_{i-1}}{h^2} + O(h)$

Similar formulas can be used for the time derivative (but lead to very different schemes)

The Finite Difference Grid



Boundary Conditions:

Time Domain: $0 \leq t \leq \frac{\sigma^2 T}{2}$

Space Domain:

Select $0 \leq S_{\min} < S_{\max}$ and set

$x_{\min} = \ln S_{\min}$, $x_{\max} = \ln S_{\max}$

Explicit Finite Difference Scheme

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} \longrightarrow$$

$$\frac{\partial u}{\partial t} \approx \frac{u_{i,j+1} - u_{i,j}}{\Delta t} \quad \text{(Forward Difference)}$$

$$\frac{\partial^2 u}{\partial x^2} \approx \frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{\Delta x^2} \quad \text{(Central Difference)}$$

$$\downarrow$$

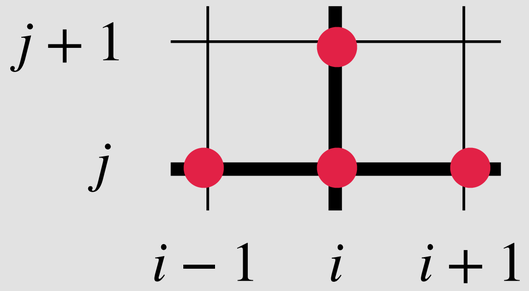
$$\frac{u_{i,j+1} - u_{i,j}}{\Delta t} = \frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{\Delta x^2}$$

$$u_{i,j+1} = u_{i,j} + \underbrace{\frac{\Delta t}{\Delta x^2}}_{\lambda} \left(u_{i+1,j} - 2u_{i,j} + u_{i-1,j} \right)$$

$$\downarrow$$

$$u_{i,j+1} = \lambda u_{i-1,j} + (1 - 2\lambda) u_{i,j} + \lambda u_{i+1,j}$$

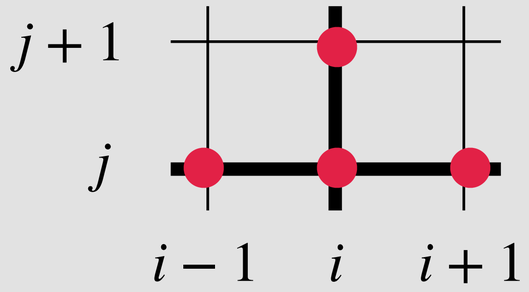
Explicit Finite Difference **Stencil**



Explicit Finite Difference Scheme

$$u_{i,j+1} = \lambda u_{i-1,j} + (1-2\lambda)u_{i,j} + \lambda u_{i+1,j}$$

Explicit Finite Difference **Stencil**



Vector Form:

$$u^j = \begin{bmatrix} u_{1,j} \\ u_{2,j} \\ \vdots \\ u_{m-1,j} \end{bmatrix}, \; u^0 \text{ is given}, A = \begin{pmatrix} 1-2\lambda & \lambda & 0 & \cdots & 0 \\ \lambda & 1-2\lambda & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \lambda \\ 0 & \cdots & 0 & \lambda & 1-2\lambda \end{pmatrix},$$

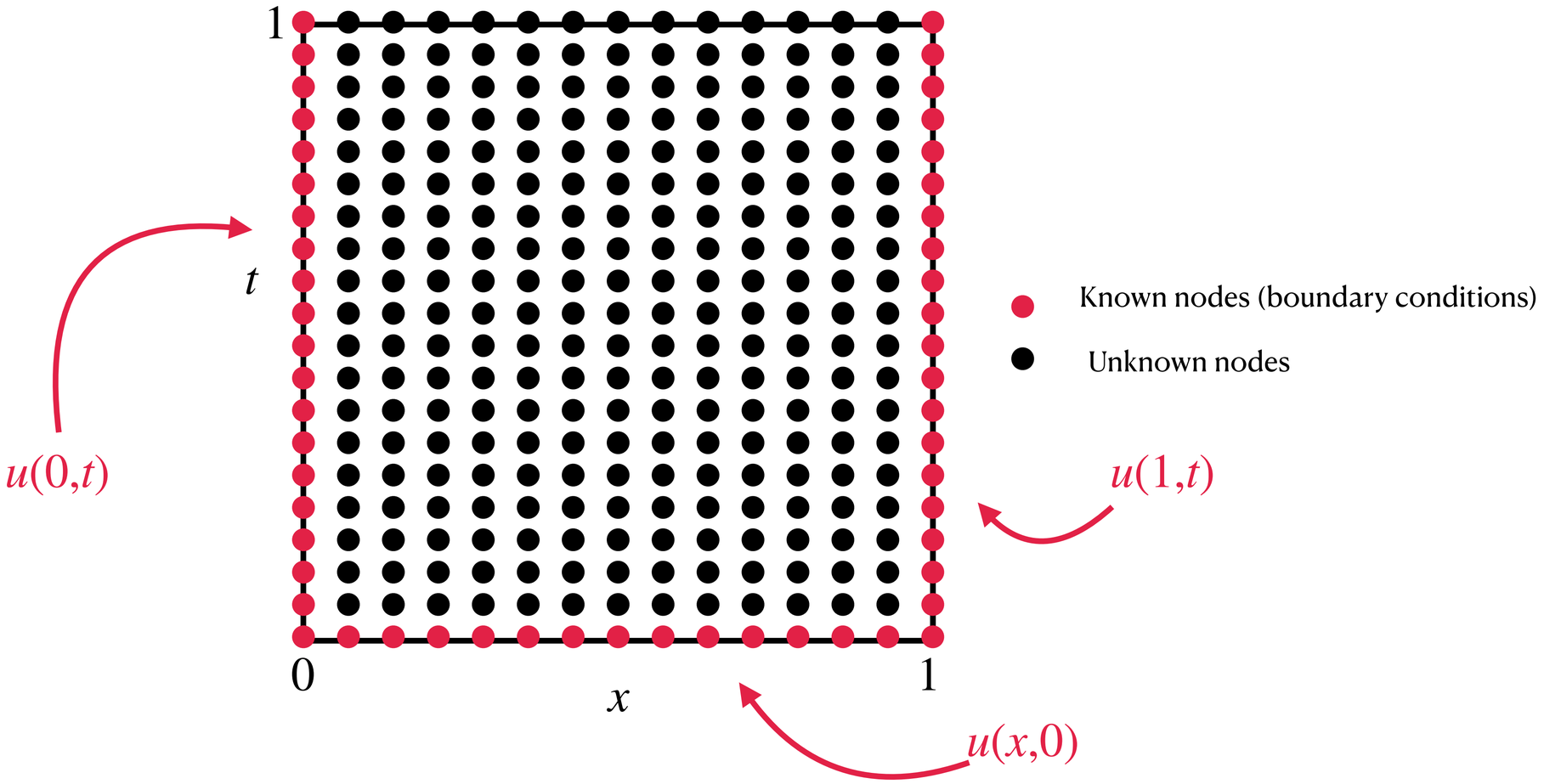
$$u^{j+1} = Au^j$$

Example

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2}$$

Boundary conditions: $u(x,0) = \sin(\pi x), u(0,t) = 0, u(1,t) = 0$

Exact Solition: $u(x,t) = e^{-\pi^2 x} \sin(\pi x)$

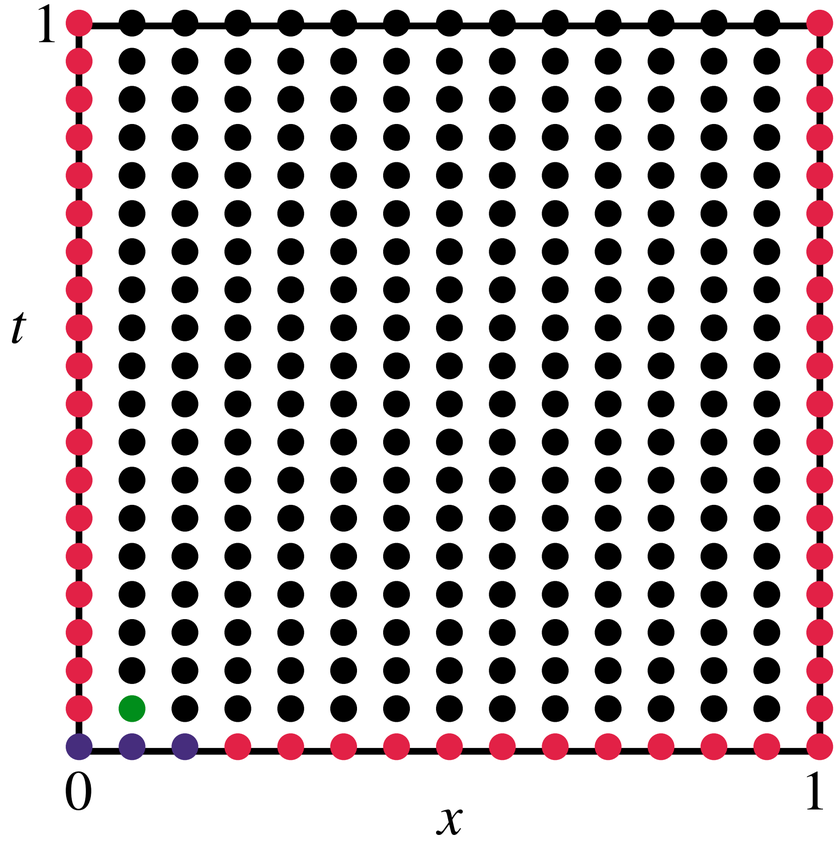


Example

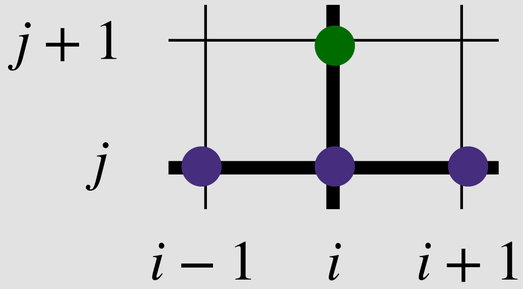
$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2}$$

Boundary conditions: $u(x,0) = \sin(\pi x), u(0,t) = 0$

Exact Solition: $u(x,t) = e^{-\pi^2 x} \sin(\pi x)$



Explicit Finite Difference **Stencil**



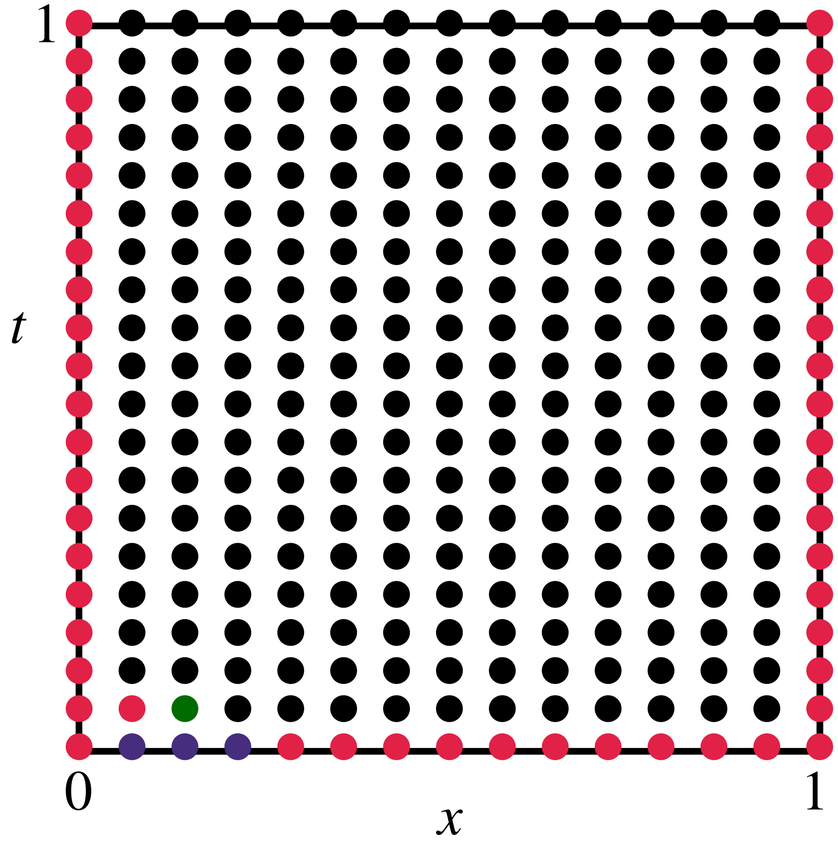
- Known nodes (boundary conditions)
- Unknown nodes
- $u(1,1) = \lambda u_{0,0} + (1 - 2\lambda)u_{1,0} + \lambda u_{2,0}$

Example

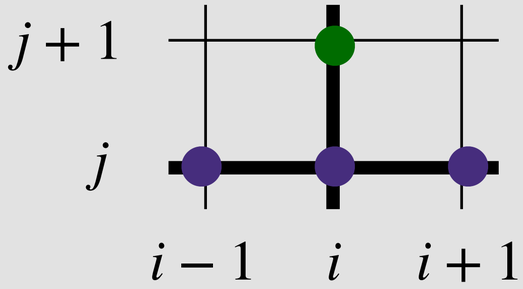
$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2}$$

Boundary conditions: $u(x,0) = \sin(\pi x), u(0,t) = 0$

Exact Solition: $u(x,t) = e^{-\pi^2 x} \sin(\pi x)$



Explicit Finite Difference **Stencil**



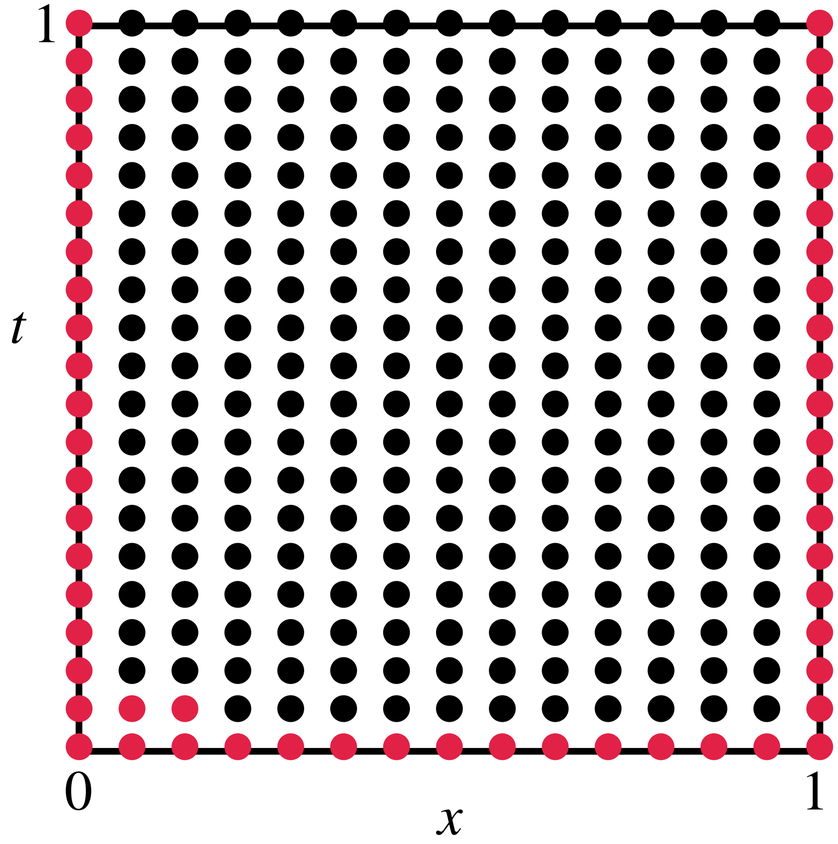
- Known nodes (boundary conditions)
- Unknown nodes
- $u(2,1) = \lambda u_{1,0} + (1 - 2\lambda)u_{2,0} + \lambda u_{3,0}$

Example

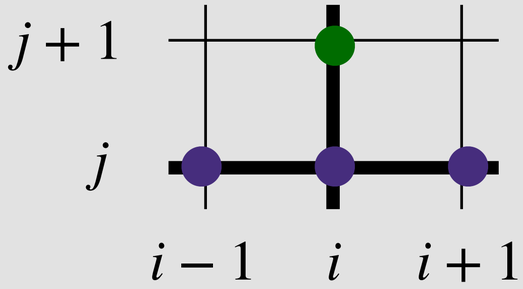
$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2}$$

Boundary conditions: $u(x,0) = \sin(\pi x), u(0,t) = 0$

Exact Solition: $u(x,t) = e^{-\pi^2 x} \sin(\pi x)$



Explicit Finite Difference **Stencil**



Known nodes (boundary conditions)

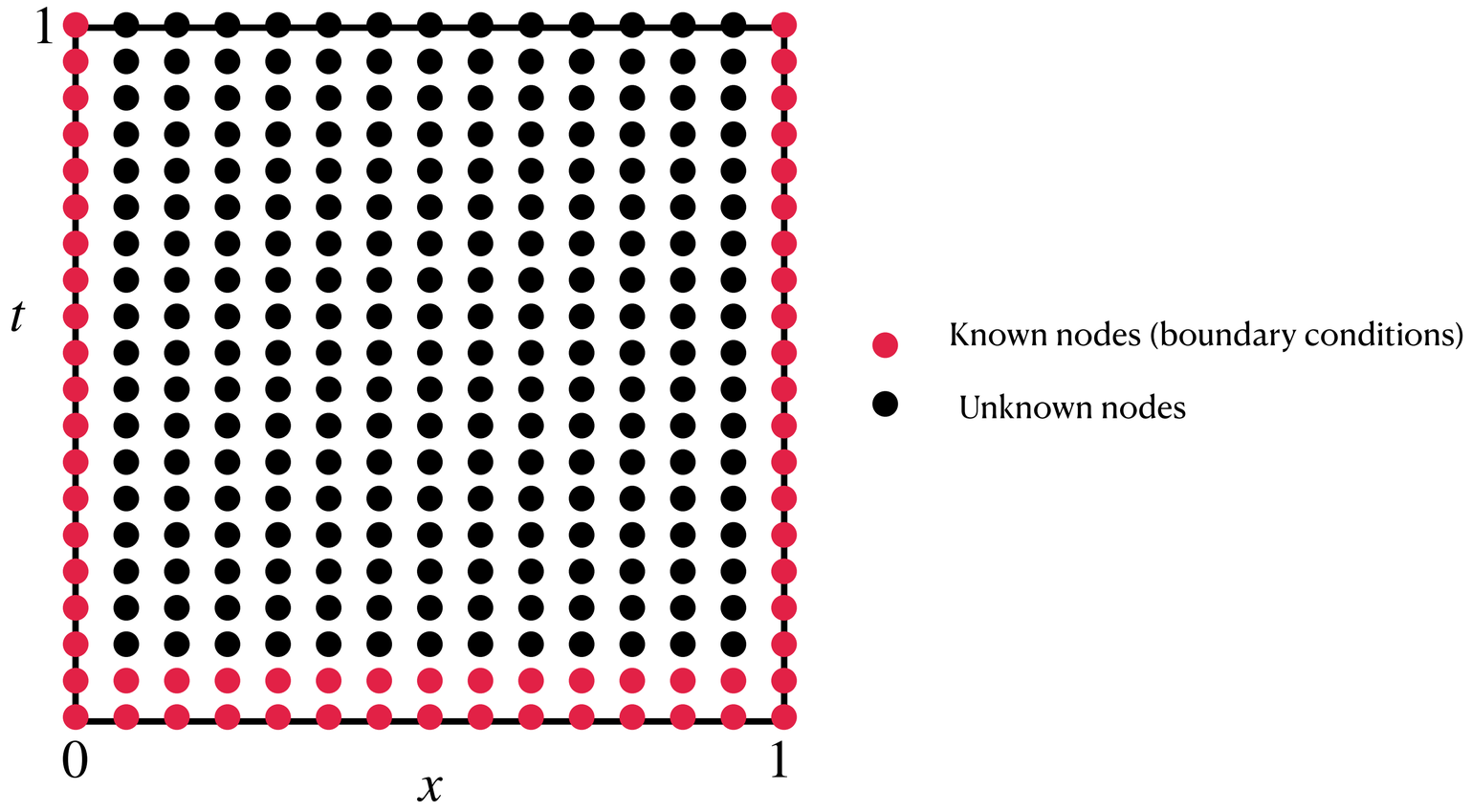
Unknown nodes

Example

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2}$$

Boundary conditions: $u(x,0) = \sin(\pi x),\ u(0,t) = 0,\ u(1,t) = 0$

Exact Solition: $u(x,t) = e^{-\pi^2 x} \sin(\pi x)$

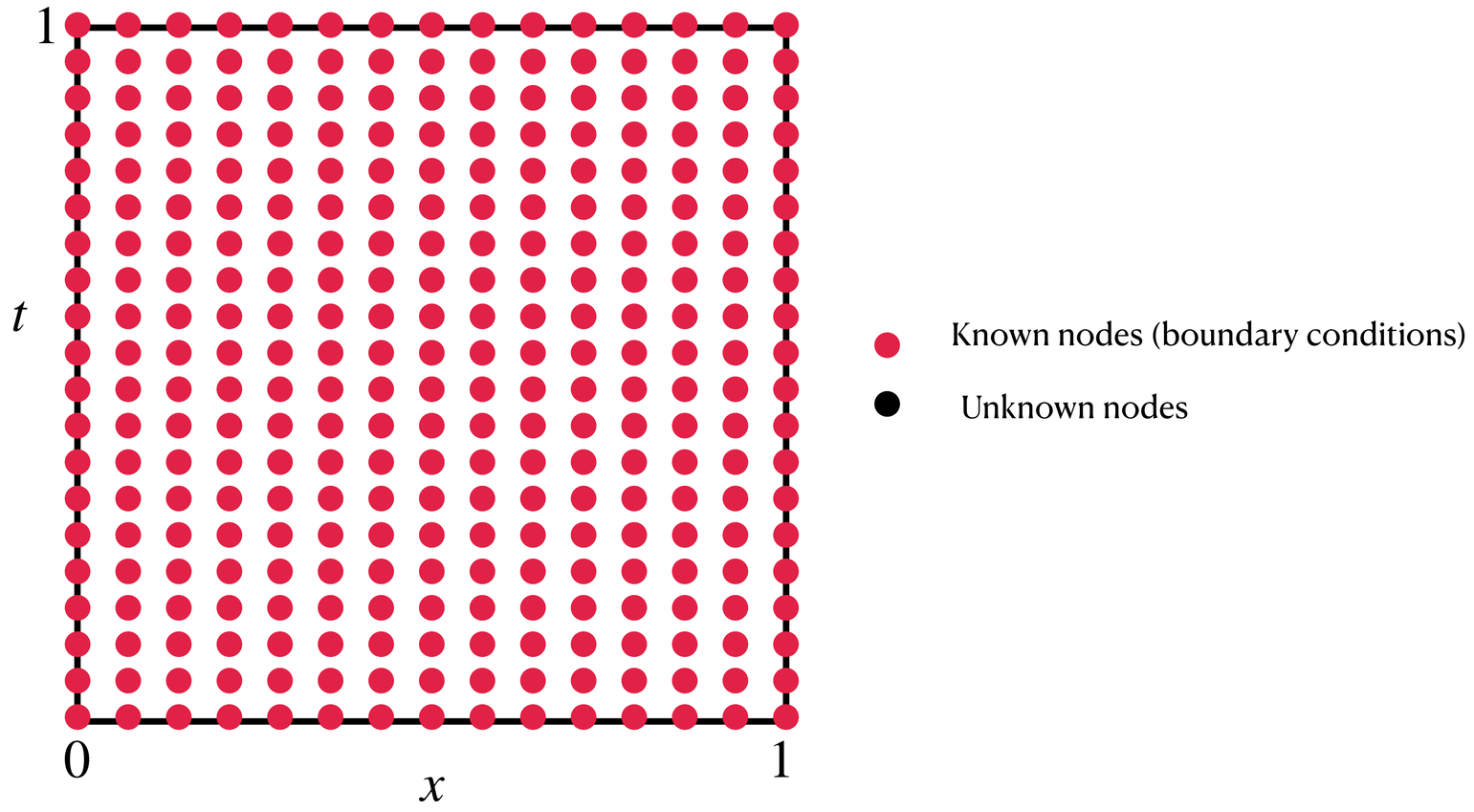


Example

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2}$$

Boundary conditions: $u(x,0) = \sin(\pi x),\ u(0,t) = 0,\ u(1,t) = 0$

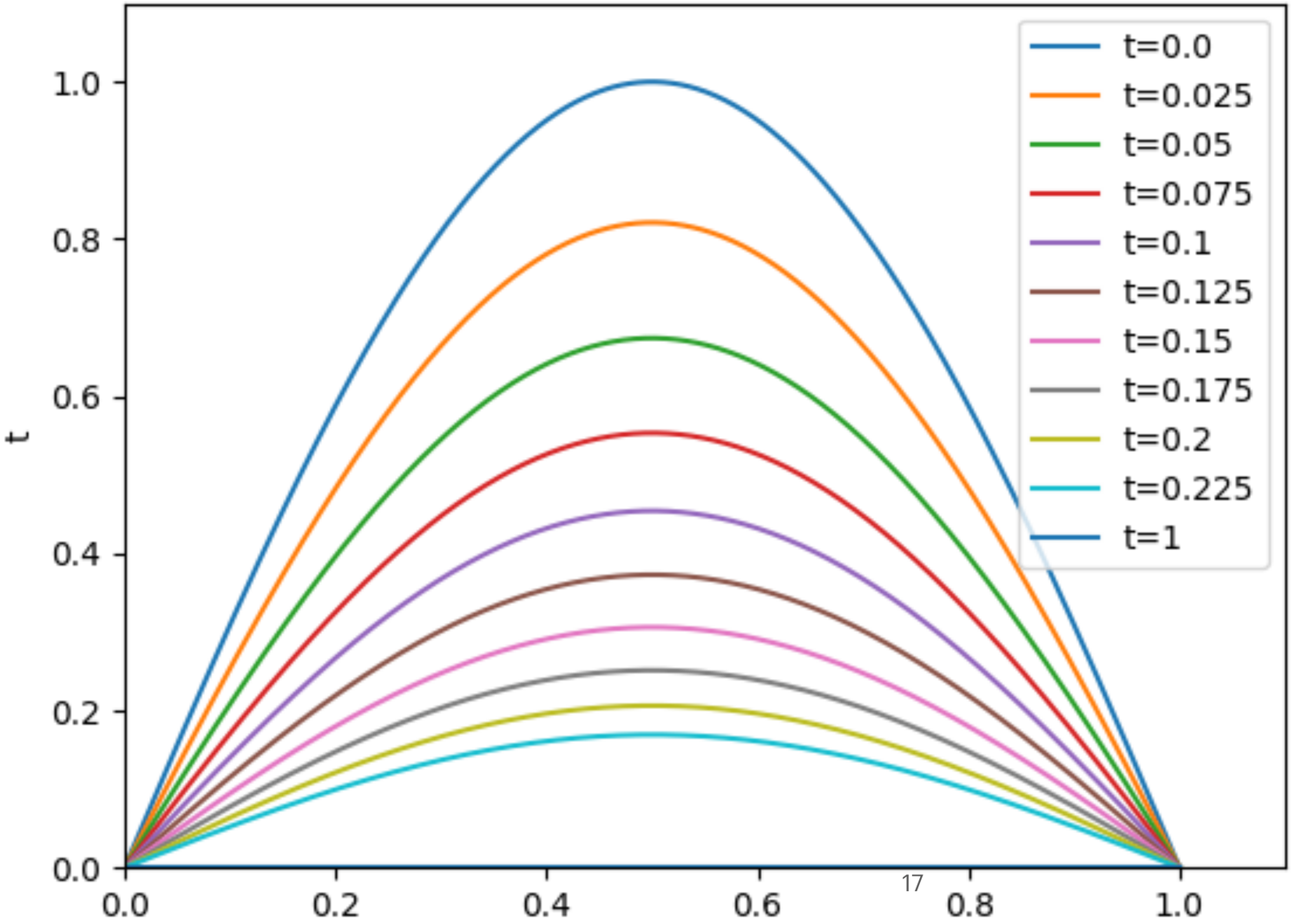
Exact Solition: $u(x,t) = e^{-\pi^2 x}\sin(\pi x)$



$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2}$$

Does the method converge?

Boundary conditions: $u(x,0) = \sin(\pi x), u(0,t) = 0, \ u(1,t) = 0$,Solution: $u(x,t) = e^{-\pi^2 x} \sin(\pi x)$



Discrete Scheme:

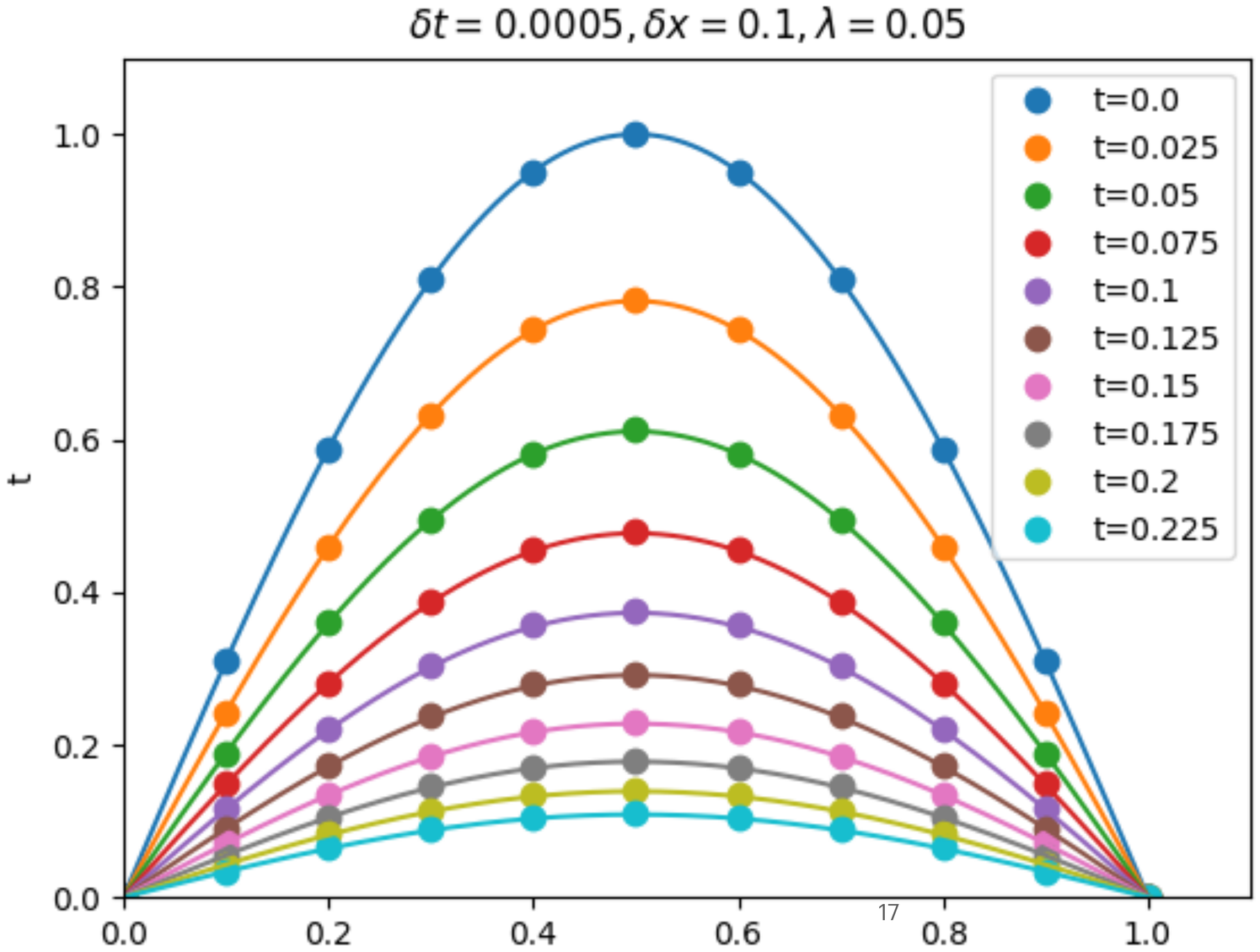
$$u_{i,j+1} = \lambda u_{i-1,j} + (1 - 2\lambda)u_{i,j} + \lambda u_{i+1,j}$$

$$\lambda = \frac{\Delta t}{\Delta x^2}$$

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2}$$

Does the method converge?

Boundary conditions: $u(x,0) = \sin(\pi x), u(0,t) = 0,\ u(1,t) = 0$,Solution: $u(x,t) = e^{-\pi^2 x}\sin(\pi x)$



Discrete Scheme:

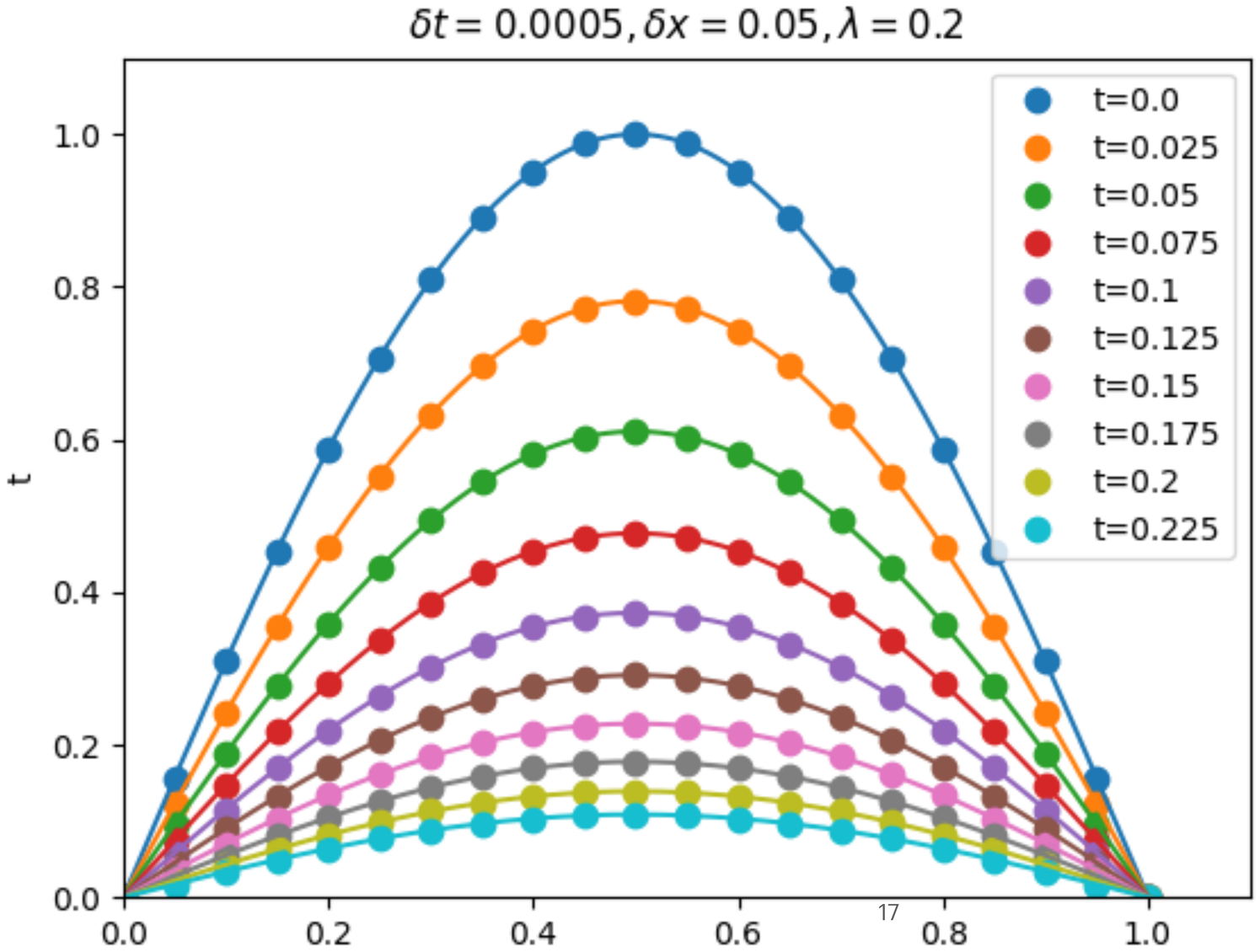
$$u_{i,j+1} = \lambda u_{i-1,j} + (1-2\lambda)u_{i,j} + \lambda u_{i+1,j}$$

$$\lambda = \frac{\Delta t}{\Delta x^2}$$

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2}$$

Does the method converge?

Boundary conditions: $u(x,0) = \sin(\pi x), u(0,t) = 0,\ u(1,t) = 0$,Solution: $u(x,t) = e^{-\pi^2 x}\sin(\pi x)$



Discrete Scheme:

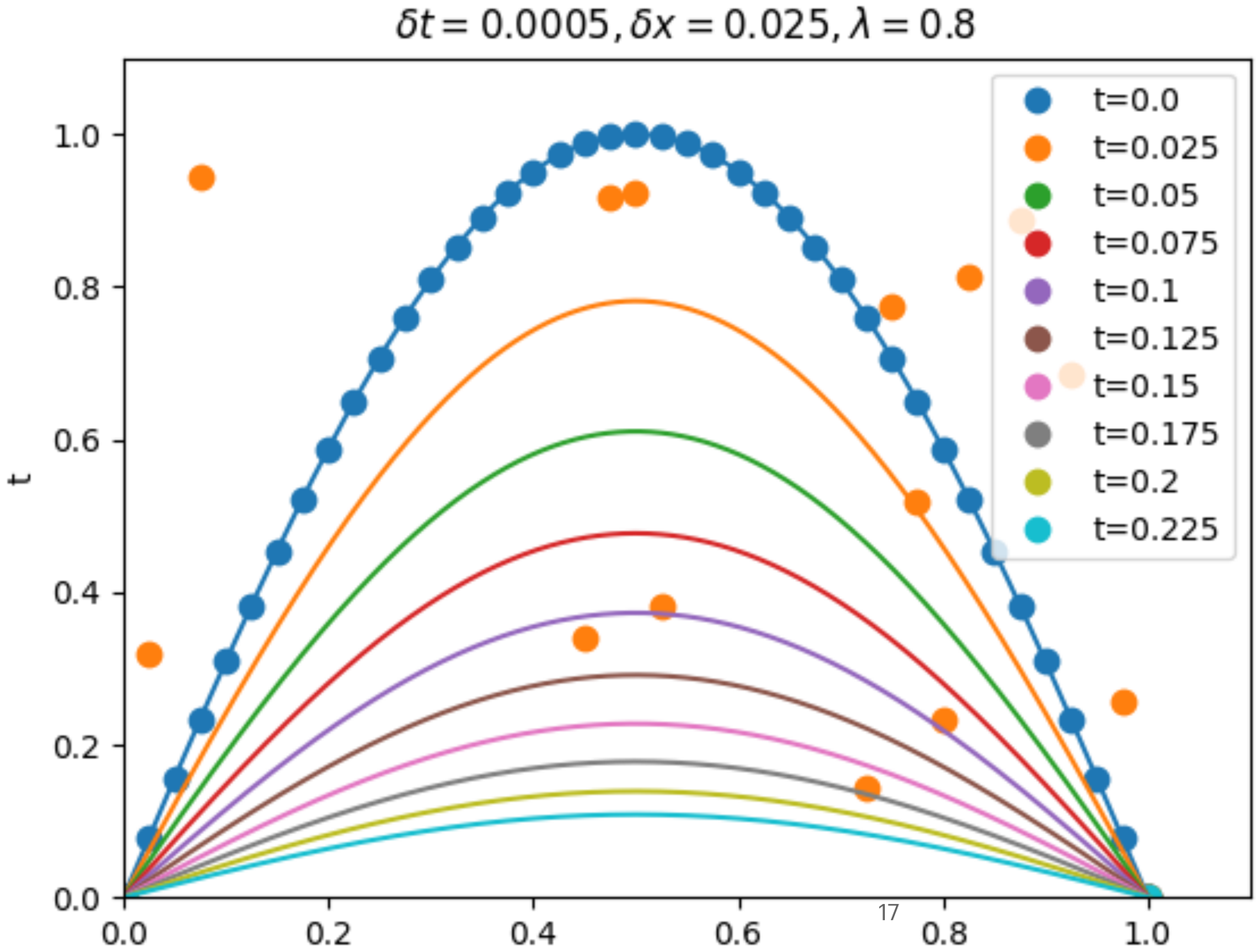
$$u_{i,j+1} = \lambda u_{i-1,j} + (1-2\lambda)u_{i,j} + \lambda u_{i+1,j}$$

$$\lambda = \frac{\Delta t}{\Delta x^2}$$

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2}$$

Does the method converge?

Boundary conditions: $u(x,0) = \sin(\pi x), u(0,t) = 0,\ u(1,t) = 0$,Solution: $u(x,t) = e^{-\pi^2 x}\sin(\pi x)$



Discrete Scheme:

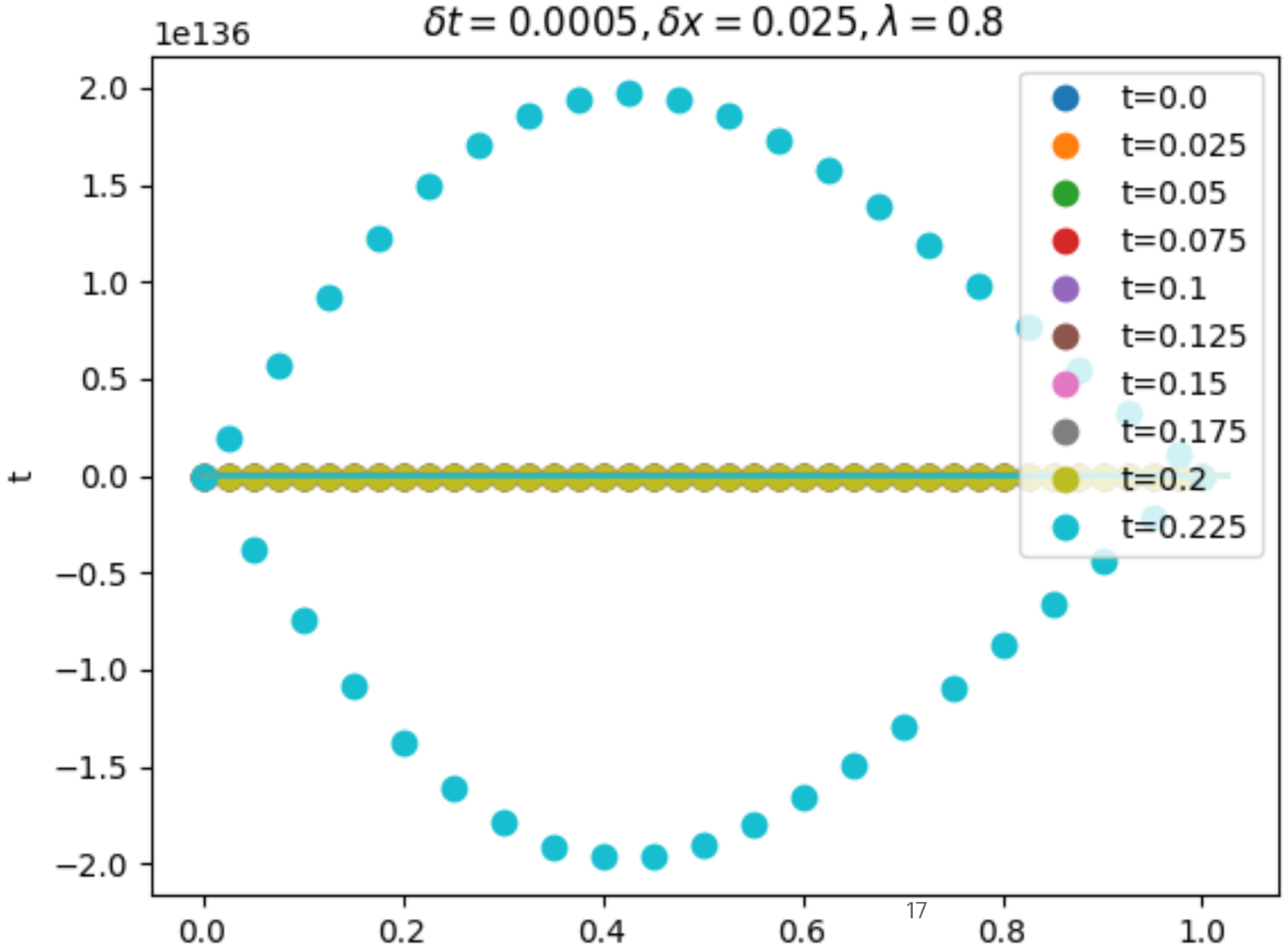
$$u_{i,j+1} = \lambda u_{i-1,j} + (1-2\lambda)u_{i,j} + \lambda u_{i+1,j}$$

$$\lambda = \frac{\Delta t}{\Delta x^2}$$

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2}$$

Does the method converge?

Boundary conditions: $u(x,0) = \sin(\pi x), u(0,t) = 0,\ u(1,t) = 0$,Solution: $u(x,t) = e^{-\pi^2 x} \sin(\pi x)$



Discrete Scheme:

$$u_{i,j+1} = \lambda u_{i-1,j} + (1 - 2\lambda)u_{i,j} + \lambda u_{i+1,j}$$

$$\lambda = \frac{\Delta t}{\Delta x^2}$$

What happened?

In numerical analysis of PDEs stability and convergence are equivalent (Lax equivalence theorem)

Stability of PDEs is beyond the scope of this course (ask for references)

A stable method must not increase errors!

Error Propagation in Explicit Finite Difference

A stable method must not increase errors! i.e.

Vector form: $u^{j+1} = Au^j$

$$u^j = \begin{bmatrix} u_{1,j} \\ u_{2,j} \\ \vdots \\ u_{m-1,j} \end{bmatrix}, u_s(j) = \begin{bmatrix} u(\Delta x, j\Delta t) \\ u(2\Delta x, j\Delta t) \\ \vdots \\ u((m-1)\Delta x, j\Delta t) \end{bmatrix}, A = \begin{pmatrix} 1-2\lambda & \lambda & 0 & \cdots & 0 \\ \lambda & 1-2\lambda & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \lambda \\ 0 & \cdots & 0 & \lambda & 1-2\lambda \end{pmatrix},$$

Where u^j is finite difference approximation and $u_s(j)$ is solution with infinite precision

Error: $e^j = u^j - u(j\Delta t)$ $\quad Ae^j = A(u^j - u(j\Delta t)) = e^{j+1}$ i.e. $e^j = A^j e^0$

A stable method must not increase errors, means:

$$\lim_{j\to\infty} A^j e^0 = 0 \iff \rho(A) < 1 \iff \lim_{j\to\infty} A^j[k,l] = 0 \iff \lambda \le \frac{1}{2}$$

Courant–Friedrichs–Lewy condition (CFL)

Backward Finite Difference Scheme

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} \longrightarrow \frac{\partial u}{\partial t} \approx \frac{u_{i,j} - u_{i,j-1}}{\Delta t}$$

(Backward Difference)

$$\frac{\partial^2 u}{\partial x^2} \approx \frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{\Delta x^2}$$

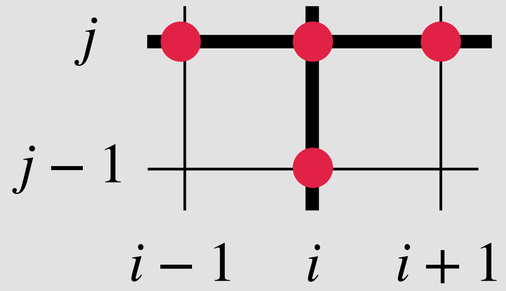
(Central Difference)

$$\frac{u_{i,j} - u_{i,j-1}}{\Delta t} = \frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{\Delta x^2}$$

$$u_{i,j-1} = -\lambda u_{i+1,j} + (1 + 2\lambda) u_{i,j} - \lambda u_{i-1,j}$$

$$\lambda = \frac{\Delta t}{\Delta x^2}$$

Backward Finite Difference **Stencil**

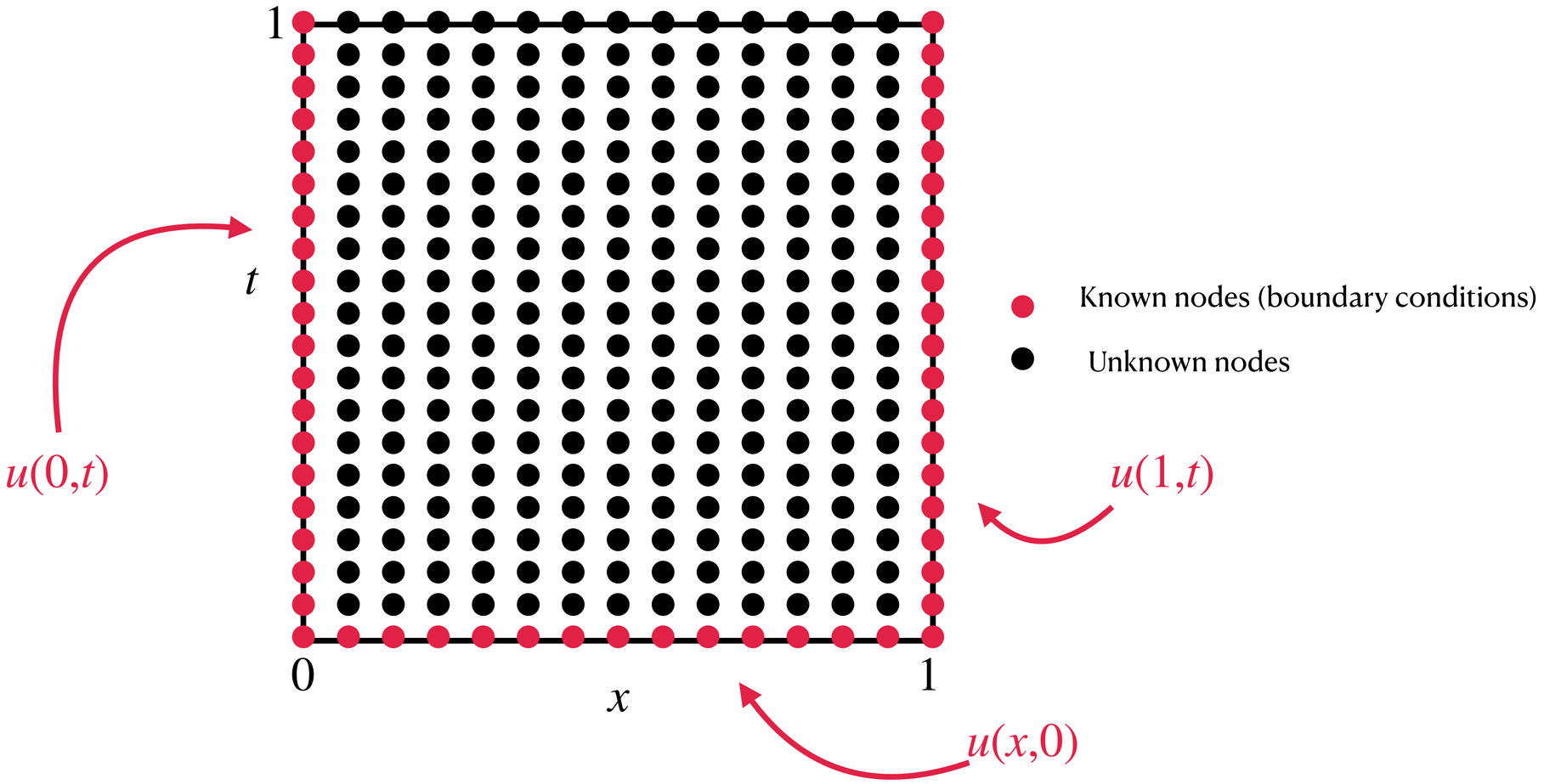


Example

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2}$$

Boundary conditions: $u(x,0) = \sin(\pi x), u(0,t) = 0, u(1,t) = 0$

Exact Solition: $u(x,t) = e^{-\pi^2 x} \sin(\pi x)$

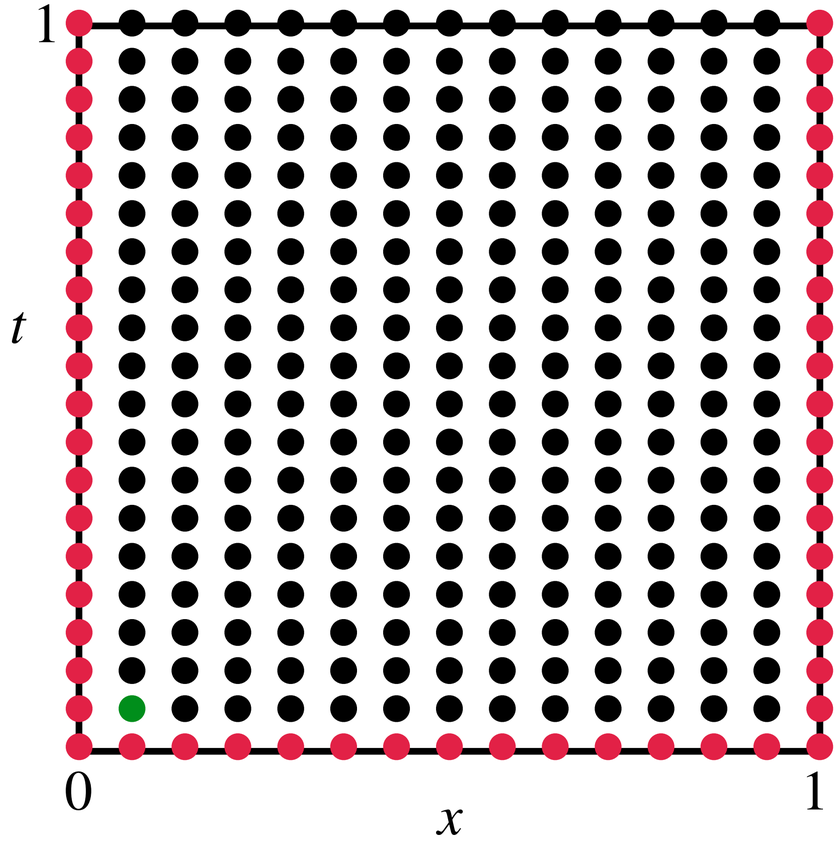


Example

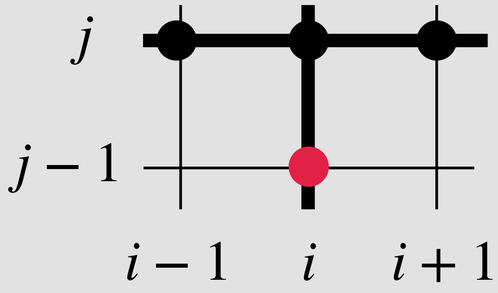
$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2}$$

Boundary conditions: $u(x,0) = \sin(\pi x), u(0,t) = 0$

Exact Solition: $u(x,t) = e^{-\pi^2 x} \sin(\pi x)$



Implicit Finite Difference **Stencil**



- Known nodes (boundary conditions)
- Unknown nodes
- $u(1,1) = ??$

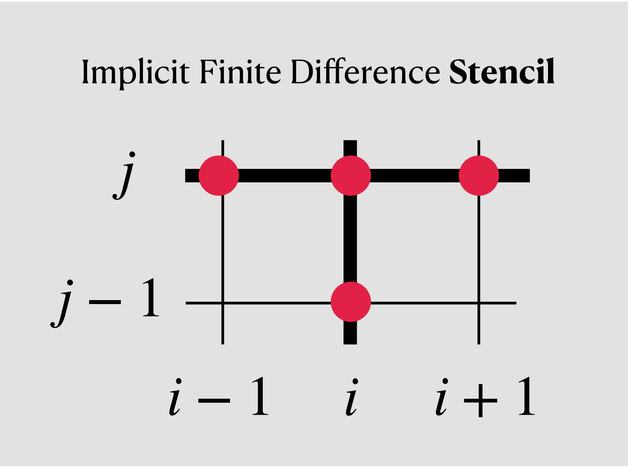
no closed form expression!

Backward finite difference is an example of an **implicit** finite difference scheme

Backward Finite Difference Scheme

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

$$u^j = \begin{bmatrix} u_{1,j} \\ u_{2,j} \\ \vdots \\ u_{m-1,j} \end{bmatrix}, A = \begin{pmatrix} 1+2\lambda & -\lambda & 0 & \cdots & 0 \\ -\lambda & 1+2\lambda & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & -\lambda \\ 0 & \cdots & 0 & -\lambda & 1+2\lambda \end{pmatrix},$$



Solution must be obtained by solving a (tri-diagonal) linear system of equations:

$Au^{j+1} = u^j$ **efficient algorithms exist for such sparse matrices**

It can be shown that this scheme is unconditionally stable i.e
it converges for all $\Delta x > 0,\ \Delta t > 0$

Some textbooks call this the implicit scheme, but there are many implicit schemes!

Backward Time Cantered Space (BTCS) is the exact name of the method

The Crank-Nicolson Scheme

Combine the implicit and explicit schemes for improved accuracy

$$\frac{u_{i,j+1} - u_{i,j}}{\Delta t} = \frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{\Delta x^2}$$

Forward j (explicit scheme)

$$\frac{u_{i,j+1} - u_{i,j}}{\Delta t} = \frac{u_{i+1,j+1} - 2u_{i,j+1} + u_{i-1,j+1}}{\Delta x^2}$$

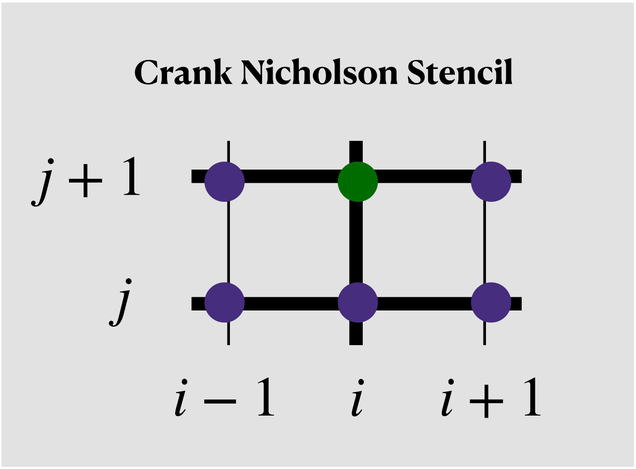
Backward j+1 (implicit scheme)

Adding the two:

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

The Crank-Nicolson Scheme

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

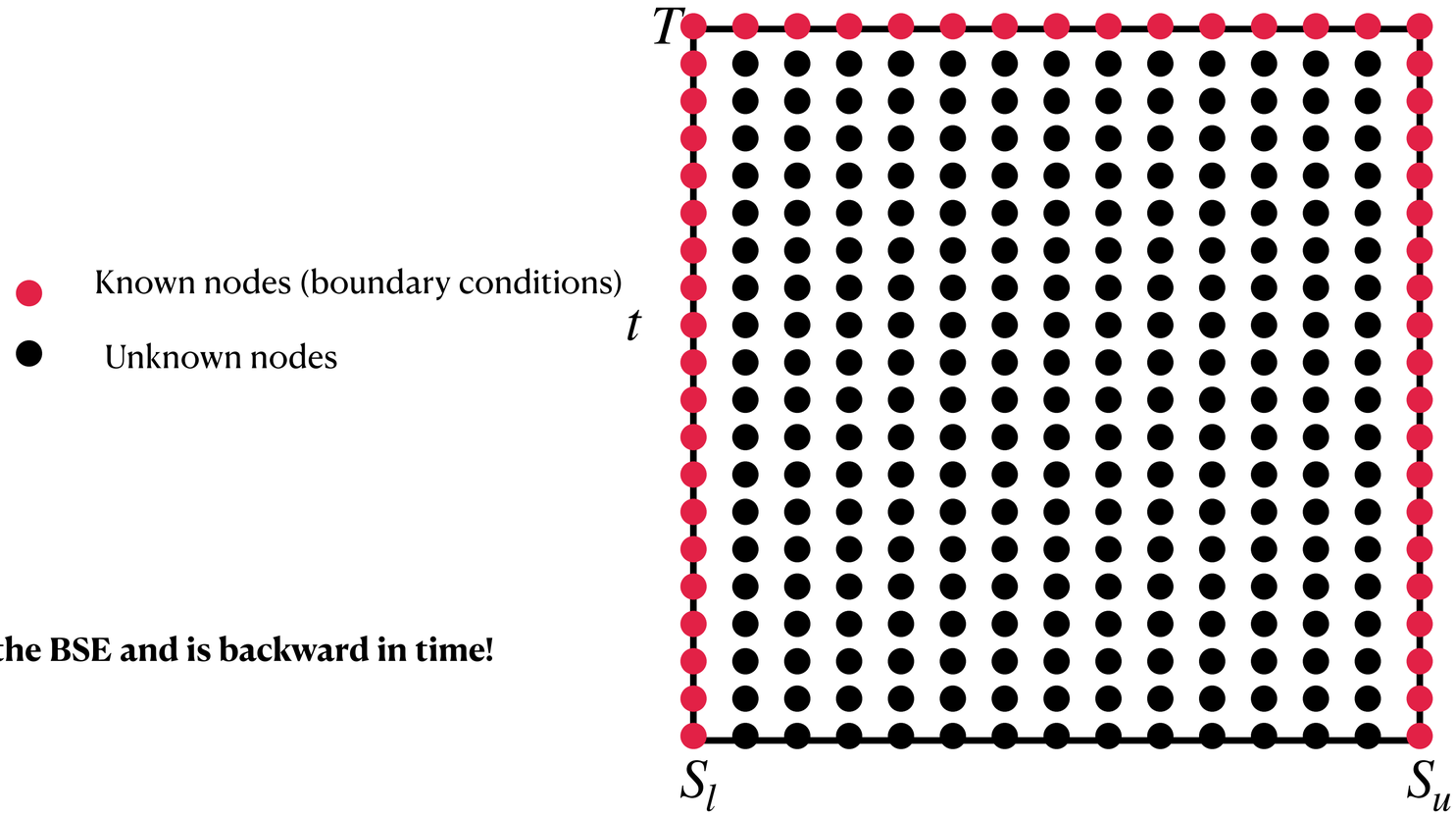


- Stability holds for all $\Delta t > 0$
- The convergence order is $O(\Delta t^2) + O(\Delta x^2)$ which is much better than methods studied so far
- Implicit scheme that requires solution of a linear system of equations

Boundary Conditions

- Finite difference methods need a grid with lower and upper bounds
- Not all boundary conditions will be in contract
- Product knowladge is required here, there are no unique solutions!
- It is easier to apply the boundary conditions on the BSE and then convert it to the heat equation

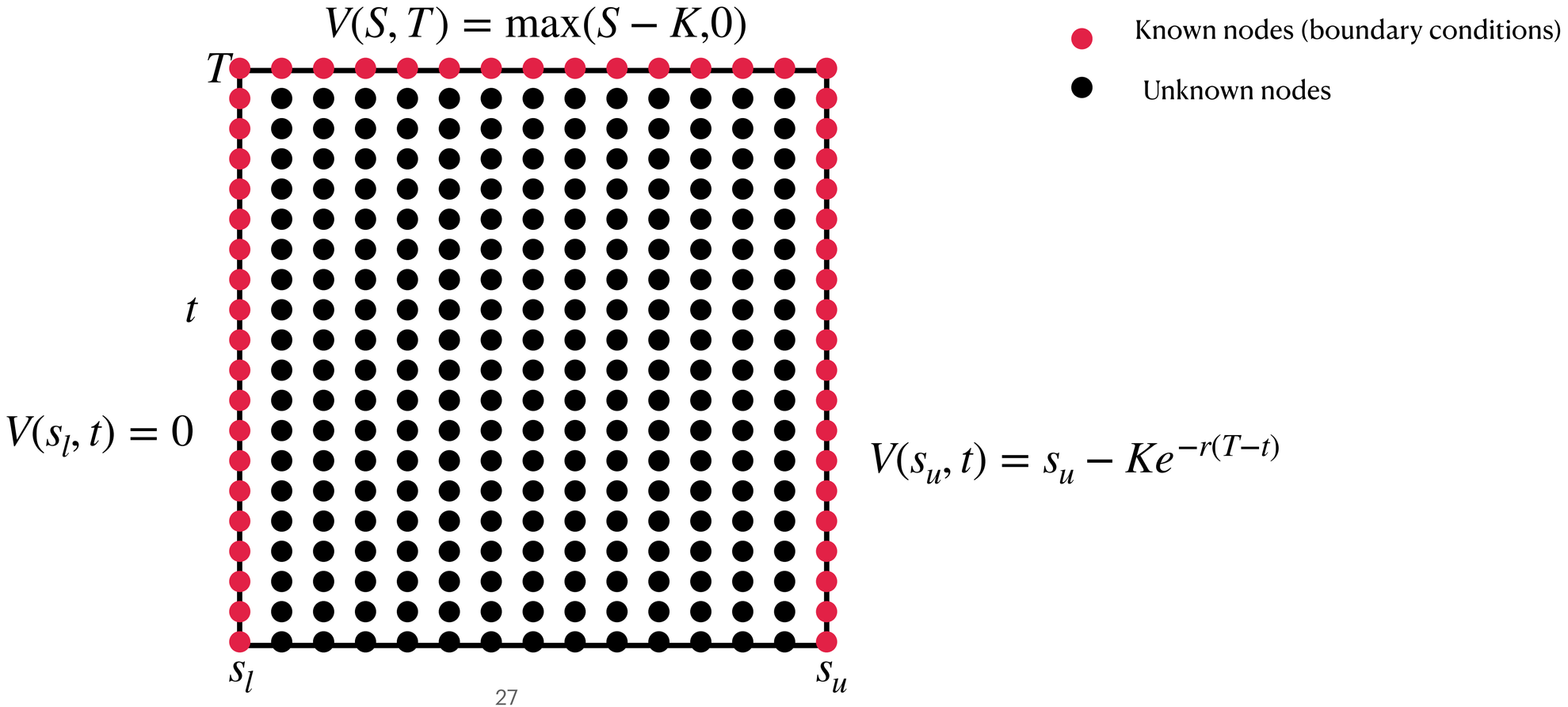
$$S_l \leq S \leq S_u$$
$$0 \leq t \leq T$$



Note: The grid on this slide is for the BSE and is backward in time!

Example: Boundary Conditions Euro Call

- Boundary conditions at expiry are easy: $V(S,T) = \max(S-K,0)$
- If stock is too low i.e. $S \to 0$ then option is wortless: $V(s_l,t) \to 0$
- If stock is too higk i.e. $S \to \infty$ then option will be exercsised: $V(s_u,t) = s_u - Ke^{-r(T-t)}$



Finite Difference vs Monte Carlo

- As the dimension increases the FD method suffers from the curse of dimensionality

  d dimensions $\implies$ grid of size Δx along each dimension $\implies$ error $\approx (\Delta x)^2$

  With a budget of N FD error is $\approx \frac{1}{N^{2d}} = N^{-2d}$

- With a budget of N Monte Carlo error is $\approx 1/\sqrt{N} = N^{-\frac{1}{2}}$ independendant of dimension

Summary: *

* This is valid for the methods we encountered in this class, reality can be more complicated but this is a good rule of thumb

Finite Difference vs Monte Carlo

- As the dimension increases the FD method suffers from the curse of dimensionality

  d dimensions $\implies$ grid of size Δx along each dimension $\implies$ error $\approx (\Delta x)^2$

  With a budget of N FD error is $\approx \frac{1}{N^{2d}} = N^{-2d}$

- With a budget of N Monte Carlo error is $\approx 1/\sqrt{N} = N^{-\frac{1}{2}}$ independendant of dimension

Summary: *

For $d = 1$ FD is better

For $d = 2$ methods are the same

For $d > 2$ MC is better

* This is valid for the methods we encountered in this class, reality can be more complicated but this is a good rule of thumb