Numerical Method for Solving the Advection-Diffusion Equation

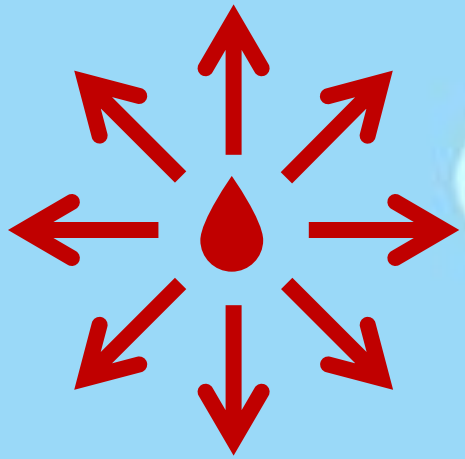
Kimberlyn Eversman
April 25th, 2024

THE UNIVERSITY OF TENNESSEE KNOXVILLE

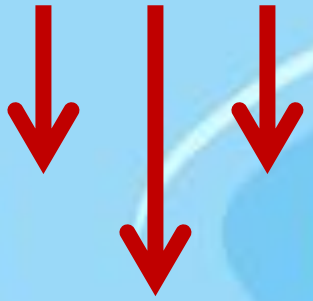
Outline

- Advection-Diffusion Equation
- Finite Difference Method
- Solving the Advection Diffusion Equation Numerically
- Programming our Scheme

Diffusion



Advection



Advection-Diffusion Equation

$$\frac{\partial u(\mathbf{x},t)}{\partial t} = \underbrace{-\nabla\cdot(\mathbf{v}u(\mathbf{x},t))}_{\text{Advection}} + \underbrace{\nabla\cdot(D\nabla u(\mathbf{x},t))}_{\text{Diffusion}} + R$$

- $u(\mathbf{x},t)$: variable of interest (concentration of dye)
- $\mathbf{v}(\mathbf{x},t)$: velocity
- $D(\mathbf{x},t)$: diffusion matrix
- $R(\mathbf{x},t)$: describes sources or sinks

Advection-Diffusion Equation

$$\frac{\partial u(\mathbf{x},t)}{\partial t} = \underbrace{-\nabla \cdot (\mathbf{v}u(\mathbf{x},t))}_{\text{Advection}} + \underbrace{\nabla \cdot (D\nabla u(\mathbf{x},t))}_{\text{Diffusion}} + R$$

First, let's make a few simplifying assumptions to make this PDE easier to work with for now…

Advection-Diffusion Equation in 1D Space

$$\frac{\partial u(x,t)}{\partial t} = \underbrace{-v\frac{\partial u(x,t)}{\partial x}}_{\text{Advection}} + \underbrace{D\frac{\partial^2 u(x,t)}{\partial x^2}}_{\text{Diffusion}} + R$$

- $u(\mathrm{x},t)$: variable of interest (concentration of dye)
- $v(\mathrm{x},t)$: velocity
- $D(\mathrm{x},t)$: diffusion
- $R(\mathrm{x},t)$: describes sources or sinks

Advection-Diffusion Equation in 1D Space

$$\frac{\partial u(x,t)}{\partial t} = \underbrace{-v\frac{\partial u(x,t)}{\partial x}}_{\text{Advection}} + \underbrace{D\frac{\partial^2 u(x,t)}{\partial x^2}}_{\text{Diffusion}}$$

- $u(\mathrm{x},t)$: variable of interest (concentration of dye)
- $v(\mathrm{x},t)$: velocity
- $D(\mathrm{x},t)$: diffusion

Advection-Diffusion Equation in 1D Space

$$\frac{\partial u(x,t)}{\partial t} = \underbrace{-v\frac{\partial u(x,t)}{\partial x}}_{\text{Advection}} + \underbrace{D\frac{\partial^2 u(x,t)}{\partial x^2}}_{\text{Diffusion}}$$

- $u(\mathrm{x}, t)$: variable of interest (concentration of dye)
- v: velocity
- D: diffusion

Advection-Diffusion Equation in 1D Space

$$\frac{\partial u(x,t)}{\partial t} = -v\frac{\partial u(x,t)}{\partial x} + D\frac{\partial^2 u(x,t)}{\partial x^2}$$

- PDEs like this can be hard to solve analytically (solve on paper).
- Instead we will *approximate* the solution (using a computer)!

Finite Difference Methods (FDMs)

A class of numerical techniques for solving differential equations by approximating derivatives with finite differences.

The main idea behind FDMs is to replace the derivatives by differences of values at nearby points.

$$\frac{\partial u(x,t)}{\partial x} = \lim_{h \to 0} \frac{u(x+h,t) - u(x,t)}{h}$$

For $h > 0$ and sufficiently small,

$$\frac{\partial u(x,t)}{\partial x} \approx \frac{u(x+h,t) - u(x,t)}{h}$$

This is called a *finite difference approximation*.

Finite Difference Operators

First Derivative Approximations:

- Forward Difference Operator

$$\delta_h^x u(x,t) = \frac{u(x+h,t) - u(x,t)}{h}$$

- Backward Difference Operator

$$\bar{\delta}_h^x u(x,t) = \frac{u(x,t) - u(x-h,t)}{h}$$

- Centered Difference Operator

$$\mathring{\delta}_h^x u(x,t) = \frac{u(x+h,t) - u(x-h,t)}{2h}$$

Second Derivative Approximations:

- Discrete Laplace Operator

$$\begin{aligned}\Delta_h^x u(x,t) &= \bar{\delta}_h^x \delta_h^x u(x,t) \\ &= \frac{u(x+h,t) - 2u(x,t) + u(x-h,t)}{2h}\end{aligned}$$

Finite Difference Method

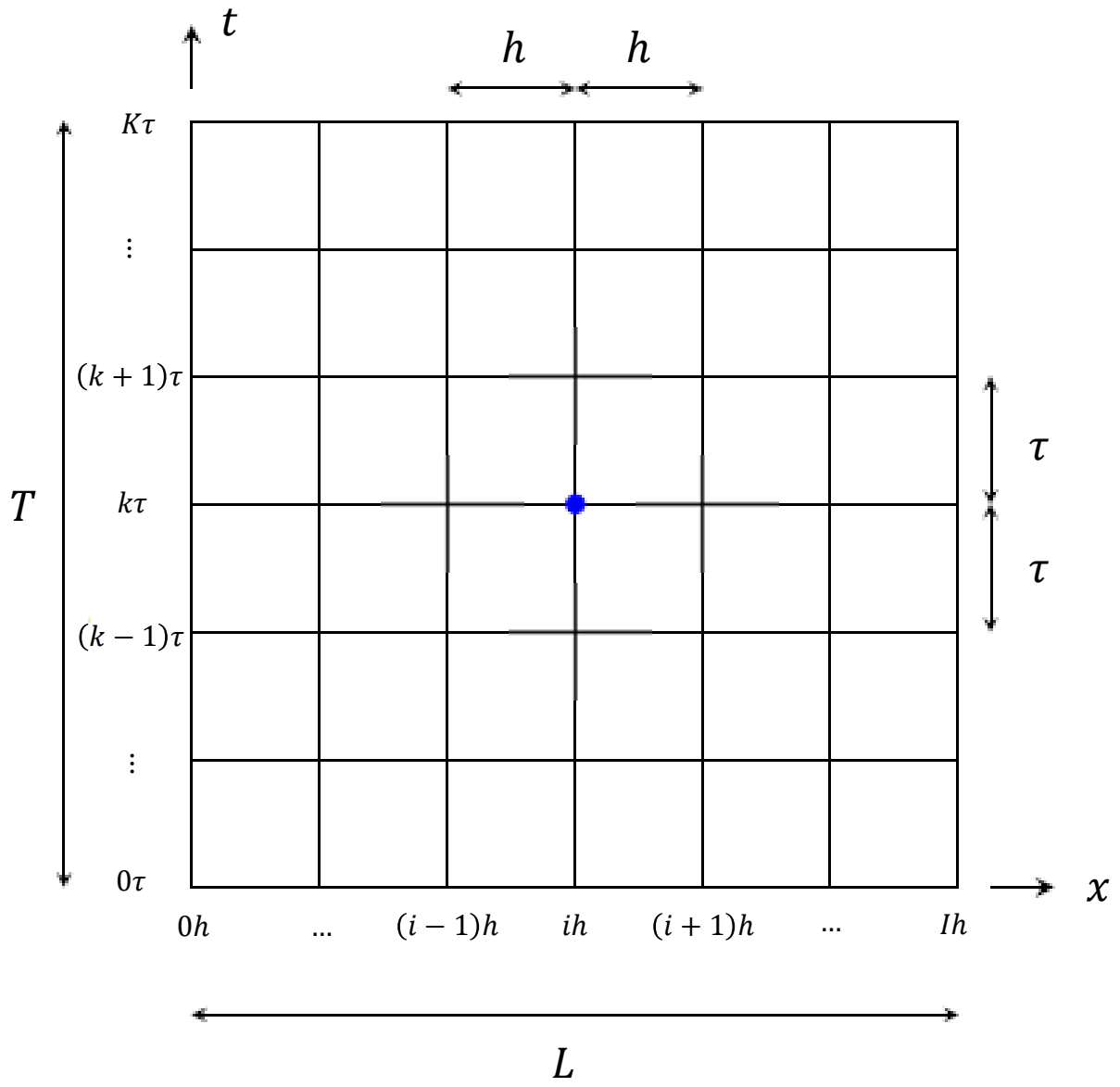
We will first need to discretize our domains.

- Let $x \in [0, \mathrm{L}], \mathrm{t} \in [0, T]$
- For $I, \mathrm{K} \in \mathbb{N}$, define our step sizes $h = {}^{L}/_{I}, \tau = {}^{T}/_{K}$

Define w_i^k to be the approximation of $u(x,t)$ such that

$$w_i^k \approx u(ih, k\tau)$$

for $i = 0, \ldots, I$ and $\mathrm{k} = 0, \ldots, K$.



Stability, Consistency, and Convergence

- We want our FDM to be stable, consistent, and to converge…

Stability, Consistency, and Convergence

- We don’t want growth of round-off errors and/or small fluctuations in initial data to produce wildly different solutions to our FDM.

- FDMs that can be proven not to magnify approximation errors are called *stable.*

- FDM can also be *conditionally or unconditionally stable.*
  - Conditionally stable means it is only stable under certain constraints on the step sizes.
  - Our scheme today will be conditionally stable.

Stability, **Consistency**, and Convergence

- *Consistency* refers to a qualitative measure of the extent to which the actual solution satisfies the FDM solution.

- i.e. if our approximation looks different than what we know our actual solution should be, the solution is inconsistent

Stability, Consistency, and **Convergence**

- If our FDM solution is getting closer to the actual solution as our step sizes get smaller, then the scheme *converges*.

- Lax's Principle:

$$\text{consistency} + \text{stability} \Rightarrow \text{convergence}$$

Our Boundary Value Problem (BVP)

$$\begin{cases} \dfrac{\partial u(x,t)}{\partial t} + v\dfrac{\partial u(x,t)}{\partial x} - D\dfrac{\partial^2 u(x,t)}{\partial x^2} = 0, & x \in (0,L), t \in (0,T] \\ u(0,t) = u(L,t) = 0, & t \in (0,T] \\ u(x,0) = u_0(x), & x \in (0,L) \end{cases}$$

Assume $u_0(0) = u_0(L) = 0$.

Now let's use FDM to approximate this solution!

Numerically Solving the BVP

Recall w_i^k approximates $u(ih, k\tau)$ for $i = 0, \dots, I$ and $k = 0, \dots, K$.
We're going to be working through the time steps $k = 0, \dots, K$.

Our initial value function is

$$u_0(x) = u(x, 0).$$

Thus define

$$w_i^0 = u_0(ih) \text{ for } i = 0, \dots, I.$$

Then working through the time steps:

- Find w_i^1 for $i = 0, \dots, I$.
- Find w_i^2 for $i = 0, \dots, I$.
- …
- Finally, find w_i^K for $i = 0, \dots, I$.

Numerically Solving the BVP

To find w_i^k for $\mathrm{k} = 1, \ldots, \mathrm{K}$ use

To find w_i^k for $\mathrm{k} = 1, \ldots, \mathrm{K}$ use

Using our finite difference operators,

Using our finite difference operators,

$$\frac{u(x,t+\tau)-u(x,t)}{\tau} + v\frac{u(x+h,t)-u(x-h,t)}{2h} - D\frac{u(x+h,t)-2u(x,t)+u(x-h,t)}{h^2} = 0$$

Using w_i^k to replace $u(x,t)$,

$$\frac{w_i^{k+1}-w_i^k}{\tau} + v\frac{w_{i+1}^k-w_{i-1}^k}{2h} - D\frac{w_{i+1}^k-2w_i^k+w_{i-1}^k}{h^2} = 0$$

Numerically Solving the BVP

$$\frac{w_i^{k+1} - w_i^k}{\tau} + v\frac{w_{i+1}^k - w_{i-1}^k}{2h} - D\frac{w_{i+1}^k - 2w_i^k + w_{i-1}^k}{h^2} = 0$$

$$\begin{aligned} w_i^{k+1} &= w_i^k - \frac{v\tau}{2h}\left[w_{i+1}^k - w_{i-1}^k\right] + \frac{D\tau}{h^2}\left[w_{i+1}^k - 2w_i^k + w_{i-1}^k\right] \\ &= \left[\frac{D\tau}{h^2} + \frac{\tau v}{2h}\right] w_{i-1}^k + \left[1 - \frac{2D\tau}{h^2}\right] w_i^k + \left[\frac{D\tau}{h^2} - \frac{\tau v}{2h}\right] w_{i+1}^k \\ &= a w_{i-1}^k + b w_i^k + c w_{i+1}^k \end{aligned}$$

where $a = \frac{D\tau}{h^2} + \frac{\tau v}{2h}$, b $= 1 - \frac{2D\tau}{h^2}$, c $= \frac{D\tau}{h^2} - \frac{\tau v}{2h}$.

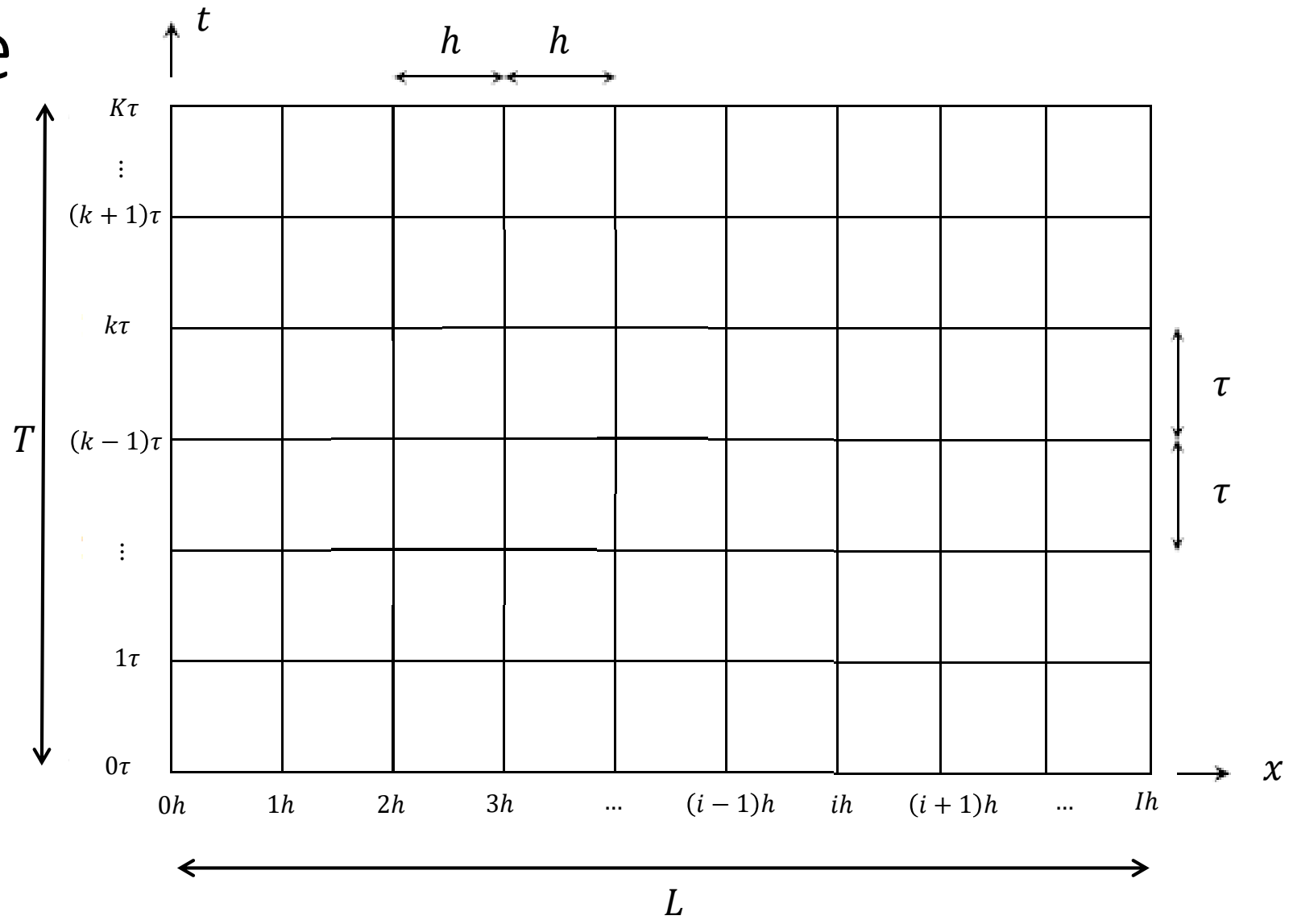
Finite Difference Method

$k = 0$:

$$w_i^0 = u_0(ih)$$

$k = 1, \dots, K$:

$$w_i^{k+1} = aw_{i-1}^k + bw_i^k + cw_{i+1}^k$$



Finite Difference Method

$k = 0$:

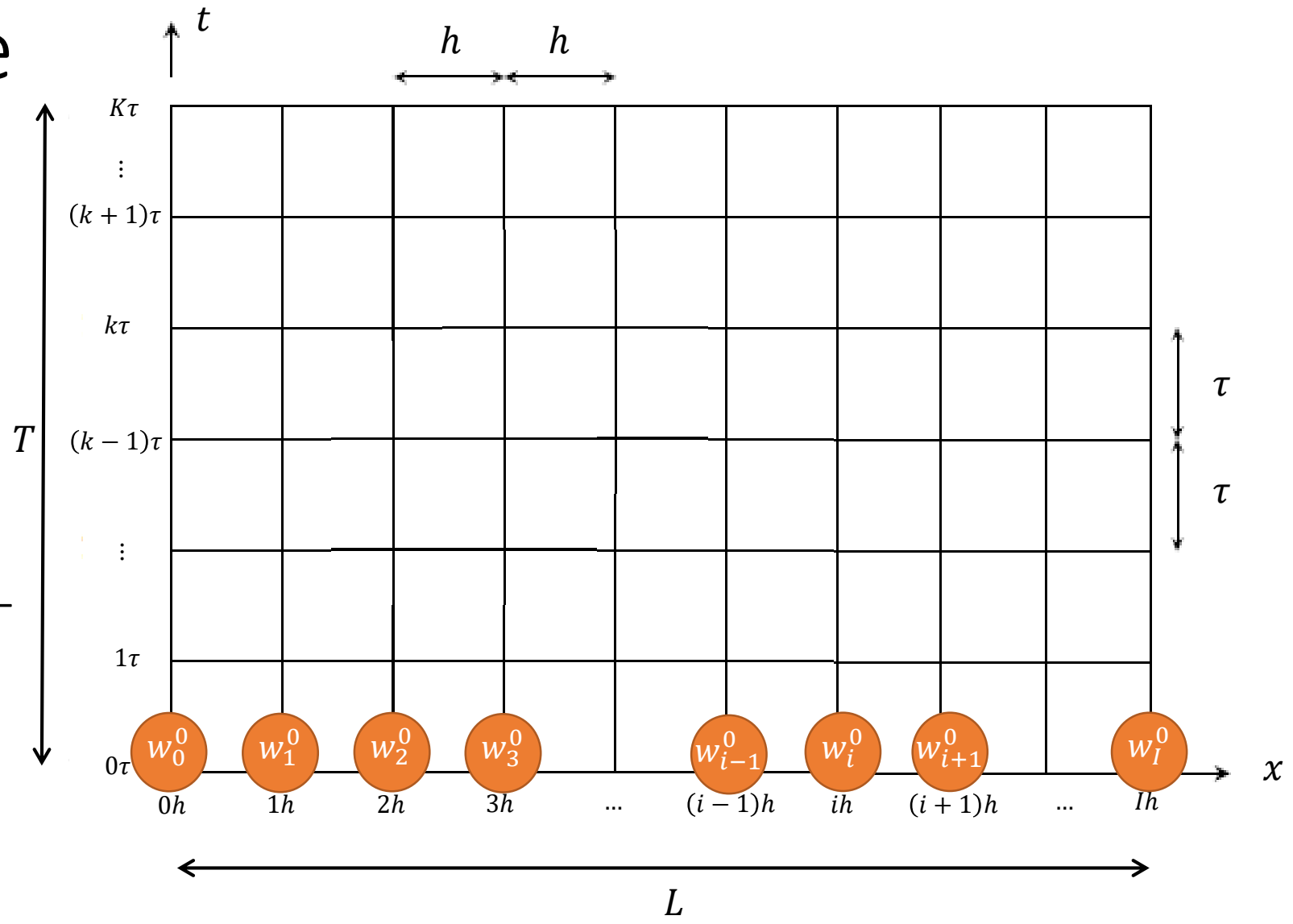
$$w_i^0 = u_0(ih)$$

$k = 1, \dots, K$:

$$w_i^{k+1} = aw_{i-1}^k + bw_i^k + cw_{i+1}^k$$

$k = 0, i = 0, \dots, \mathrm{I}$:

$$w_i^0 = u_0(ih)$$



Finite Difference Method

$k = 0$:

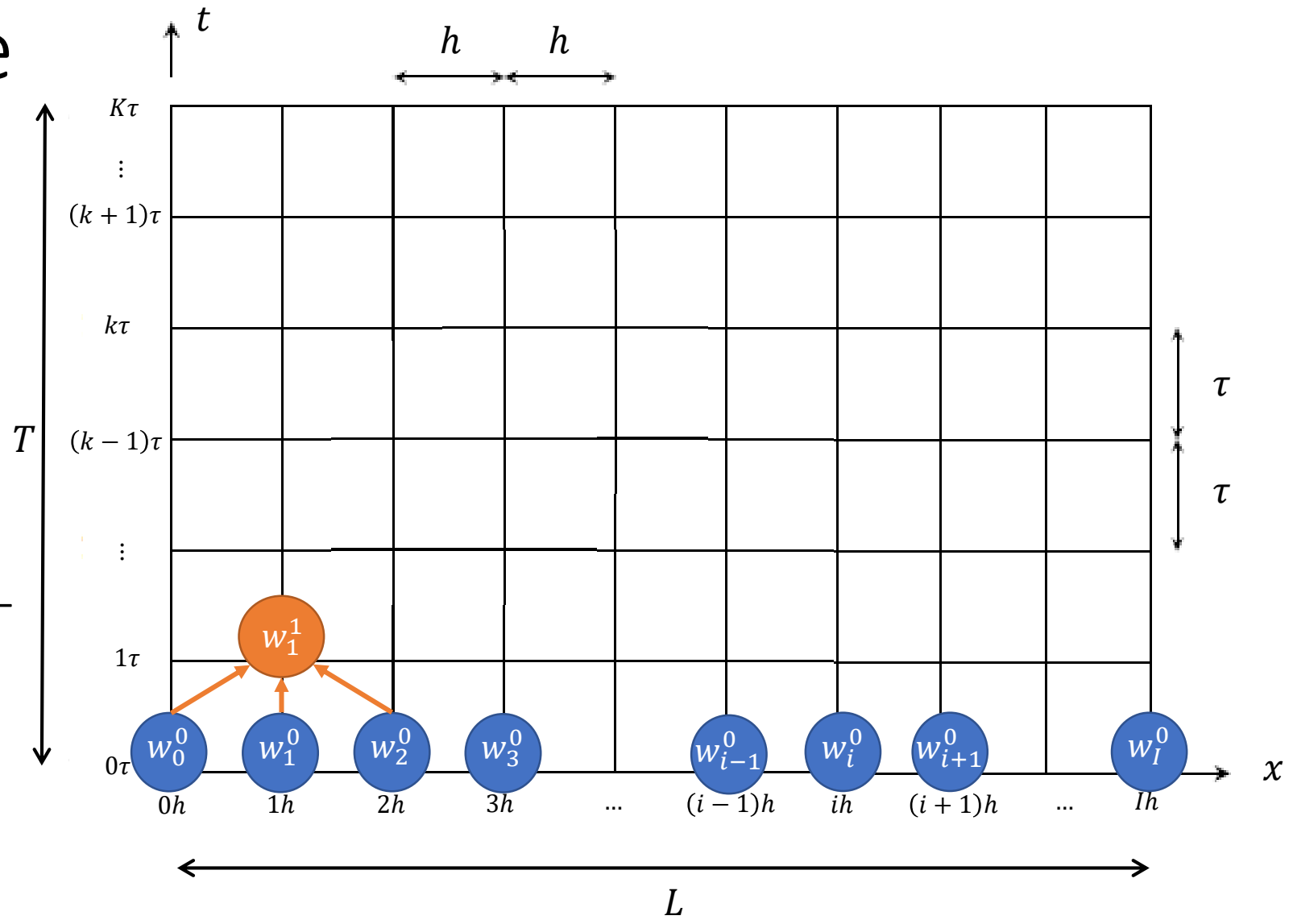
$$w_i^0 = u_0(ih)$$

$k = 1, \dots, K$:

$$w_i^{k+1} = aw_{i-1}^k + bw_i^k + cw_{i+1}^k$$

$k = 1, i = 1$:

$$w_1^1 = aw_0^0 + bw_1^0 + cw_2^0$$



Finite Difference Method

$k = 0$:

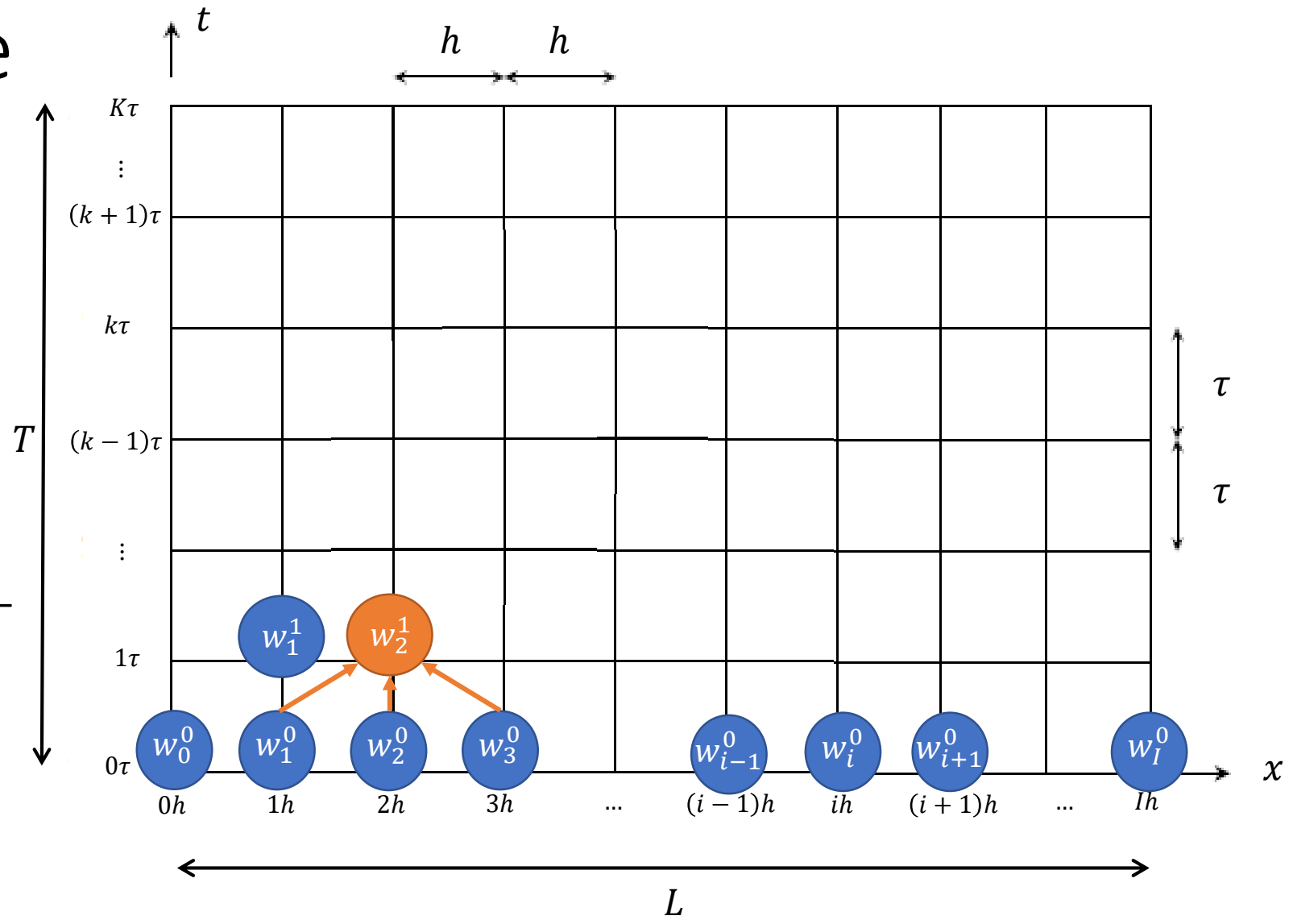
$$w_i^0 = u_0(ih)$$

$k = 1, \dots, K$:

$$w_i^{k+1} = aw_{i-1}^k + bw_i^k + cw_{i+1}^k$$

$k = 1, i = 2$:

$$w_2^1 = aw_1^0 + bw_2^0 + cw_3^0$$



Finite Difference Method

$k = 0$:

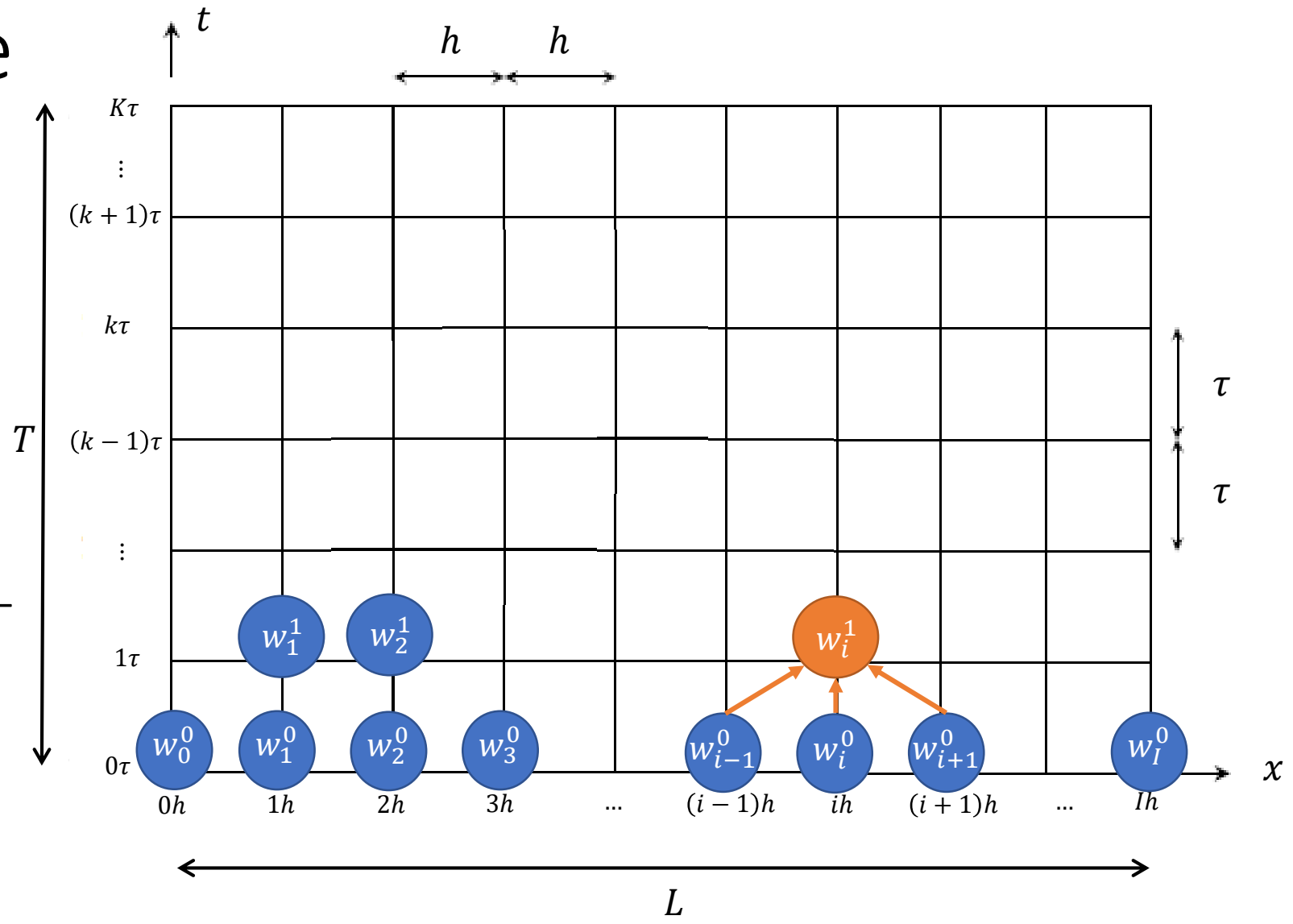
$$w_i^0 = u_0(ih)$$

$k = 1, \dots, K$:

$$w_i^{k+1} = aw_{i-1}^k + bw_i^k + cw_{i+1}^k$$

$k = 1, i = i$:

$$w_i^1 = aw_{i-1}^0 + bw_i^0 + cw_{i+1}^0$$



Finite Difference Method

$k = 0$:

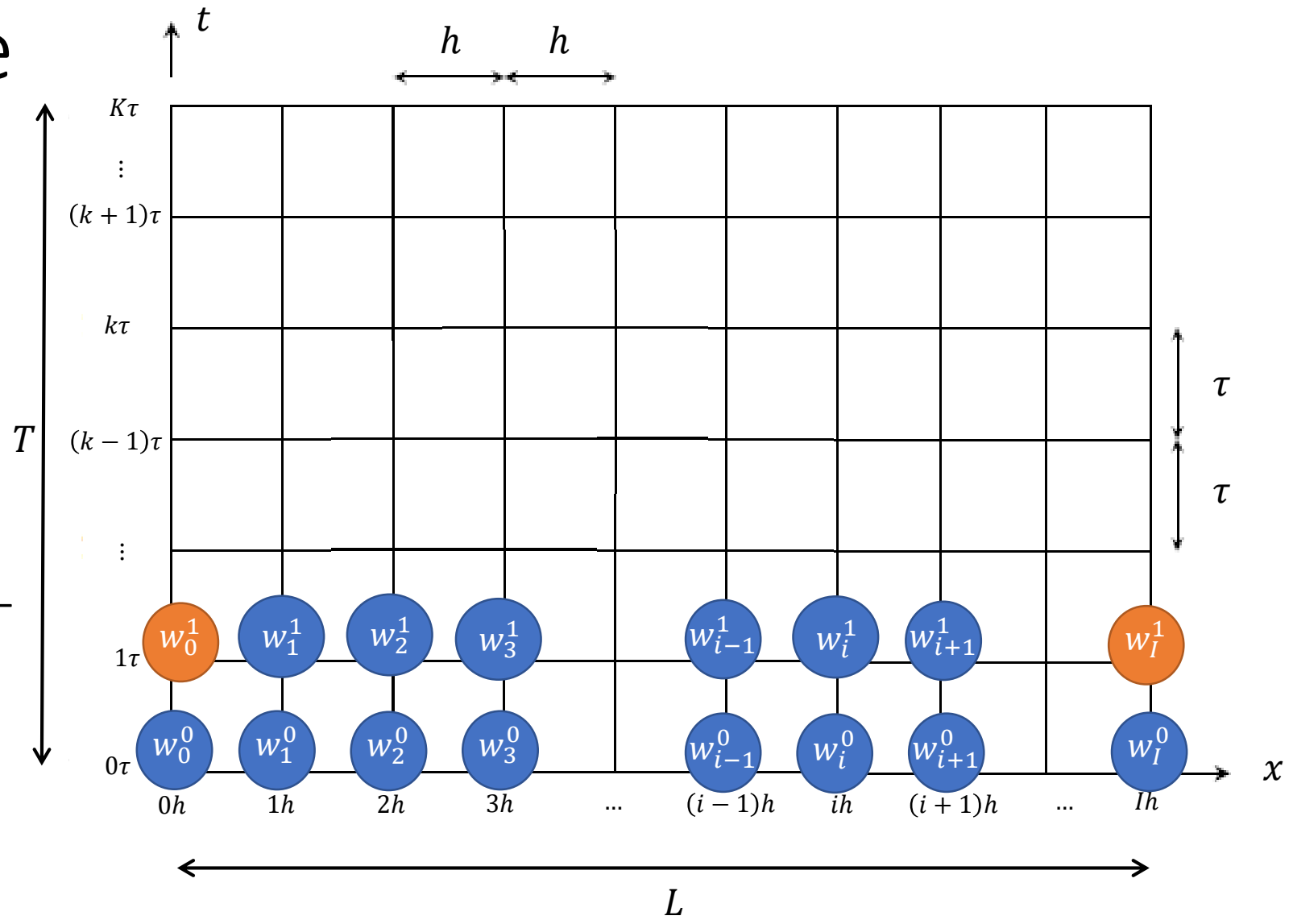
$$w_i^0 = u_0(ih)$$

$k = 1, \dots, K$:

$$w_i^{k+1} = aw_{i-1}^k + bw_i^k + cw_{i+1}^k$$

Recall that $u(0,t) = u(L,t) = 0$:

$w_0^1 = w_I^1 = 0$



Numerically Solving the BVP

Assume we know w_i^k for $i = 0, \dots, I$. Then using $w_i^{k+1} = aw_{i-1}^k + bw_i^k + cw_{i+1}^k$ find w_i^{k+1}.

$w_0^{k+1} = 0$, since $u(0,t) = 0$

$w_1^{k+1} = aw_0^k + bw_1^k + cw_2^k$

$w_2^{k+1} = \quad aw_1^k + bw_2^k + cw_3^k$

$w_3^{k+1} = \quad\quad aw_2^k + bw_3^k + cw_4^k$

$\vdots$

$w_{I-1}^{k+1} = \quad\quad\quad\quad\quad\quad\quad\quad aw_{I-2}^k + bw_{I-1}^k + cw_I^k$

$w_I^{k+1} = 0$, since $u(L,t) = 0$

Numerically Solving the BVP

Assume we know w_i^k for $i = 0, \dots, I$. Then using $w_i^{k+1} = aw_{i-1}^k + bw_i^k + cw_{i+1}^k$,

$w_0^{k+1} = 0$, since $u(0,t) = 0$

$$w_1^{k+1} = aw_0^k + bw_1^k + cw_2^k + 0w_3^k + 0w_4^k + \cdots + 0w_{I-2}^k + 0w_{I-1}^k + 0w_I^k$$

$$w_2^{k+1} = 0w_0^k + aw_1^k + bw_2^k + cw_3^k + 0w_4^k + \cdots + 0w_{I-2}^k + 0w_{I-1}^k + 0w_I^k$$

$$w_3^{k+1} = 0w_0^k + 0w_1^k + aw_2^k + bw_3^k + cw_4^k + \cdots + 0w_{I-2}^k + 0w_{I-1}^k + 0w_I^k$$

$$\dots$$

$$w_{I-1}^{k+1} = 0w_0^k + 0w_1^k + 0w_2^k + 0w_3^k + 0w_4^k + \cdots + aw_{I-2}^k + bw_{I-1}^k + cw_I^k$$

$w_I^{k+1} = 0$, since $u(L,t) = 0$

Numerically Solving the BVP

Assume we know w_i^k for $i = 0, \dots, I$. Then using $w_i^{k+1} = aw_{i-1}^k + bw_i^k + cw_{i+1}^k$,

$$w_0^{k+1} = 0w_0^k + 0w_1^k + 0w_2^k + 0w_3^k + 0w_4^k + \cdots + 0w_{I-2}^k + 0w_{I-1}^k + 0w_I^k$$

$$w_1^{k+1} = aw_0^k + bw_1^k + cw_2^k + 0w_3^k + 0w_4^k + \cdots + 0w_{I-2}^k + 0w_{I-1}^k + 0w_I^k$$

$$w_2^{k+1} = 0w_0^k + aw_1^k + bw_2^k + cw_3^k + 0w_4^k + \cdots + 0w_{I-2}^k + 0w_{I-1}^k + 0w_I^k$$

$$w_3^{k+1} = 0w_0^k + 0w_1^k + aw_2^k + bw_3^k + cw_4^k + \cdots + 0w_{I-2}^k + 0w_{I-1}^k + 0w_I^k$$

$$\dots$$

$$w_{I-1}^{k+1} = 0w_0^k + 0w_1^k + 0w_2^k + 0w_3^k + 0w_4^k + \cdots + aw_{I-2}^k + bw_{I-1}^k + cw_I^k$$

$$w_I^{k+1} = 0w_0^k + 0w_1^k + 0w_2^k + 0w_3^k + 0w_4^k + \cdots + 0w_{I-2}^k + 0w_{I-1}^k + 0w_I^k$$

Numerically Solving the BVP

$$\mathrm{A} \coloneqq \begin{bmatrix} 0 & 0 & 0 & 0 & \dots & 0 \\ a & b & c & 0 & \dots & 0 \\ 0 & a & b & c & \dots & 0 \\ 0 & \ddots & \ddots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & a & b & c \\ 0 & \dots & 0 & 0 & 0 & 0 \end{bmatrix} \qquad \vec{w}^k \coloneqq \begin{bmatrix} w_0^k \\ w_1^k \\ w_2^k \\ \vdots \\ w_{I-1}^k \\ w_I^k \end{bmatrix}$$

$$A\vec{w}^k = \begin{bmatrix} 0w_0^k + 0w_1^k + 0w_2^k + 0w_3^k + \cdots + 0w_{I-2}^k + 0w_{I-1}^k + 0w_I^k \\ aw_0^k + bw_1^k + cw_2^k + 0w_3^k + \cdots + 0w_{I-2}^k + 0w_{I-1}^k + 0w_I^k \\ 0w_0^k + aw_1^k + bw_2^k + cw_3^k + \cdots + 0w_{I-2}^k + 0w_{I-1}^k + 0w_I^k \\ \vdots \\ 0w_0^k + 0w_1^k + 0w_2^k + 0w_3^k + \cdots + aw_{I-2}^k + bw_{I-1}^k + cw_I^k \\ 0w_0^k + 0w_1^k + 0w_2^k + 0w_3^k + \cdots + 0w_{I-2}^k + 0w_{I-1}^k + 0w_I^k \end{bmatrix} = \begin{bmatrix} w_0^{k+1} \\ w_1^{k+1} \\ w_2^{k+1} \\ \vdots \\ w_{I-1}^{k+1} \\ w_I^{k+1} \end{bmatrix} = \vec{w}^{k+1}$$

Programming the Scheme

Step 1: Generate matrix $W \in \mathbb{R}^{(I+1)\times(K+1)}$ to store our solutions as such $W = [\vec{w}^0 \quad \vec{w}^1 \quad \dots \quad \vec{w}^K]$

Step 2: Calculate $\vec{w}^0$ using $w_i^0 = u_0(ih)$ for $i = 0, \dots, I$ (storing in W)

Step 3: Generate the tridiagonal matrix $A \in \mathbb{R}^{(I+1)\times(I+1)}$

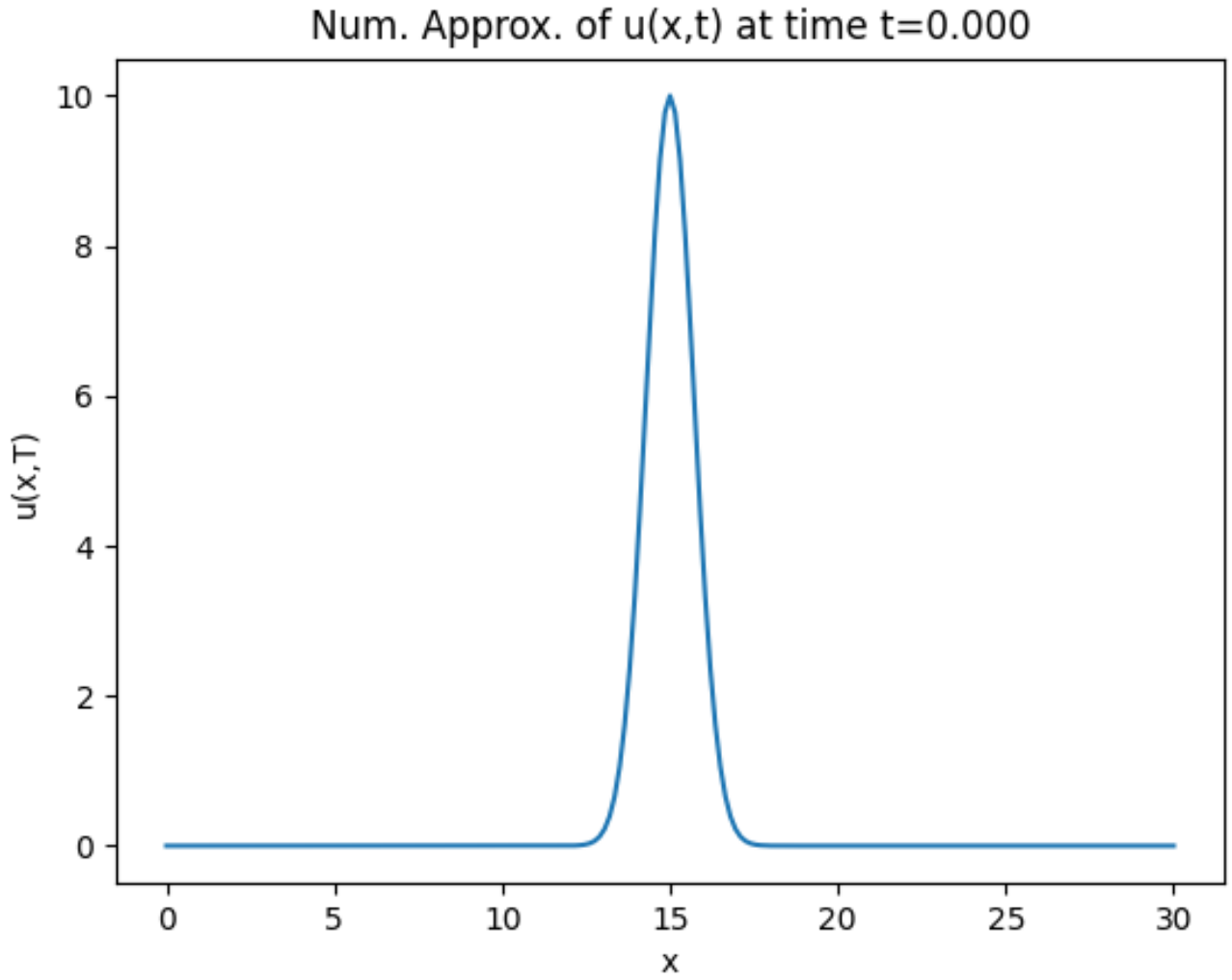
Step 4: For $k = 0, \dots, K-1$, calculate $\vec{w}^{k+1} = A\vec{w}^k$ (storing in W)

Step 5 (optional): Graph your solution using W

Running the Scheme

$L = 30$
$T = 2$
$v = 3$
$D = 0.5$
$I = 200$
$K = 300$
$u_0(x) = 10e^{-(x-L/2)^2}$

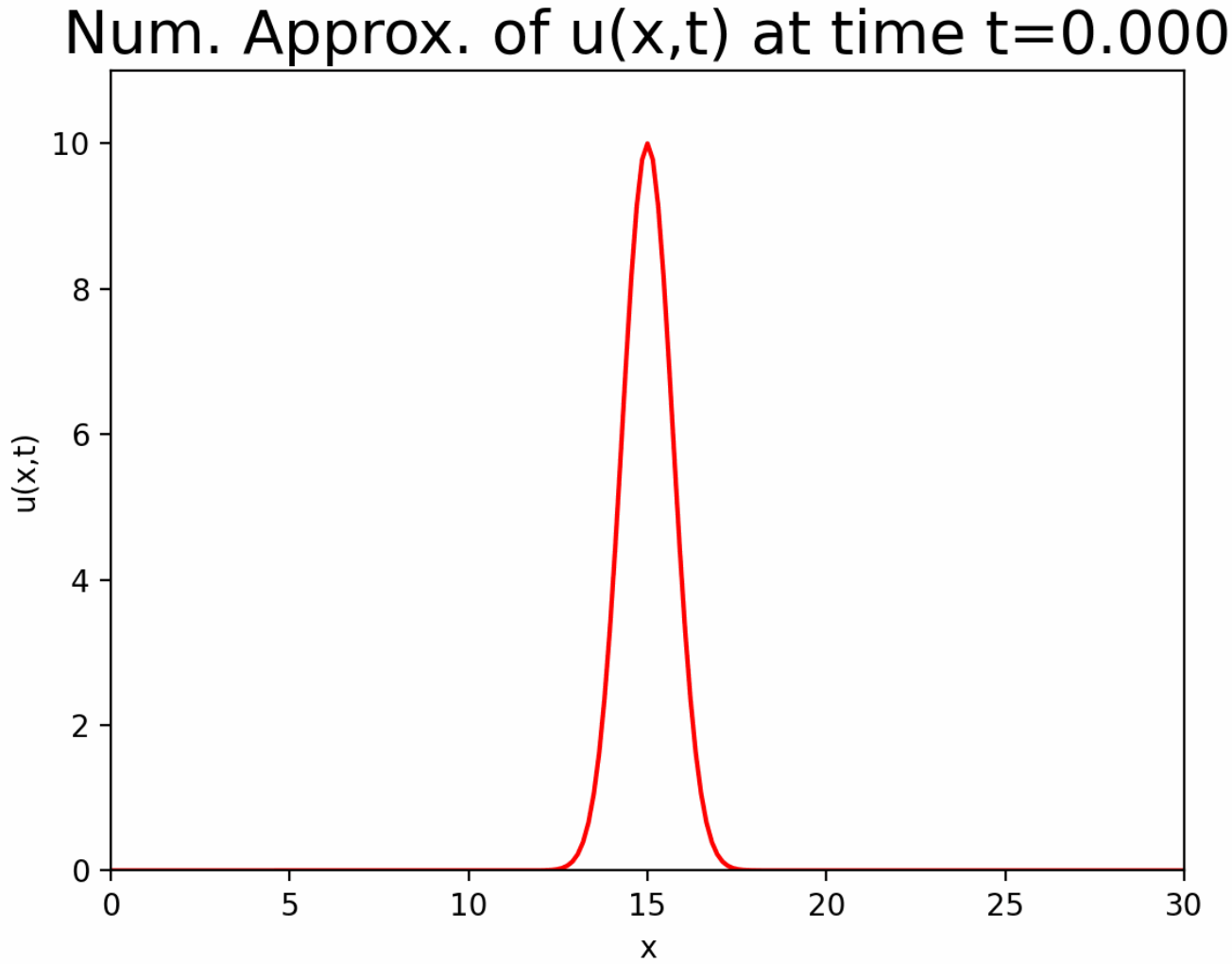
From what we know about the advection diffusion equation, what do we except to happen to our solution over time?



Running the Scheme

$L = 30$
$T = 2$
$v = 3$
$D = 0.5$
$I = 200$
$K = 300$

$u_0(x) = 10e^{-(x-L/2)^2}$



Conditional Stability

Recall that we said that this scheme is *conditionally stable*.

Stability conditions:

$$h < \frac{2D}{|v|}, \qquad \tau < \frac{h^2}{2D}$$

If you don't want to get into the theory/prove stability…

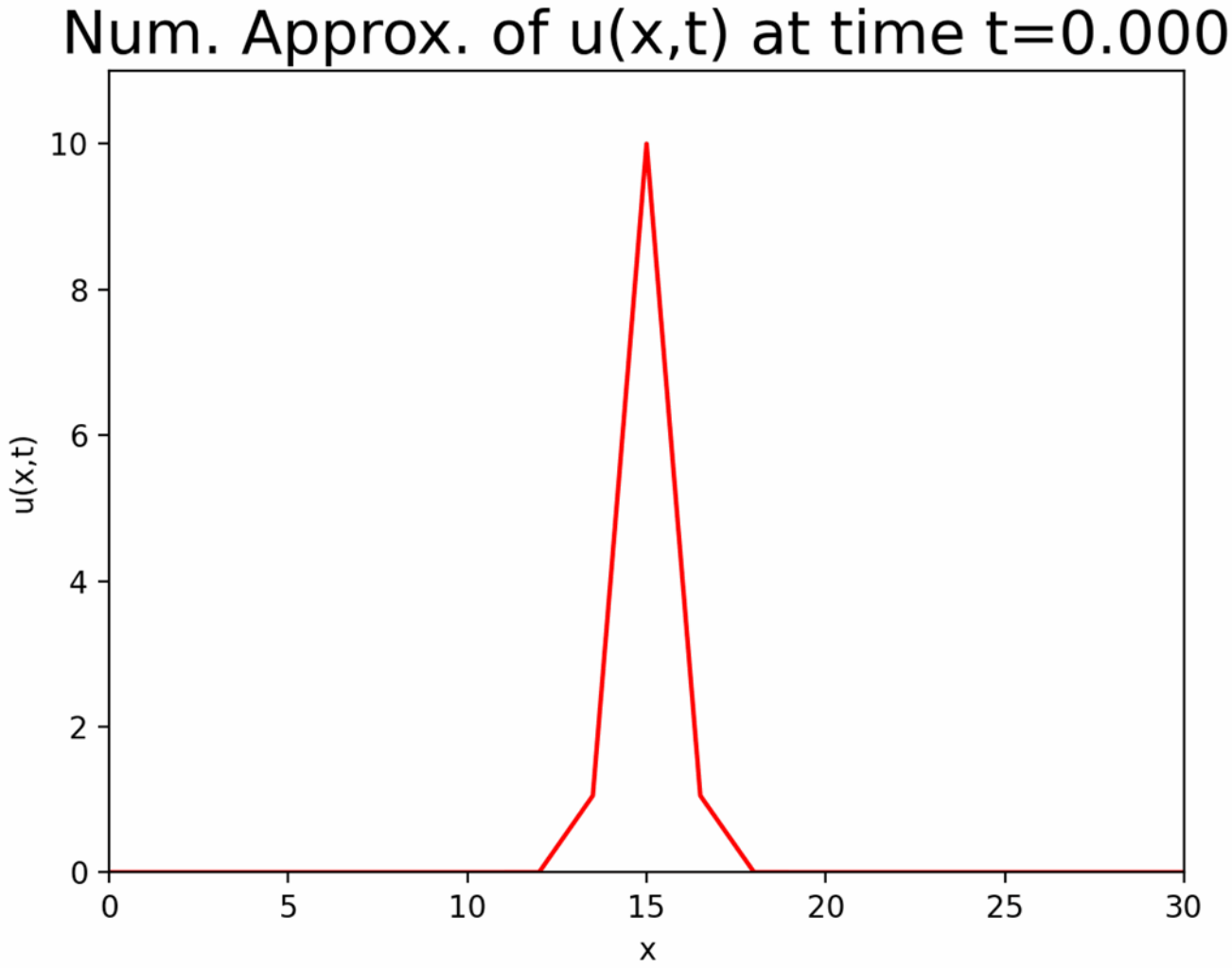
- For well known equations (like advection diffusion), you can sometimes find their stability conditions online.

What happens if these conditions aren't met?

Ruining the Scheme

What if our spatial step size, h, is too big?

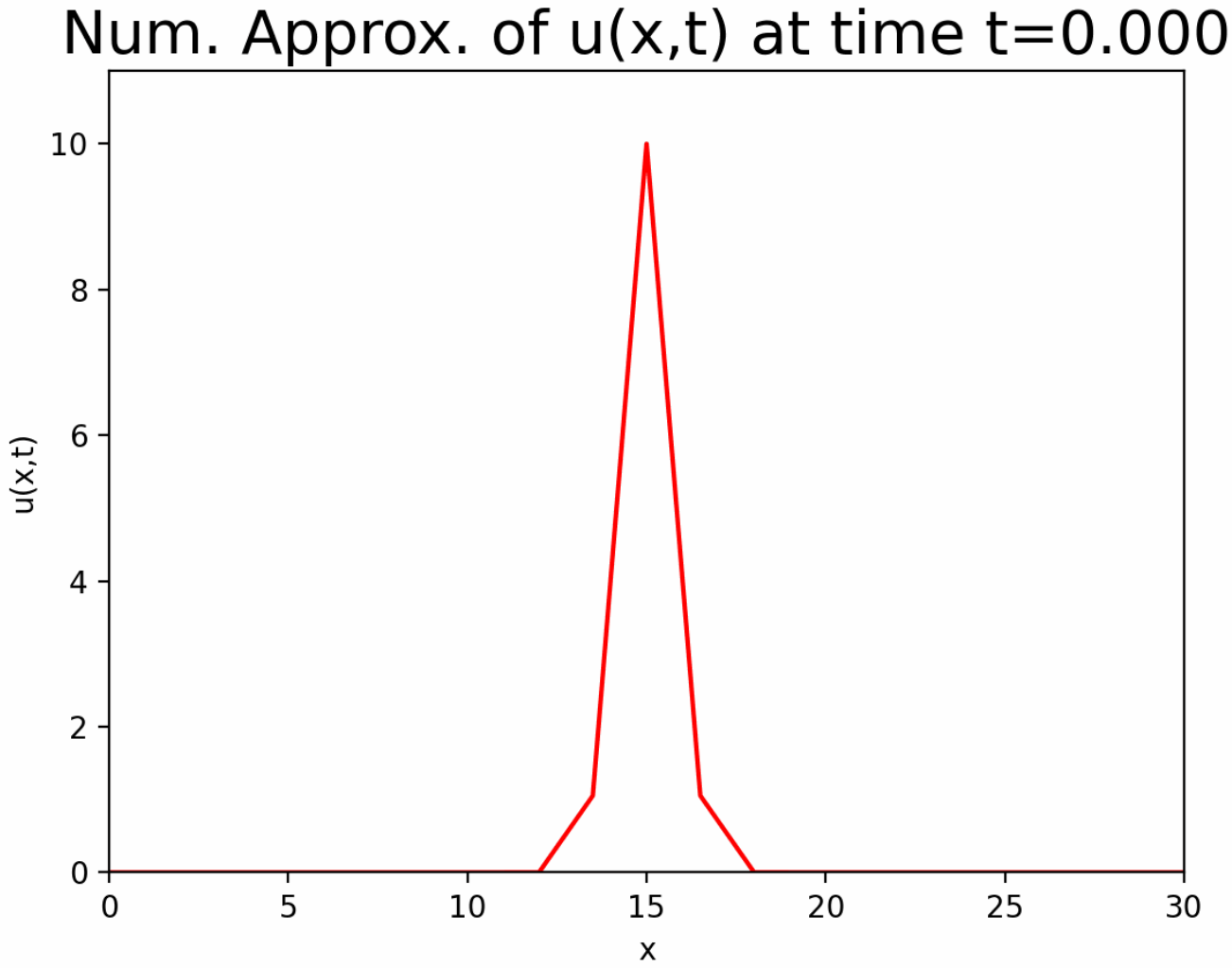
$L = 30$
$T = 2$
$D = 0.5$
$I = \cancel{200}\ 20$
$K = 300$
$v = 3$
$u_0(x) = 10e^{-(x-L/2)^2}$



Ruining the Scheme

What if our spatial step size, h, is too big?

$L = 30$
$T = 2$
$D = 0.5$
$I = \cancel{200}\ 20$
$K = 300$
$v = 3$
$u_0(x) = 10e^{-(x-L/2)^2}$



Time Dependent Velocity/Diffusion

Let's change our velocity from a constant to be dependent on time $v(t)$.

Recall when v was constant

$$\mathrm{A} \coloneqq \begin{bmatrix} 0 & 0 & 0 & 0 & \dots & 0 \\ a & b & c & 0 & \dots & 0 \\ 0 & a & b & c & \dots & 0 \\ 0 & \ddots & \ddots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & a & b & c \\ 0 & \dots & 0 & 0 & 0 & 0 \end{bmatrix},$$

where $a = \frac{D\tau}{h^2} + \frac{\tau v}{2h}, \mathrm{b} = 1 - \frac{2D\tau}{h^2}, \mathrm{c} = \frac{D\tau}{h^2} - \frac{\tau v}{2h}.$

Thus, we must regenerate A at every time step, k (instead of just once).

Time Dependent Velocity/Diffusion

Let's change our velocity from a constant to be dependent on time $v(t)$.

For time dependent $v(t)$,

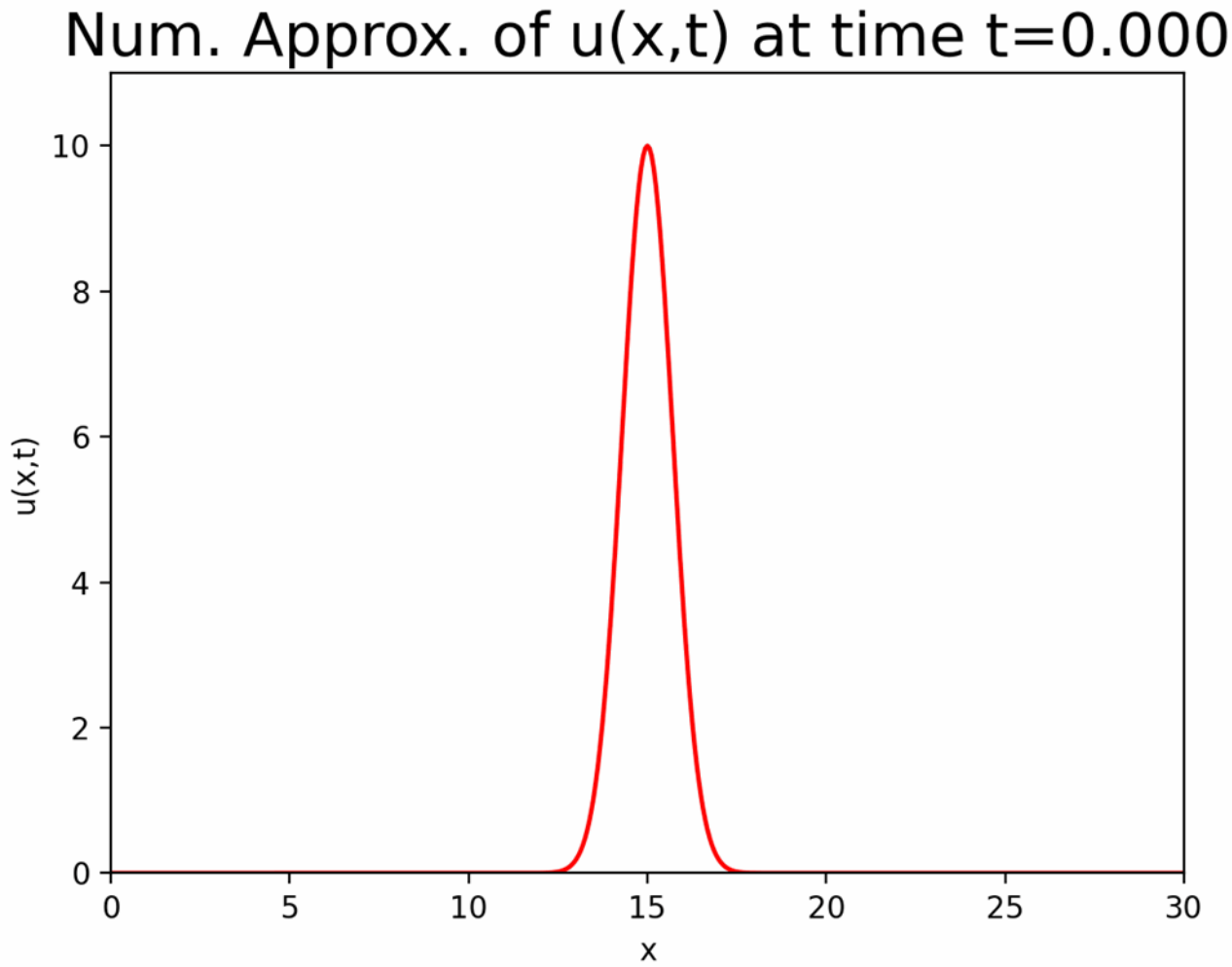
$$A_k := \begin{bmatrix} 0 & 0 & 0 & 0 & \dots & 0 \\ a & b & c & 0 & \dots & 0 \\ 0 & a & b & c & \dots & 0 \\ 0 & \ddots & \ddots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & a & b & c \\ 0 & \dots & 0 & 0 & 0 & 0 \end{bmatrix},$$

where $a = \frac{D\tau}{h^2} + \frac{\tau v(k\tau)}{2h}, \mathrm{b} = 1 - \frac{2D\tau}{h^2}, \mathrm{c} = \frac{D\tau}{h^2} - \frac{\tau v(k\tau)}{2h}.$

Thus, we must regenerate A_k at every time step, k (instead of just once).

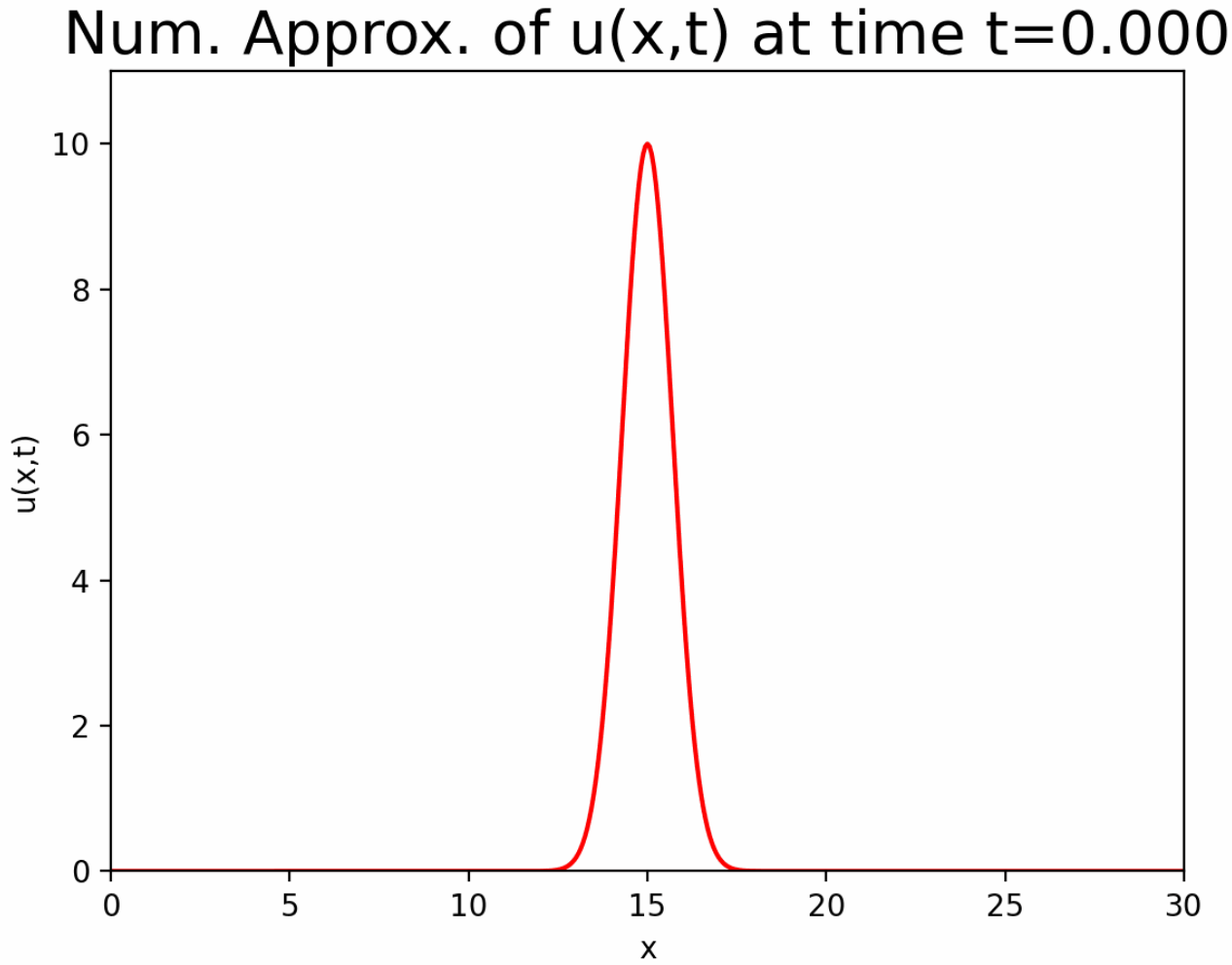
Running the Scheme with $v(t)$

$L = 30$
$T = 2$
$D = 0.5$
$v(t) = 15\sin(20t)$
$I = 500$
$K = 600$
$u_0(x) = 10e^{-(x-L/2)^2}$



Running the Scheme with $v(t)$

$L = 30$
$T = 2$
$D = 0.5$
$v(t) = 15\sin(20t)$
$I = 500$
$K = 600$
$u_0(x) = 10e^{-(x-L/2)^2}$



Your turn!

Clone the repository from GitHub:

- Either go to my GitHub @keversma and copy the repo link,
- Or type the below into your terminal (make sure you are already in whatever folder you want this to show up in):

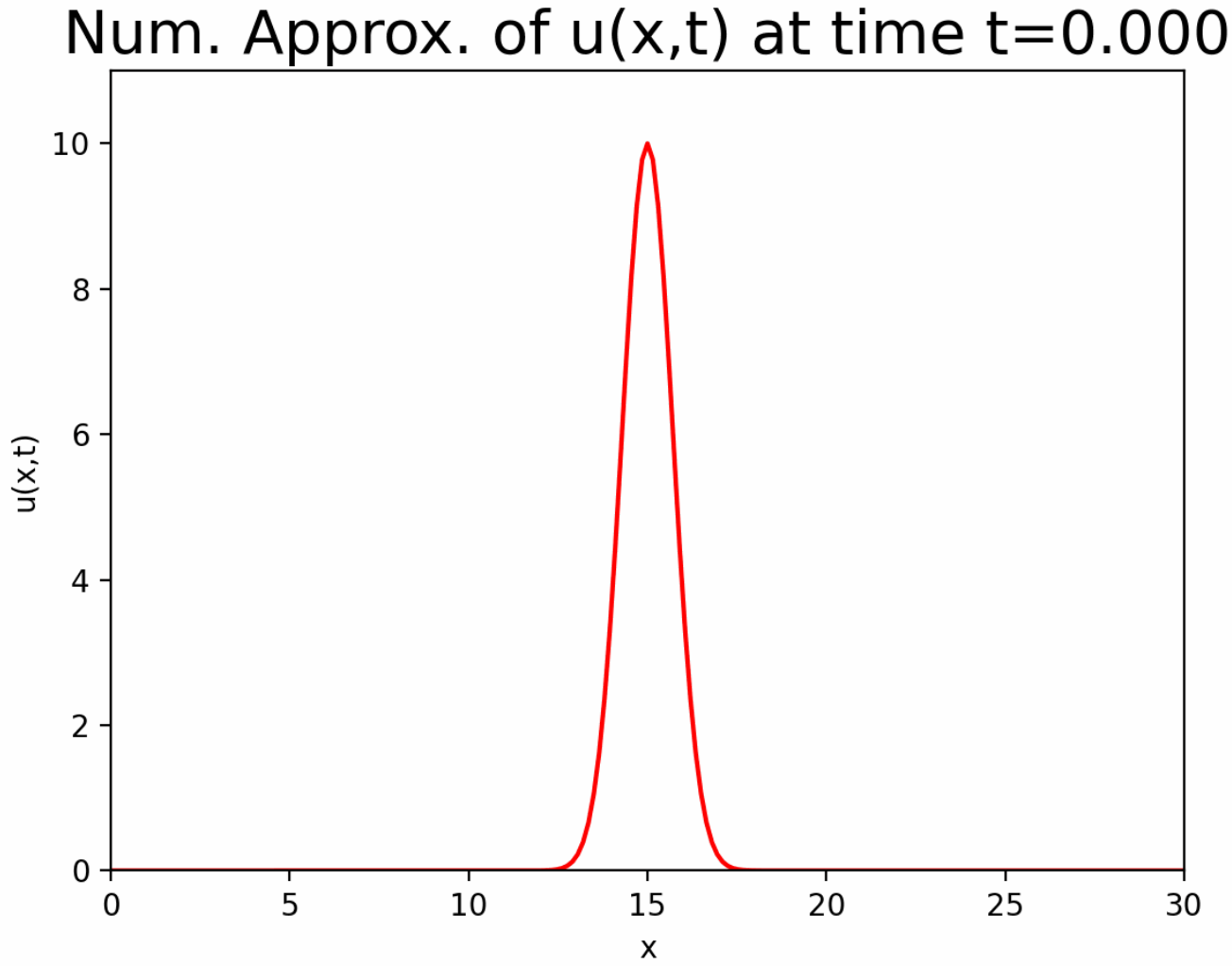
```
> git clone https://github.com/keversma/Solving_Adv_Diff_Lesson.git
> cd Solving_Adv_Diff_Lesson
```

- Run one_dim_advection_diffusion_scheme.py

Your turn!

$L = 30$
$T = 2$
$v = 3$
$D = 0.5$
$I = 200$
$K = 300$
$u_0(x) = 10e^{-(x-L/2)^2}$

1. Make sure you can get this graph working on your computer.
2. How do we make the graph move left instead of right?
3. Try changing other inputs from above such as step sizes, initial value functions, velocity, diffusion, etc.!



Helpful Resources

A.J. Salgado and Steven M. Wise. *Classical Numerical Analysis: A Comprehensive Course*. Cambridge University Press, 2022.

- Link on Abner's website
- Chapter 24 is FDM

Questions?