

# Numerical methods to solve differential equations

Enrico Baglione



Download PDF

# Outline

1. Introduction and Discretization

1. Explicit and Implicit Methods

2. Runge-Kutta Methods

2. The Butcher Tableaux

3. Linear Multistep Methods

4. Backward Differentiation Formula (BDF)

5. Spatial Temporal Discretization

6. Direction of Propagation in the Transport Equation

# Introduction to Numerical Methods

The bridge between theory and reality

# What Are Numerical Methods?

## From Continuous to Discrete

Numerical methods are computational techniques that approximate solutions to mathematical problems where exact methods often fail.

### The Core Process

- **Discretization:** Converting continuous expressions (PDEs/ODEs) into manageable discrete forms.
- **Techniques:** Finite Differences (FDM), Finite Elements (FEM), Iterative Solvers.

### The Philosophy

- Focus on **practicality** over pure symbolic elegance.
- Delivers answers with **controllable and quantifiable accuracy**.
- Invaluable for data-driven and computation-heavy environments.

# Why Are They Indispensable?

## 1. Tackling Real-World Complexity

Analytical solutions are rare for nonlinear, coupled equations or complex boundary conditions.

- **Jet Engines:** Simulating heat transfer via Finite Difference methods.
- **Skyscrapers:** Modeling stress distribution using Finite Element Analysis (FEA).
- **Acoustics:** Solving the Wave Equation for concert hall design.

# Why Are They Indispensable?

## 2. Scalability and Interdisciplinary Utility

### Scalability

Handling massive systems of equations for:

- Climate simulations
- Large-scale Neural Networks
- Quantum simulations

### Disciplines

- **Physics:** Chaotic systems and plasma dynamics.
- **Finance:** Option pricing via Stochastic Differential Equations.
- **Biology:** Disease spread and population dynamics.

# Theoretical Underpinning

## Beyond Pure Computation

The success of any numerical algorithm is grounded in three pillars:

- **Convergence:** Does the approximate solution  $v$  approach the exact solution  $u$  as  $\Delta x, \Delta t \rightarrow 0$ ?
- **Stability:** Does the algorithm suppress errors (round-off/truncation) over time?
- **Consistency:** Does the numerical scheme truly represent the original PDE?

**Lax Equivalence Theorem:** For a well-posed linear problem, Consistency + Stability = Convergence.

# Challenges and the Path Forward

## Precision vs. Cost

### Current Hurdles

- **Trade-offs:** High accuracy vs. substantial computational resources.
- **Conditioning:** Managing ill-conditioned systems and truncation errors.
- **Design:** Adapting general methods to specific, creative problems.

### The Future

- High-Performance Computing (HPC)
- Machine Learning-assisted solvers
- Adaptive Mesh Refinement (AMR)

# Real-World Impact

## Inspiration for Engineers and Scientists



**The Millau Viaduct:** Designed using FEM for maximum safety and efficiency.



**Tsunami Warnings:** Numerical propagation models providing life-saving early alerts.



**Space Exploration:** Runge-Kutta methods calculating spacecraft trajectories.

## Some Thoughts

As Master's students, you are not just learning tools—**you are shaping the future.**

*Numerical methods are the engine behind progress in science, engineering, and technology.*

**Go beyond the numbers: Solve problems, understand the world, and build the future.**

# Discretization

# Differential Equations

Numerical methods to solve differential equations:

$$u_t = f(u, t) \quad u_t = \frac{du}{dt}$$

Example:

$$u_t = A(t) \cdot u + B(t) \quad u_t = A(t) \cdot u$$

Generally, we want to pass from  $N$  order differential equations to a system of  $N$  first order diff. equations:

$$\begin{cases} u_t = v \\ v_t = g(u, v, t) \end{cases}$$

# Separation of variables

Given the Cauchy problem:

$$\begin{cases} u_t = Au \\ u(0) = u_0 \end{cases}$$

We are going to solve this system with the separation of variables:

$$\begin{aligned} \frac{du}{dt} &= Au \\ \Rightarrow \frac{du}{u} &= A dt \\ \Rightarrow \int_{u_0}^u \frac{du'}{u'} &= \int_0^t A dt \\ \Rightarrow \ln \left( \frac{u}{u_0} \right) &= At \\ \Rightarrow u(t) &= u_0 e^{At} \end{aligned}$$

# Discretization of the Solution

Given the Cauchy problem:

$$\begin{cases} u_t = f(u, t) \\ u(0) = u_0 \end{cases}$$

Let's consider a temporal step  $\Delta t = k$ :

$$t_0 = 0; \quad t_1 = k; \quad t_2 = 2k; \quad \dots \quad t_k = nk; \quad 0 \leq n \leq N$$

I want to obtain an approximation  $v^n$  of  $u(t_n)$ :

$$u(t_n) = u_n \simeq v^n \quad \Rightarrow \quad f(u_n, t_n) \simeq f(v^n, t_n) = f^n$$

Then I should find a criterion to build the sequence  $v^0, v^1, \dots, v^n$  that furnishes the discrete representation of  $u(t)$  in time. That will represent my **numerical solution**.

# Finite Difference Quotient

I can apply the definition of derivative as **difference quotient**:

$$\frac{u(t_{n+1}) - u(t_n)}{t_{n+1} - t_n} \simeq u_t(t_n) \Rightarrow u_t(t_n) \simeq \frac{v^{n+1} - v^n}{k} = f^n$$

Obtaining then a recursive formula:

$$\begin{cases} v^{n+1} = v^n + kf^n \\ v^0 = u_0 \end{cases}$$

1. Starting from  $v^0$ , I evaluate  $v^1 = v^0 + kf^0$  with  $v^0 = u_0$  and  $f^0 = f(u_0, 0)$ .
2. Once found  $v^1$ , I calculate  $f^1 = f(v^1, k)$ .
3. ... and so on.

# Explicit vs. Implicit Schemes

ACHTUNG!

I can have both:

$$\frac{v^{n+1} - v^n}{k} = f^n \quad \text{and} \quad \frac{v^{n+1} - v^n}{k} = f^{n+1}$$

Because I evaluate the derivative at endpoints of interest interval  $[t_n; t_{n+1}]$ .

I obtain then two recursive formulas:

$$v^{n+1} = v^n + k \cdot f^n$$

Explicit Euler

$$v^{n+1} = v^n + k \cdot f^{n+1}$$

Implicit Euler



Implicit because the  $f^{n+1}$  term contains  $v^{n+1}$ , a value that already appears at the first member.

# Slide: Central and Trapezium Schemes

I can also evaluate the derivative in a symmetrical point of the interval:

$$u_t(t_n) = \frac{u(t_{n+1}) - u(t_{n-1})}{t_{n+1} - t_{n-1}} = \frac{v^{n+1} - v^{n-1}}{2k}$$



$$v^{n+1} = v^{n-1} + 2k \cdot f^n$$

Central Value

Taking the central value I cannot apply the formula to evaluate  $v^1$ : I have to use Euler for that.



Or I can also evaluate the average of  $f^n$  and  $f^{n+1}$ :

$$\frac{v^{n+1} - v^n}{k} = \frac{1}{2} (f^{n+1} + f^n)$$



$$v^{n+1} = v^n + \frac{k}{2} (f^n + f^{n+1})$$

Trapezium

# Slide: Example - Explicit Euler

## Example

Now let's apply the previous formulas to the Cauchy problem:

$$\begin{cases} u_t = f(u, t) = u \\ u(0) = u_0 \end{cases}$$

whose analytical solution is  $u = u_0 \cdot e^t$ .

$$[1] \quad v^1 = u_0 + u_0 \cdot k = u_0(1 + k)$$

$$[2] \quad v^2 = v^1 + k \cdot f^1 = u_0(1 + k)^2$$

$$[3] \quad v^3 = u_0(1 + k)^3$$

$$[n] \quad v^n = u_0(1 + k)^n$$

RECURSIVE STEP

$$v^{n+1} = v^n + k f^n$$

Explicit Euler

# Slide: Example - Implicit

$$[1] \quad v^1 = v^0 + kv^1 \implies v^1 = \frac{1}{1-k}u_0$$

$$[2] \quad v^2 = v^1 + kv^2 \implies v^2 = \frac{1}{(1-k)^2}u_0$$

$$[3] \quad v^3 = u_0(1-k)^{-3}$$

$$[n] \quad v^n = u_0(1-k)^{-n}$$

RECURSIVE STEP

$$v^{n+1} = v^n + kf^{n+1}$$

*Implicit Euler*

# Slide: Example - Trapezium

$$[1] \quad v^1 = v^0 + \frac{k}{2}(v^0 + v^1) \implies v^1 = u_0 \frac{1 + k/2}{1 - k/2}$$

$$[2] \quad v^2 = v^1 + \frac{k}{2}(v^1 + v^2) \implies v^2 = u_0 \frac{(1 + k/2)^2}{(1 - k/2)^2}$$

$$[3] \quad v^3 = u_0 \frac{(1 + k/2)^3}{(1 - k/2)^3}$$

$$[n] \quad v^n = u_0 \left( \frac{1 + k/2}{1 - k/2} \right)^n$$

RECURSIVE STEP

$$v^{n+1} = v^n + \frac{k}{2}(f^n + f^{n+1})$$

Trapezium Method

# Slide: Comparison of Approximations (n=1)

Now let's try to understand which of the outlined methods better approximate the analytical solution. The latter, evaluated at time  $t_n$ , is:  $u(t_n) = u_0 e^{nk}$ . Let's consider the 4 formulas at the first time step ( $n = 1$ ).

Explicit:

$$\frac{v^1}{u_0} = 1 + k$$

Implicit:

$$\frac{v^1}{u_0} = \frac{1}{1 - k} \approx 1 + k + k^2 + k^3 + \dots$$

Trapezium:

$$\frac{v^1}{u_0} = \frac{1 + k/2}{1 - k/2} \approx 1 + k + \frac{k^2}{2} + \frac{k^3}{4} + \dots$$

The trapezium method offers the best approximation, exact until the third term (second order in  $k$ ). [The above formulas are obtained using Taylor expansions]

ANALYTICAL SOLUTION

$$\frac{u(t_1)}{u_0} = e^k$$

$$1 + k + \frac{k^2}{2} + \dots + \frac{k^m}{m!}$$

# Explicit vs. Implicit Methods

Advantages, Disadvantages, and Applications

# Choosing the right numerical approach

The choice between **Explicit** and **Implicit** schemes depends on the specific nature of the physical problem.

## Explicit Methods

Compute the next state directly from current known values.

- **Analogy:** Looking at the current map to take the next step.

## Implicit Methods

Compute the next state by solving a system of equations.

- **Analogy:** Solving a puzzle where the next step must satisfy a global balance.

# Explicit Numerical Methods

Direct computation:  $u^{n+1} = F(u^n)$

## Advantages

- **Simplicity:** Straightforward implementation.
- **Memory:** Low requirements (no large matrices).
- **Efficiency:** Very fast per time step for high-resolution needs.

## Disadvantages

- **Stability Constraints:** Must satisfy the **CFL Condition:**

$$c \frac{\Delta t}{\Delta x} \leq 1$$

- **Stiff Problems:** Inefficient; requires extremely small  $\Delta t$  to avoid "blowing up."

# Example: The Wave Equation

## Classical Explicit Application

The 1D Wave Equation describes propagation where information travels at a finite speed  $c$ :

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

**Discretization:**

$$u_j^{n+1} = 2u_j^n - u_j^{n-1} + C^2(u_{j+1}^n - 2u_j^n + u_{j-1}^n)$$

where  $C = c \frac{\Delta t}{\Delta x}$ .

# Implicit Numerical Methods

Coupled computation:  $G(u^{n+1}, u^n) = 0$

## Advantages

- **Unconditional Stability:** Allows much larger time steps  $\Delta t$ .
- **Stiff Systems:** The only viable choice for chemical kinetics or heavy diffusion.
- **Long-Term:** Efficient for long duration simulations.

## Disadvantages

- **Complexity:** Requires solving a system of equations (Matrix Inversion).
- **Memory:** High; needs to store and solve large matrices.
- **Non-linear Problems:** Requires iterative solvers (e.g., Newton-Raphson).

# Example: The Heat Equation

## Implicit Mastery with Crank-Nicolson

Modeling heat conduction in a solid:  $\frac{\partial u}{\partial t} = \alpha \frac{\partial^2 u}{\partial x^2}$

The **Crank-Nicolson** scheme is a popular implicit method:

$$\frac{u_j^{n+1} - u_j^n}{\Delta t} = \frac{\alpha}{2} [\delta_x^2 u_j^{n+1} + \delta_x^2 u_j^n]$$

- Results in a **Tridiagonal Matrix** system.
- **Unconditionally Stable:** You can pick  $\Delta t$  based on accuracy, not just to prevent crashes.



# Summary Comparison

Feature	Explicit	Implicit
<b>Computation</b>	Direct (Algebraic)	Iterative/Matrix (System)
<b>Stability</b>	Conditional (CFL)	Often Unconditional
<b>Time Step</b>	Must be small	Can be large
<b>Memory</b>	Low	High
<b>Best for...</b>	Wave propagation, acoustics	Diffusion, chemical kinetics, stiffness
<ul style="list-style-type: none"><li>▪ <b>Use Explicit</b> for high-speed dynamics (Sound, Impact).</li><li>▪ <b>Use Implicit</b> for slow-evolving, diffusion-dominated systems (Heat, Groundwater).</li></ul>		

# Runge-Kutta methods

# Runge-Kutta methods

Up to now, we have limited ourselves to evaluating the function only at the limits of each time step.

Now, given a time interval  $[t_n, t_{n+1}]$ , the idea is to also evaluate the function at intermediate points within this interval to improve accuracy.

STEP 1: Intermediate Value (Euler)

$$\tilde{v}^{n+\frac{1}{2}} = v^n + \frac{k}{2} f^n$$

STEP 2: Evaluation

$$\tilde{f}^{n+\frac{1}{2}} = f(\tilde{v}^{n+\frac{1}{2}}, t_{n+\frac{1}{2}})$$

FINAL STEP:

$$v^{n+1} = v^n + k \tilde{f}^{n+\frac{1}{2}}$$

RECURSIVE STEP

$$v^{n+1} = v^n + k f^{n+\frac{1}{2}}$$

Runge-Kutta 2 (RK2)

The **Midpoint** method uses an intermediate estimate to improve the slope accuracy.

# Two-Stage Intermediate Time Integration

We consider two intermediate times within the interval  $[t_n, t_{n+1}]$ , namely  $t_{n+\frac{1}{3}}$  and  $t_{n+\frac{2}{3}}$ . Starting from the known values  $(t_n, v^n, f^n)$ , we can define intermediate estimates as follows:

[1] First Stage:

$$\tilde{v}^{n+\frac{1}{3}} = v^n + \frac{k}{3} f^n, \quad \tilde{f}^{n+\frac{1}{3}} = f(\tilde{v}^{n+\frac{1}{3}}, t_{n+\frac{1}{3}})$$

[2] Second Stage:

$$\tilde{v}^{n+\frac{2}{3}} = v^n + \frac{2k}{3} \tilde{f}^{n+\frac{1}{3}}, \quad \tilde{f}^{n+\frac{2}{3}} = f(\tilde{v}^{n+\frac{2}{3}}, t_{n+\frac{2}{3}})$$

[3] Final Step:

$$v^{n+1} = v^n + \frac{k}{2} \left( \tilde{f}^{n+\frac{1}{3}} + \tilde{f}^{n+\frac{2}{3}} \right)$$

MULTI-STAGE STEP

$$v^{n+1} = v^n + \sum c_i f_i$$

Runge-Kutta Variant

The idea is to divide the interval into intermediate times (in principle, we could use more stages), and recursively construct  $\tilde{v}^{n+c_i}$  and  $\tilde{f}^{n+c_i}$  for  $0 < c_i < 1$ . This approach improves the accuracy by incorporating multiple evaluations of the derivative function  $f(v, t)$  within each time step.

# General Recursive Formula

Starting from  $c_1 = 0$ , let  $\alpha_{ij}$  denote the intermediate coefficients and  $b_i$  the final weights.

[1]

Known  $v^{n+c_1}$  and  $f^{n+c_1} = f(v^{n+c_1}, t_{n+c_1})$

[2]

$$\tilde{v}^{n+c_2} = v^n + k \alpha_{21} f^{n+c_1}; \quad \tilde{f}^{n+c_2} = f(\tilde{v}^{n+c_2}, t_{n+c_2})$$

[3]

$$\tilde{v}^{n+c_3} = v^n + k (\alpha_{31} f^{n+c_1} + \alpha_{32} f^{n+c_2}); \quad \tilde{f}^{n+c_3} = f(\tilde{v}^{n+c_3}, t_{n+c_3})$$

[s]

$$\tilde{v}^{n+c_s} = v^n + k (\alpha_{s1} f^{n+c_1} + \dots + \alpha_{s,s-1} f^{n+c_{s-1}})$$

BUTCHER TABLEAU

$c$	$A$
	$b^T$

General RK Structure

[END]

$$v^{n+1} = v^n + k \sum_{i=1}^s b_i f^{n+c_i}$$

# Runge–Kutta Coefficient Matrix (Butcher Tableau)

$c_1$				
$c_2$	$\alpha_{21}$			
$c_3$	$\alpha_{31}$	$\alpha_{32}$		
$\dots$	$\dots$	$\dots$	$\ddots$	
$c_s$	$\alpha_{s1}$	$\alpha_{s2}$	$\cdots$	$\alpha_{s,s-1}$
	$b_1$	$b_2$	$\cdots$	$b_{s-1}$
				$b_s$

$c_i$ : stage time coefficients

$\alpha_{ij}$ : intermediate weights,  $i = 1, \dots, s, j = 1, \dots, i - 1$

$b_i$ : final weights

# Runge–Kutta of Order 1 (Midpoint Method)

[1]

$$\tilde{v}^{n+\frac{1}{2}} = v^n + \frac{k}{2} f^n \quad \Rightarrow \quad \tilde{f}^{n+\frac{1}{2}}$$

[2]

$$v^{n+1} = v^n + k \tilde{f}^{n+\frac{1}{2}}$$

MIDPOINT (RK2)

0	$\frac{1}{2}$
$\frac{1}{2}$	0

The simple Euler method can be viewed as a Runge–Kutta method of order zero, with matrix:

0	
	1



$$v^{n+1} = v^n + k f^n$$

# From Matrix to Implementation

[1]

$$\tilde{v}^{n+\frac{1}{3}} = v^n + \frac{k}{3} f^n \Rightarrow \tilde{f}^{n+\frac{1}{3}}$$

[2]

$$\tilde{v}^{n+\frac{2}{3}} = v^n + k \left( 0 \cdot f^n + \frac{2}{3} \tilde{f}^{n+\frac{1}{3}} \right) \Rightarrow \tilde{f}^{n+\frac{2}{3}}$$

---

[END]

$$v^{n+1} = v^n + \frac{k}{4} \left( f^n + 3 \tilde{f}^{n+\frac{2}{3}} \right)$$

*[Heun formula, 2nd order]*

HEUN (RK3 VARIANT)

0	$\frac{1}{3}$	
$\frac{1}{3}$	0	$\frac{2}{3}$
$\frac{1}{4}$	0	$\frac{3}{4}$

# Application Example

(1)

Initial state:  $t_n, v^n, f^n$

(2)

$$\tilde{v}^{n+c_2} = v^n + k\alpha_{21}f^n \quad \Rightarrow \quad \tilde{f}^{n+c_2}$$

(3)

$$v^{n+1} = v^n + k(b_1f^n + b_2\tilde{f}^{n+c_2})$$

GENERAL 2-STAGE  
RK

0		
$c_2$	$\alpha_{21}$	
	$b_1$	$b_2$

Apply this to the Cauchy problem:

$$\begin{cases} u_t = u \\ u(0) = u_0 \end{cases}$$



Analytical Solution:

$$u(t) = u_0 e^t$$

# Order Conditions

(1)

$$t_0 = 0, \quad v^0 = u_0, \quad f^0 = u_0$$

(2)

$$\tilde{v}^{c_2} = u_0 + k\alpha_{21}u_0 = u_0(1 + k\alpha_{21}) \quad \Rightarrow \quad \tilde{f}^{c_2} = \tilde{v}^{c_2}$$

(3)

$$v^1 = u_0 [1 + k(b_1 + b_2) + k^2 b_2 \alpha_{21}]$$

## STABILITY ANALYSIS

$$u(t_1) = u_0 e^k$$

$$u_0(1 + k + \frac{1}{2}k^2 + \dots)$$

Comparing with the exact solution, for **second-order accuracy** we require:

$$\begin{cases} b_1 + b_2 = 1 \\ \alpha_{21} b_2 = \frac{1}{2} \end{cases}$$

Two equations for three unknowns  $(b_1, b_2, \alpha_{21})$ .

We add the common constraint relating coefficients of the same row.

# Consistency Constraints

General row-sum constraint (consistency):

$$\begin{aligned}\alpha_{21} &= c_2 \\ \alpha_{31} + \alpha_{32} &= c_3 \\ &\vdots \\ \alpha_{s1} + \cdots + \alpha_{s,s-1} &= c_s\end{aligned}$$



$$\sum_{j=1}^{i-1} \alpha_{ij} = c_i$$

IMPOSING THE CONSTRAINT FOR OUR CASE:

$$\begin{cases} b_1 + b_2 = 1 \\ c_2 b_2 = \frac{1}{2} \end{cases}$$

These conditions ensure second-order accuracy by reducing the number of independent unknowns.

ORDER 2  
CONDITIONS

$$s = 2$$

THE RESULTING R-K MATRIX IS:

$$\begin{array}{c|cc} 0 & c_2 \\ \hline c_2 & c_2 \\ \hline 1 - \frac{1}{2c_2} & \frac{1}{2c_2} \end{array}$$

PARAMETRIC FORM

*General solution for 2nd order*

*accuracy with  $s = 2$*

In general, it also holds that:

$$\sum_{i=1}^s b_i = 1$$

*For specific choices of coefficients, canonical reference matrices exist.*

# Butcher Tableaux

The DNA of Runge-Kutta Methods

# The Anatomy of the Table

## Systematic representation of coefficients

A **Butcher Tableau** provide a compact way to represent the coefficients of Runge-Kutta (RK) methods:

$$\begin{array}{c|c} \mathbf{c} & \mathbf{A} \\ \hline & \mathbf{b}^T \end{array}$$

- $\mathbf{A} = [a_{ij}]$ : Matrix of coefficients for intermediate stages.
- $\mathbf{c} = [c_i]$ : Nodes (time offsets).
- $\mathbf{b} = [b_i]$ : Weights for the final update.

# Explicit vs. Implicit: The Matrix Test

## Identifying the scheme type at a glance

 Explicit Methods

 Implicit Methods

**Condition:**  $\mathbf{A}$  is **strictly lower triangular**.

- $a_{ij} = 0$  for all  $j \geq i$ .
- Stage  $k_i$  depends only on  $k_1, \dots, k_{i-1}$ .
- **Result:** Direct evaluation, no iterative solvers needed.

**Condition:**  $\mathbf{A}$  has **nonzero entries** on or above the diagonal.

- $a_{ij} \neq 0$  for some  $j \geq i$ .
- Stage  $k_i$  depends on itself or future stages.
- **Result:** Requires solving algebraic equations (e.g., Newton's method).

# Comparative Examples

## Case studies in matrix structure

Explicit Euler

$$\begin{array}{c|c} 0 & \mathbf{0} \\ \hline & 1 \end{array}$$

**Diagnosis:** Strictly lower triangular. **Type:** Explicit.

Implicit Euler

$$\begin{array}{c|c} 1 & \mathbf{1} \\ \hline & 1 \end{array}$$

**Diagnosis:** Nonzero diagonal. **Type:** Implicit.

Trapezoidal Rule

$$\begin{array}{c|cc} 0 & 0 & 0 \\ 1 & 1/2 & \mathbf{1/2} \\ \hline & 1/2 & 1/2 \end{array}$$

**Diagnosis:** Diagonal entry ( $a_{22}$ ) is non-zero. **Type:** Implicit.

# Deep Dive: The Classical RK4

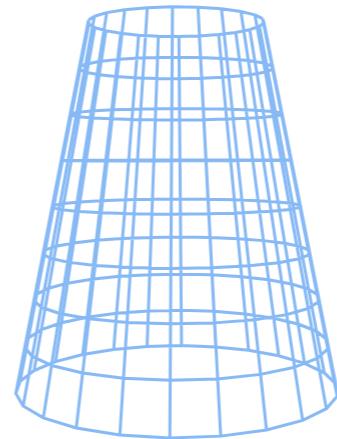
## Verifying the structure

Let's examine the standard fourth-order Runge-Kutta matrix  $\mathbf{A}$ :

$$\mathbf{A} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 1/2 & 0 & 0 & 0 \\ 0 & 1/2 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

- **Diagonal check:** All  $a_{ii}$  are 0.
- **Upper triangle check:** All  $a_{ij}$  (where  $j > i$ ) are 0.
- **Conclusion:** The matrix is strictly lower triangular.
- **Final Verdict:** RK4 is an **explicit** method.

# Butcher Tableaux Examples



# 1. First & Third Order Methods

From Euler to Heun

Explicit Euler (1st Order)

$$\begin{array}{c|c} 0 & 0 \\ \hline & 1 \end{array}$$

**Formula:**  $y_{n+1} = y_n + h f(t_n, y_n)$

Heun's Method (3rd Order)

$$\begin{array}{c|ccc} 0 & 0 & 0 & 0 \\ 1/3 & 1/3 & 0 & 0 \\ 2/3 & 0 & 2/3 & 0 \\ \hline & 1/4 & 0 & 3/4 \end{array}$$

**Formula:**  $y_{n+1} = y_n + h(\frac{1}{4}k_1 + \frac{3}{4}k_3)$

## 2. Strong Stability Preserving

### SSPRK3 (3rd Order Explicit)

Often used for hyperbolic PDEs to avoid oscillations.

$$\begin{array}{c|ccc} 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ \hline 1/2 & 1/4 & 1/4 & 0 \\ \hline & 1/6 & 1/6 & 2/3 \end{array}$$

**Why it matters:** SSPRK3 ensures that the solution does not grow in norm, making it ideal for non-linear conservation laws.

### 3. Fourth-Order Explicit Methods

The Gold Standard

Classical RK4

0	0	0	0	0
1/2	1/2	0	0	0
1/2	0	1/2	0	0
1	0	0	1	0
	1/6	1/3	1/3	1/6

The most widely used integrator.

3/8-Rule

0	0	0	0	0
1/3	1/3	0	0	0
2/3	-1/3	1/3	0	0
1	1	-1	1	0
	1/8	3/8	3/8	1/8

An alternative with different stability properties.

## 4. Implicit Schemes: Lobatto IIIC

### Stability for Stiff Problems

Unlike explicit methods, the  $A$  matrix is dense, requiring a system solver.

$$\begin{array}{c|ccc} 0 & 1/6 & -1/6 & 0 \\ 1/2 & 1/6 & 1/3 & -1/6 \\ 1 & 1/6 & 5/6 & 1/6 \\ \hline & 1/6 & 2/3 & 1/6 \end{array}$$

# Analyzing an Invalid Table

## Case Study in Consistency

Consider this provided table:

0	0	0	0	0
1/2	1/2	0	0	0
1/2	1/2	1	0	0
1	0	0	1	0
	1/2	1/3	1/3	1/2

✗ Row Sum Violation

For  $c_3 = 1/2: 1/2 + 1 = 3/2 \neq 1/2$  **Result:**

Violation of consistency.

✗ Weight Violation

$$\sum b_i = 1/2 + 1/3 + 1/3 + 1/2 = 5/3$$

**Requirement:**  $\sum b_i = 1$  **Result:** Incorrect total step size.

# Conclusion: Why Validity Matters

A Butcher table is **invalid** if it fails consistency and order conditions.

- **Non-Convergence:** If  $\sum b_i \neq 1$ , the error does not vanish as  $h \rightarrow 0$ .
- **Stability:** Inconsistent  $c_i$  values lead to unphysical phase shifts.
- **Accuracy:** Higher-order terms will accumulate, causing the solution to diverge rapidly.

**Takeaway:** Always verify row sums and weight sums before implementing a custom tableau.

# Linear Multistep Methods

# Linear Multistep

A general form of a linear multistep scheme is given by:

$$\begin{aligned} \alpha_{n+1}v^{n+1} + \alpha_n v^n + \cdots + \alpha_{n-s+1} v^{n-s+1} &= \\ = k (\beta_{n+1} f^{n+1} + \beta_n f^n + \cdots + \beta_{n-s+1} f^{n-s+1}) \end{aligned}$$

This class of schemes is called a **Linear Multistep Method (LMM)**.

Unlike Runge–Kutta methods, which use multiple function evaluations within one step, LMMs use several past time levels to advance the solution.

We start again from the Cauchy problem:

$$\left\{ \begin{array}{l} u_t = f(u, t) \\ u(0) = u_0 \end{array} \right.$$

# Integral Formulation

We discretize time as  $t_n = nk$  and define:

$$v^n = u(t_n), \quad f^n = f(v^n, t_n)$$

Integrating between two consecutive time levels:

$$\int_{t_n}^{t_{n+1}} u_t dt = \int_{t_n}^{t_{n+1}} f(u, t) dt$$

Let  $q(t)$  be a polynomial that interpolates  $f$  within  $[t_n, t_{n+1}]$ .

Then:

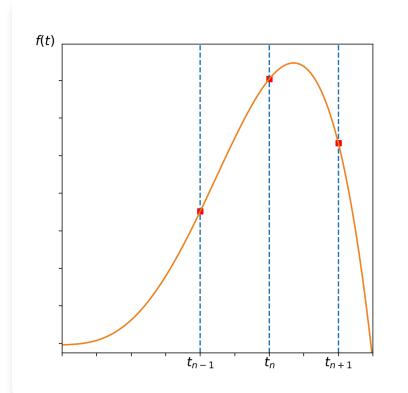
$$\begin{aligned} u(t_{n+1}) - u(t_n) &= \int_{t_n}^{t_{n+1}} f(t) dt \\ \Rightarrow v^{n+1} &= v^n + \int_{t_n}^{t_{n+1}} q(t) dt \end{aligned}$$

# Polynomial Interpolation and Implicitness

If  $f(t)$  is known at several points, we can build higher-order interpolating polynomials  $q(t)$ .

Depending on whether  $f^{n+1}$  is included in  $q(t)$ , the resulting scheme is:

- **Explicit**, if  $f^{n+1}$  is not included.
- **Implicit**, if  $f^{n+1}$  appears in  $q(t)$ .



*Interpolation of  $f(t)$  at multiple time levels.*

# Adams-Bashforth Methods

Consider  $q_{AB}(t)$  as the interpolating polynomial. We seek the solution in the time interval  $[t_n, t_{n+1}]$ .

## 1st ORDER

Supposing we know the function at  $t_n$ :  $q_{AB}(t) = f^n$ .

Applying  $t_{n+1} - t_n = k$ :

$$v^{n+1} = v^n + \int_{t_n}^{t_{n+1}} f^n dt$$

⇒  $v^{n+1} = v^n + kf^n$

| ADAMS-BASHFORTH, 1°

# Adams-Bashforth Methods

## 2nd ORDER

Suppose we know the function at  $f^{n-1}$  and  $f^n$ . Then  $q_{AB}(t)$  is the line passing through the points  $(t_{n-1}, f^{n-1})$  and  $(t_n, f^n)$ :

$$q_{AB}(t) = f^{n-1} + \frac{f^n - f^{n-1}}{t_n - t_{n-1}}(t - t_{n-1})$$

Integrating between  $t_n$  and  $t_{n+1}$  yields:

$$v^{n+1} = v^n + \frac{k}{2}(3f^n - f^{n-1})$$

ADAMS-  
BASHFORTH,  
2° ORDER

# Adams-Moulton Methods

Including also  $f^{n+1}$  in the interpolation makes the scheme **implicit**. We are interested in the solution in  $[t_n, t_{n+1}]$ .

## 1st ORDER

Supposing we know the function at  $t_{n+1}$ :  $q_{AM}(t) = f^{n+1}$ .

Integrating between  $t_n$  and  $t_{n+1}$  gives:

$$v^{n+1} = v^n + k f^{n+1}$$

ADAMS-MOULTON,  
1° ORDER

# Adams–Moulton Methods

## 2nd ORDER

Suppose  $q_{AM}(t)$  is the linear interpolant passing through  $(t_n, f^n)$  and  $(t_{n+1}, f^{n+1})$ :

$$q_{AM}(t) = f^n + \frac{f^{n+1} - f^n}{t_{n+1} - t_n}(t - t_n)$$

Integrating between  $t_n$  and  $t_{n+1}$  yields:

$$v^{n+1} = v^n + \frac{k}{2}(f^{n+1} + f^n)$$

ADAMS-  
MOULTON,  
2° ORDER

**Trapezoidal Rule:** This 2nd order implicit scheme is famously known as the Trapezoidal method. At the 3° ORDER I will have a parabola instead.

# Backward Differentiation Methods

# Backward Differentiation

We now approximate the solution  $u(t)$  itself with an interpolating polynomial  $q(t)$  passing through known points  $(t_{n+1}, v^{n+1}), (t_n, v^n), \dots$

Depending on the points selected,  $q$  will be a polynomial of  $1^\circ, 2^\circ, \dots$  order.

*Note: Approximating  $q$  as a constant at  $v^{n+1}$  results in a zero derivative, which is a too low order approximation.*

If  $q$  is of **degree 1**, it passes through  $(t_n, v^n)$  and  $(t_{n+1}, v^{n+1})$ :

$$q_{BD}(t) = v^n + \frac{v^{n+1} - v^n}{k}(t - t_n)$$

$$\Rightarrow \dot{q}_{BD}(t) = \frac{v^{n+1} - v^n}{k}$$

# Backward Differentiation – Order 1

We can now evaluate  $\dot{q}_{BD}(t)$  at different time levels:

- In  $t_n$ :  $\frac{v^{n+1} - v^n}{k} = f^n$

$$v^{n+1} = v^n + kf^n \quad (\textit{Explicit Euler})$$

- In  $t_{n+1}$ :  $\frac{v^{n+1} - v^n}{k} = f^{n+1}$

$$v^{n+1} = v^n + kf^{n+1} \quad (\textit{Implicit Euler})$$

---

These represent the simplest members of the Backward Differentiation family.

## Backward Differentiation – Order 2

Now consider three points:  $(t_{n-1}, v^{n-1})$ ,  $(t_n, v^n)$ , and  $(t_{n+1}, v^{n+1})$ .

The quadratic interpolant (a parabola) is:

$$\begin{aligned} q_{BD}(t) &= v^{n-1} + (t - t_{n-1}) \frac{v^n - v^{n-1}}{k} \\ &\quad + \frac{(t - t_n)(t - t_{n-1})}{2k^2} (v^{n+1} - 2v^n + v^{n-1}) \end{aligned}$$

Differentiating with respect to time:

$$\dot{q}_{BD}(t) = \frac{v^n - v^{n-1}}{k} + \frac{v^{n+1} - 2v^n + v^{n-1}}{2k^2} (2t - t_n - t_{n-1})$$

# Backward Differentiation – Evaluation

Evaluating the derivative  $\dot{q}_{BD}(t)$  at different time levels:

- **In  $t_n$ :**  $\dot{q}_{BD}(t_n) = f^n$

$\Rightarrow$

$$v^{n+1} = v^{n-1} + 2k f^n$$

(Explicit 2-step scheme)

- **In  $t_{n+1}$ :**  $\dot{q}_{BD}(t_{n+1}) = f^{n+1}$

$\Rightarrow$

$$v^{n+1} = -\frac{1}{3}v^{n-1} + \frac{4}{3}v^n + \frac{2}{3}k f^{n+1}$$

(Implicit 2-step scheme / BDF2)

# Spatial-temporal discretization

# Spatial-temporal discretization

Suppose we know the solution at discrete points  $(x_j, t_n)$ . We define  $v_j^n$  as the numerical approximation:  $v_j^n \simeq u(x_j, t_n)$ .

## Spatial Grid

$$x_{j+1} = x_j + h$$

$$\Delta x = h \quad (\text{spatial step})$$

## Temporal Grid

$$t_{n+1} = t_n + k$$

$$\Delta t = k \quad (\text{temporal step})$$

Let's introduce a series of **discrete operators**. Those acting on the **time** coordinate use a **superscript** ( $n$ ), while those for the **spatial** coordinate use a **subscript** ( $j$ ).

# Discrete Operators

Temporal (Time)

$$\delta^+ v_j^n = \frac{1}{k} (v_j^{n+1} - v_j^n)$$

$$\delta^- v_j^n = \frac{1}{k} (v_j^n - v_j^{n-1})$$

$$\delta^0 v_j^n = \frac{1}{2k} (v_j^{n+1} - v_j^{n-1})$$

Spatial (Space)

$$\delta_+ v_j^n = \frac{1}{h} (v_{j+1}^n - v_j^n)$$

$$\delta_- v_j^n = \frac{1}{h} (v_j^n - v_{j-1}^n)$$

$$\delta_0 v_j^n = \frac{1}{2h} (v_{j+1}^n - v_{j-1}^n)$$

Properties:

$$\delta^- v_j^{n+1} = \delta^+ v_j^n$$

$$\delta^0 = \frac{1}{2} (\delta^- + \delta^+)$$

$$\delta_- v_{j+1}^n = \delta_+ v_j^n$$

$$\delta_0 = \frac{1}{2} (\delta_- + \delta_+)$$

# Composition of Operators

Consider the composition of the two operators  $\delta^-$  and  $\delta^+$ . By defining  $\delta^X = \delta^- \cdot \delta^+$ , we obtain the second-order central difference:

$$\delta^X v_j^n = \frac{1}{k^2} (v_j^{n+1} - 2v_j^n + v_j^{n-1})$$

*Commutative Property:*

$$\delta^X = \delta^- \cdot \delta^+ = \delta^+ \cdot \delta^-$$

The spatial analog is defined similarly:

$$\delta_X v_j^n = \frac{1}{h^2} (v_{j+1}^n - 2v_j^n + v_{j-1}^n)$$

# Geometric Interpretation

If I know the solution near  $v_j^n$ , I could know it everywhere interpolating with a straight line:

$$u(x, t_n) = v_j^n + \frac{v_{j+1}^n - v_j^n}{h}(x - x_j) \quad [\text{line by } v_j^n; v_{j+1}^n]$$

The slope of the straight line is given by:

$$\delta_+ v_j^n = \frac{1}{h}(v_{j+1}^n - v_j^n) \quad \Rightarrow \quad \boxed{\delta_+ v_j^n = \frac{\partial}{\partial x} u(x, t_n)}$$

The same holds for  $\delta_-$  interpolating between  $x_{j-1}$  and  $x_j$ :

$$\delta_- v_j^n = \frac{1}{h}(v_j^n - v_{j-1}^n) \quad \Rightarrow \quad \boxed{\delta_- v_j^n = \frac{\partial}{\partial x} u(x, t_n)}$$

# Higher Order Interpolation

Now suppose we interpolate with a parabola passing through  $v_{j-1}^n, v_j^n, v_{j+1}^n$ . After some passages, we arrive at the second order derivative:

$$\frac{\partial^2}{\partial x^2} u(x, t_n) = \frac{1}{h^2} (v_{j+1}^n - 2v_j^n + v_{j-1}^n) = \delta_X v_j^n$$

This yields the central difference for the second derivative:

$$\Rightarrow \boxed{\frac{\partial^2 u}{\partial x^2} = \delta_X v_j^n} \quad \text{curvature of the parabola}$$

Summary of approximations:

- Interpolating polynomial 1° degree:  $D^{(1)} = \delta_{\pm}$
- Interpolating polynomial 2° degree:  $D^{(1)} = \delta_0; D^{(2)} = \delta_X$

It is possible to achieve the same result by expanding in a **Taylor series** around  $(x_j, t_n)$ .

# The 1D Transport Equation

The foundation of Advection and Wave Propagation

# Introduction

## What is the Transport Equation?

The 1D Transport (or Advection) equation is a first-order partial differential equation that describes how a scalar quantity  $u$  is carried along by a velocity field  $c$ .

### The Equation

$$\frac{\partial u}{\partial t} + c \frac{\partial u}{\partial x} = 0$$

- $u(x, t)$ : The quantity being transported (e.g., concentration, temperature).
- $c$ : The constant velocity of propagation.

### The Physical Meaning

It represents **pure translation**. The initial profile  $u(x, 0)$  moves along the x-axis at speed  $c$  without changing its shape.

# The Analytical Solution

## Method of Characteristics

For the initial condition  $u(x, 0) = \phi(x)$ , the exact solution is:

$$u(x, t) = \phi(x - ct)$$

### Key Observations:

1. **Constant Information:** The value of  $u$  is constant along the characteristic lines  $x - ct = \text{constant}$ .
2. **No Diffusion:** Unlike the heat equation, there is no smoothing; sharp edges remain sharp forever.

### Challenges for Numerical Methods:

- **Numerical Diffusion:** Some methods "smear" the wave.
- **Numerical Dispersion:** Some methods create artificial oscillations (wiggles).

# Discretizing the Domain

## Setting the stage for Numerical Methods

To solve this numerically, we replace the continuous domain with a discrete computational grid:

- **Spatial step ( $\Delta x$ ):**  $x_j = j\Delta x$
- **Time step ( $\Delta t$ ):**  $t^n = n\Delta t$

$$u(x_j, t^n) \approx u_j^n$$

Target: Find a scheme for  $u_j^{n+1}$  using values from  $u^n$

# Preview: The CFL Condition

## The Limit of Stability

For any explicit numerical method applied to the transport equation, stability is governed by the **Courant-Friedrichs-Lowy (CFL)** condition:

$$\nu = \frac{c\Delta t}{\Delta x} \leq 1$$

- $\nu$ : The Courant number.
- If  $\nu > 1$ , the numerical "information" travels slower than the physical wave, leading to total instability (the solution "blows up").
- **Physical interpretation:** The numerical domain of dependence must contain the physical domain of dependence.

# Next Step:

## Exploring Numerical Schemes

### Upwind Scheme

Stable & Dissipative

### Lax-Wendroff

2nd Order & Dispersive

### Beam-Warming

Second-order Upwind

# Numerical Schemes for Advection

Upwind vs. Lax-Friedrichs

# UP-WIND (UW1)

Let's start from the transport equation:  $u_t + cu_x = 0$ .

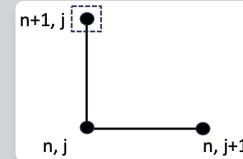
Starting from:

$$\delta^+ v_j^n = -c \delta_+ v_j^n$$

$$\frac{1}{k} (v_j^{n+1} - v_j^n) = -\frac{c}{h} (v_{j+1}^n - v_j^n) \quad \text{setting } \lambda = \frac{k}{h}$$

We obtain the update formula:

$$v_j^{n+1} = v_j^n - c\lambda (v_{j+1}^n - v_j^n)$$



*Two points at the base allow to find the upper one.*

# UP-WIND (UW2)

Starting from the transport equation:  $u_t + cu_x = 0$ .

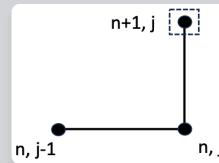
Starting from:

$$\delta^+ v_j^n = -c \delta_- v_j^n$$

$$\frac{1}{k} (v_j^{n+1} - v_j^n) = -\frac{c}{h} (v_j^n - v_{j-1}^n) \quad \text{setting } \lambda = \frac{k}{h}$$

We obtain the update formula:

$$v_j^{n+1} = v_j^n - c\lambda (v_j^n - v_{j-1}^n)$$



The backward spatial stencil uses points  
 $j - 1$  and  $j$ .

# UP-WIND Choice

But how can I choose between the previous formulas?

**CASE 1**    If  $c > 0$ :

The wave propagates backward, and it is useful to use the first formula:

$$\delta^+ v_j^n = c \delta_+ v_j^n$$

**CASE 2**    If  $c < 0$ :

The wave propagates forward, and it is useful to use the second one:

$$\delta^+ v_j^n = c \delta_- v_j^n$$

*Note: The choice of the spatial operator depends on the **direction of information flow** (characteristic direction).*

# EULER (EU)

Starting from the transport equation:  $u_t + cu_x = 0$ .

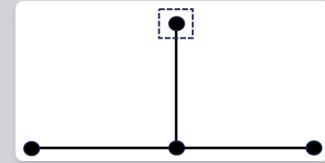
Starting from:

$$\delta^+ v_j^n = -c\delta_0 v_j^n$$

$$\frac{1}{k} (v_j^{n+1} - v_j^n) = -\frac{c}{2h} (v_{j+1}^n - v_{j-1}^n) \quad \text{setting } \lambda = \frac{k}{h}$$

We obtain the update formula:

$$v_j^{n+1} = v_j^n - \frac{1}{2} c \lambda (v_{j+1}^n - v_{j-1}^n)$$



The two points at the base allow to find  
the upper one.

# LEAP-FROG (LF)

Starting from the transport equation:  $u_t + cu_x = 0$ .

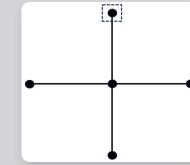
Starting from:

$$\delta^0 v_j^n = -c \delta_0 v_j^n$$

$$\frac{1}{2k} (v_j^{n+1} - v_j^{n-1}) = -\frac{c}{2h} (v_{j+1}^n - v_{j-1}^n) \quad \text{setting } \lambda = \frac{k}{h}$$

We obtain the update formula:

$$v_j^{n+1} = v_j^{n-1} - c\lambda (v_{j+1}^n - v_{j-1}^n)$$



The leap-frog scheme uses the time level  
 $n - 1$  to find  $n + 1$ .

# CRANK-NICHOLSON (CN)

Starting from the transport equation:  $u_t + cu_x = 0$ .

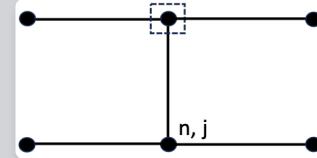
Starting from:

$$\delta^+ v_j^n = -\frac{c}{2} (\delta_0 v_j^n + \delta_0 v_j^{n+1})$$

$$\frac{1}{k} (v_j^{n+1} - v_j^n) = -\frac{c}{4h} (v_{j+1}^n - v_{j-1}^n + v_{j+1}^{n+1} - v_{j-1}^{n+1}) \quad \text{setting } \lambda = \frac{k}{h}$$

We obtain the update formula:

$$v_j^{n+1} = v_j^n - \frac{1}{4} c \lambda (v_{j+1}^n - v_{j-1}^n + v_{j+1}^{n+1} - v_{j-1}^{n+1})$$



The Crank-Nicholson stencil involves points at both time levels  $n$  and  $n + 1$ .

[implicit]

# BACKWARD EULER (B-EU)

Starting from the transport equation:  $u_t + cu_x = 0$ .

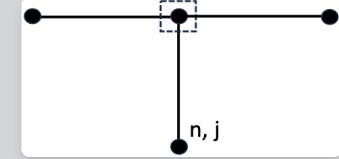
Starting from:

$$\delta^+ v_j^n = -c\delta_0 v_j^{n+1}$$

$$\frac{1}{k} (v_j^{n+1} - v_j^n) = -\frac{c}{2h} (v_{j+1}^{n+1} - v_{j-1}^{n+1}) \quad \text{setting } \lambda = \frac{k}{h}$$

We obtain the update formula:

$$v_j^{n+1} = v_j^n - \frac{1}{2} c \lambda (v_{j+1}^{n+1} - v_{j-1}^{n+1})$$



The Backward Euler scheme evaluates  
the spatial derivative at the future time  
level  $n + 1$ .

[implicit]

# LAX-FRIEDRICH (LF)

A central scheme stabilized by replacing the local value with a spatial average.

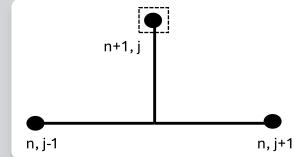
Starting from the centered approximation:

$$v_j^{n+1} = \frac{v_{j+1}^n + v_{j-1}^n}{2} - \frac{c\lambda}{2} (v_{j+1}^n - v_{j-1}^n)$$

$$\frac{1}{k} \left( v_j^{n+1} - \frac{v_{j+1}^n + v_{j-1}^n}{2} \right) = -\frac{c}{2h} (v_{j+1}^n - v_{j-1}^n) \quad \text{setting } \lambda = \frac{k}{h}$$

We obtain the update formula:

$$v_j^{n+1} = \frac{1}{2}(1 - c\lambda)v_{j+1}^n + \frac{1}{2}(1 + c\lambda)v_{j-1}^n$$



The update depends on the neighbors at time  $n$ , skipping the central node  $j$ .

# LAX-WENDROFF (LW)

It was born as an attempt to stabilize Euler: it adds a  $2^\circ$  order term to Euler's formula. [ some like  $u_t = -cu_x + \frac{c^2}{2}u_{xx}$  ]

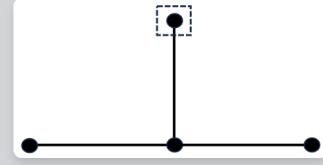
Starting from:

$$\delta^+ v_j^n = c \left( -\delta_0 v_j^n + c \frac{k}{2} \delta_X v_j^n \right)$$

$$\frac{1}{k} (v_j^{n+1} - v_j^n) = -\frac{c}{2h} (v_{j+1}^n - v_{j-1}^n) + \frac{c^2 k}{2h^2} (v_{j+1}^n - 2v_j^n + v_{j-1}^n)$$

We obtain the update formula:

$$v_j^{n+1} = v_j^n - \frac{1}{2} c \lambda (v_{j+1}^n - v_{j-1}^n) + \frac{1}{2} c^2 \lambda^2 (v_{j+1}^n - 2v_j^n + v_{j-1}^n)$$



*The Lax-Wendroff scheme adds numerical diffusion for stability.*

# BEAM-WARMING (BW)

A second-order accurate upwind scheme that utilizes a wider stencil to improve precision.

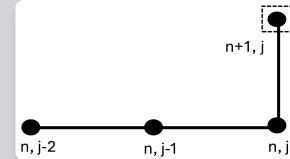
Starting from the second-order upwind discretization:

$$v_j^{n+1} = v_j^n - \nu \delta_- v_j^n + \frac{\nu(\nu-1)}{2} \delta_- \delta_- v_j^n$$

Where  $\delta_- v_j^n = v_j^n - v_{j-1}^n$  is the backward difference operator (for  $c > 0$ ) and  $\nu = \frac{c\Delta t}{\Delta x}$ .

We obtain the update formula:

$$v_j^{n+1} = v_j^n - \frac{\nu}{2} (3v_j^n - 4v_{j-1}^n + v_{j-2}^n) + \frac{\nu^2}{2} (v_j^n - 2v_{j-1}^n + v_{j-2}^n)$$



The stencil is fully biased upwind, requiring two points behind the current node.

# Composition of Operators

By composing the backward operator  $\delta_-$  with itself, we define  $(\delta_-)^2$ . This operator provides the second-order upwind spatial analog used in Beam-Warming:

Defining the second-order backward difference:

$$\delta_{--} v_j^n = \frac{1}{h^2} (v_j^n - 2v_{j-1}^n + v_{j-2}^n)$$

This is a purely upwinded operator, "looking" only at nodes  $j$ ,  $j - 1$ , and  $j - 2$ .

Derivation via operator composition:

$$\delta_-(\delta_- v_j^n) = \frac{1}{h} \left( \frac{v_j^n - v_{j-1}^n}{h} - \frac{v_{j-1}^n - v_{j-2}^n}{h} \right) = \delta_{--} v_j^n$$

(j-2) --- (j-1) --- (j)

*Unlike the central operator  $\delta_X$ , this stencil is asymmetric and biased upstream.*

# The Upwind Scheme

## Respecting the Physics of Flow ( $c > 0$ )

The Upwind scheme is the most intuitive approach: if the wind blows from the left, we look to the left to see what's coming.

### The Formula

Using a **Forward in Time** and **Backward in Space** (FTBS) discretization:

$$\frac{u_j^{n+1} - u_j^n}{\Delta t} + c \frac{u_j^n - u_{j-1}^n}{\Delta x} = 0$$

Solving for the future:

$$u_j^{n+1} = u_j^n - \nu(u_j^n - u_{j-1}^n)$$

where  $\nu = \frac{c\Delta t}{\Delta x}$  is the Courant number.

### Key Characteristics

- **First-order accurate** in time and space.
- **Stable** only if  $0 \leq \nu \leq 1$ .
- **Highly Dissipative:** It acts like "Numerical Diffusion," smoothing out sharp gradients and peaks.

# The Lax-Friedrichs Scheme

## Stability through Averaging

Standard central differences are unconditionally unstable for advection. Lax-Friedrichs fixes this by replacing  $u_j^n$  with a spatial average.

### The Modification

Replace  $u_j^n$  with  $\frac{1}{2}(u_{j+1}^n + u_{j-1}^n)$ :

$$u_j^{n+1} = \frac{1}{2}(u_{j+1}^n + u_{j-1}^n) - \frac{\nu}{2}(u_{j+1}^n - u_{j-1}^n)$$

### The Stencil

It uses  $u_{j-1}$  and  $u_{j+1}$ , skipping the central point  $u_j$  at the current time step.

### Pros & Cons

- **Pro:** Stable for  $|\nu| \leq 1$  (works for both  $c > 0$  and  $c < 0$ ).
- **Con:** Extremely dissipative. The "averaging" term introduces massive artificial viscosity, which can ruin accuracy in long simulations.

# Visualizing Numerical Diffusion

## Comparing the Exact vs. Numerical Profile

Numerical methods for transport equations often struggle to preserve the "shape" of the wave.

What you expect:

A square wave moving right, keeping its sharp corners ( $u_t + cu_x = 0$ ).

What Upwind/Lax-F gives:

The corners round off and the wave spreads out, looking like the Heat Equation solution.

Image comparing exact solution vs numerical solution with numerical diffusion for advection

*"Numerical diffusion is the price we pay for first-order stability."*

## Next Step: Higher Order

Can we achieve stability without the "Smearing"?

Enter: **Lax-Wendroff**

Second-order accuracy using a Taylor expansion in time.

# High-Order Accuracy

Lax-Wendroff & Von Neumann Stability

# The Lax-Wendroff Scheme

## Second-order precision via Taylor Expansion

To avoid the excessive diffusion of Upwind, we use a second-order Taylor expansion in time:  $u^{n+1} = u^n + \Delta t u_t + \frac{\Delta t^2}{2} u_{tt}$

### The Strategy

Using the transport equation  $u_t = -cu_x$ , we find that  $u_{tt} = c^2 u_{xx}$ . Substituting these:

$$u_j^{n+1} = u_j^n - \frac{\nu}{2}(u_{j+1}^n - u_{j-1}^n) + \frac{\nu^2}{2}(u_{j+1}^n - 2u_j^n + u_{j-1}^n)$$

### The Result: Dispersion

- **Pro:** Much less "smearing" (diffusion) than Upwind.
- **Con:** Introduces **Numerical Dispersion**. Sharp gradients create artificial "wiggles" (oscillations) behind the wave.

# Von Neumann Stability Analysis

## How to prove a method won't blow up

We assume the numerical error can be decomposed into a Fourier series. We test a single mode:  $u_j^n = G^n e^{ikj\Delta x}$

### The Amplification Factor ( $G$ )

After substituting the mode into our scheme, we solve for the ratio:

$$G = \frac{u^{n+1}}{u^n}$$

**Stability Condition:** The method is stable if and only if:

$$|G| \leq 1 \quad \forall \text{ wave numbers } k$$

For Advection Schemes:

- **Upwind:** Stable for  $0 \leq \nu \leq 1$ .
- **Lax-Wendroff:** Stable for  $|\nu| \leq 1$ .
- **Central Diff:**  $|G| > 1$  (Always Unstable!)



# Precision vs. Oscillations

Why we use 2nd-order schemes and how to fix them.

# The Numerical Trade-off

## Dissipation vs. Dispersion

In numerical methods, we are usually forced to choose between two types of errors:

### 1st Order (Upwind)

#### Numerical Dissipation

- **The "Smear":** Information is lost to artificial diffusion. The wave is smoothed out. Peaks lose height.
- **Physics:** High-frequency data (sharp edges) "melts" away.
- **Result:** No wiggles, but the solution vanishes over time.
- **No oscillations**

### 2nd Order (Lax-Wendroff)

#### Numerical Dispersion

- **The "Wiggle":** Different frequencies travel at different speeds (Phase errors).
- **Physics:** Energy and peak heights are conserved better. The wave keeps its height.
- **Result:** High precision, but trailing oscillations appear.

# Why is 2nd-Order Superior?

## Even with the Corners issues

If Upwind is so "clean," why bother with Lax-Wendroff?

- **Long-term Accuracy:** 1st-order schemes act as a low-pass filter. Over long distances, your signal will completely disappear.
- **Grid Efficiency:** To get the same accuracy as a 2nd-order grid of **100 points**, a 1st-order scheme might require **10,000 points**.
- **Conservation:** 2nd-order schemes maintain the "mass" and "peak" of your pulse much more effectively.

**The Lesser of Two Evils:** Most scientists prefer "wiggles" because they can see the data. When a solution is smeared, you don't know what you've lost.

# Godunov's Theorem

## The Fundamental Limit

Why can't we have a high-order linear scheme without wiggles?

**Godunov's Theorem:** "Linear numerical schemes for solving advection equations that do not create new under- or overshoots can be at most first-order accurate."

## The Solution: Non-Linearity

To get high precision without oscillations, we must use **Non-linear Limiter Schemes**.

- Near smooth regions: **High Order (2nd)**
- Near sharp corners: **Low Order (1st)**

# TVD & Flux Limiters

## Total Variation Diminishing

We "blend" the schemes using a **Limiter Function**  $\phi(r)$ :

$$f_{j+1/2} = f_{Low} + \phi(r) \cdot (f_{High} - f_{Low})$$

### Common Limiters

- **Min-Mod:** Conservative/Stable.
- **Superbee:** Aggressive/Sharp.
- **van Leer:** Smooth/Balanced.

### The Result

We achieve **High Resolution** on the plateaus and **Sharp Corners** without the dispersive wiggles. *This is the standard approach in modern CFD solvers.*

# Comprehensive Comparison of Advection Schemes

Scheme	Order	Stability ( $\nu$ )	Behavior at "Spigoli"	Main Error	Best Use Case
<b>Upwind</b>	1st	$0 \leq \nu \leq 1$	Smears / Melts	Dissipation	Simple tests, robust stability
<b>Lax-Friedrichs</b>	1st	$\nu \leq 1$	Heavy Smearing	Dissipation	General robust 1st-order solver
<b>Lax-Wendroff</b>	2nd	$\nu \leq 1$	Trailing Wiggles	Dispersion	Standard 2nd-order benchmark
<b>Beam-Warming</b>	2nd	$0 \leq \nu \leq 2$	Leading Wiggles	Dispersion	Purely directional upwind flows

**Numerical Dissipation (1st Order):** Information is lost as the wave "melts." The energy of the signal decreases over time, rounding off all sharp features.

**Numerical Dispersion (2nd Order):** Information is conserved, but the phase is shifted. Different wavelengths travel at different speeds, causing "wiggles."

*"The choice of scheme is a trade-off between losing information (Diffusion) and distorting information (Dispersion)."*

# Stability of the Solution

# Stability of the solution

We study the stability of discrete solutions for the transport equation:

Starting from the IVP (Initial Value Problem) with constant velocity  $c > 0$ :

$$\begin{cases} u_t + cu_x = 0, \\ u(x, 0) = u_0(x). \end{cases}$$

We consider the Upwind (UW) scheme:

$$v_j^{n+1} = v_j^n - c\lambda (v_j^n - v_{j-1}^n)$$

where  $\lambda = \frac{\Delta t}{\Delta x}$  is the mesh ratio.

# Von Neumann Hypothesis

Assume a harmonic solution of the form:

$$v_j^n = A_n e^{i\xi x_j}$$

Then:

$$v_{j-1}^n = A_n e^{i\xi(x_j - h)} = v_j^n e^{-i\xi h}$$

Define the amplification factor:

$$\boxed{z = \frac{A_{n+1}}{A_n}}$$

# Amplification Factor

Substituting the von Neumann ansatz into the upwind scheme:

$$\begin{aligned} zv_j^n &= v_j^n - c\lambda(v_j^n - v_{j-1}^n) \\ &= v_j^n - c\lambda(1 - e^{-i\xi h})v_j^n \end{aligned}$$

Therefore:

$$z = 1 - c\lambda(1 - e^{-i\xi h})$$

Let  $\theta = \xi h$

Since  $z$  does not depend on  $n$ :

$$v_j^n = z^n v_j^0 = z^n u_0(x_j)$$

# Real and Imaginary Parts of $z$

Using Euler's formula:  $e^{-i\theta} = \cos \theta - i \sin \theta$ , we obtain:

$$\begin{aligned} z &= 1 - c\lambda + c\lambda(\cos \theta - i \sin \theta) \\ &= \underbrace{(1 - c\lambda + c\lambda \cos \theta)}_{\Re(z)} - i \underbrace{(c\lambda \sin \theta)}_{\Im(z)} \end{aligned}$$

This decomposition allows us to analyze the **magnitude**  $|z|$ , which must be  $\leq 1$  for stability.

# Modulus of the Amplification Factor

For a complex number  $z = a + ib$ , the modulus is defined as:

$$|z| = \sqrt{a^2 + b^2}, \quad |z|^2 = a^2 + b^2$$

Thus, for our specific amplification factor:

$$|z|^2 = (1 - c\lambda + c\lambda \cos \theta)^2 + (c\lambda \sin \theta)^2$$

# Simplification of $|z|^2$

Expanding the terms and using the identity  $\sin^2 \theta + \cos^2 \theta = 1$ :

$$|z|^2 = (1 - c\lambda)^2 + 2c\lambda(1 - c\lambda) \cos \theta + (c\lambda)^2$$

Rewriting the expression:

$$|z|^2 = 1 - 2c\lambda(1 - c\lambda)(1 - \cos \theta)$$

# Stability Condition

Since we know that  $1 - \cos \theta \geq 0$  for all  $\theta$ , the stability condition  $|z| \leq 1$  (or  $|z|^2 \leq 1$ ) becomes:

$$2c\lambda(1 - c\lambda) \geq 0$$

Hence, we find the required range for  $c\lambda$ :

$$0 \leq c\lambda \leq 1$$

*This is the CFL (Courant-Friedrichs-Lowy) condition for the upwind scheme.*

# Exact Solution Comparison

For a harmonic initial condition:

$$u_0(x) = Ae^{i\xi x}$$
$$u(x, t) = Ae^{i\xi(x-ct)}$$

The exact amplification factor over one time step  $\Delta t$  is:

$$z_{\text{exact}} = e^{-i\xi c \Delta t}, \quad |z_{\text{exact}}| = 1$$

The ideal numerical scheme should satisfy  $|z| = 1$  to preserve the amplitude of the solution.

# Effect of the Amplification Factor

The discrete solution can be expressed as:

$$v_j^n = |z|^n e^{in\phi} e^{i\xi x_j}$$

The behavior of the scheme depends on the magnitude of  $z$ :

- $|z| < 1$ : stable but **numerically diffusive** (amplitude decreases)
- $|z| = 1$ : stable and **amplitude-preserving**
- $|z| > 1$ : **unstable** (amplitude grows exponentially)

# Stability of Common Schemes (Von Neumann)

Summary of stability conditions for the 1D transport equation:

- **Upwind:** stable  $\iff 0 \leq c\lambda \leq 1$
- **Forward Euler (FTCS):** always unstable
- **Backward Euler:** unconditionally stable (diffusive)
- **Crank-Nicolson:**  $|z| = 1$  (neutrally stable)
- **Lax-Friedrichs:** stable  $\iff c\lambda \leq 1$
- **Lax-Wendroff:** stable  $\iff c\lambda \leq 1$

*Note: Unconditionally stable schemes allow for larger time steps, but they may introduce significant numerical dissipation.*

# CFL Condition

# CFL Condition: Characteristics

We consider the linear transport equation:

$$u_t + cu_x = 0, \quad c \in \mathbb{R}$$

The exact solution is constant along characteristic curves  $x - ct = \alpha$ :

$$u(x, t) = u_0(x - ct)$$

*The value of the solution at a point  $(x, t)$  depends only on values taken along the same characteristic line at previous times.*

# Mathematical Domain of Dependence

Let  $x_j$  be a grid point and  $t_{n+1} = t_n + \Delta t$ . Tracing the characteristic backward in time:

$$x' - ct_n = x_j - ct_{n+1}$$

Which gives:

$$x' = x_j - c\Delta t$$

Thus, the value  $u(x_j, t_{n+1})$  depends on the interval:

$$I_{\text{MAT}} = [x_j - c\Delta t, x_j]$$

*This is the mathematical domain of dependence.*

# Numerical Domain of Dependence

The numerical domain of dependence is the set of grid points used to compute  $v_j^{n+1}$ . For the upwind scheme ( $c > 0$ ):

$$v_j^{n+1} = v_j^n - c\lambda(v_j^n - v_{j-1}^n)$$

The numerical domain of dependence is:

$$I_{\text{NUM}} = [x_{j-1}, x_j]$$

**CFL principle:**

$$I_{\text{NUM}} \supseteq I_{\text{MAT}}$$

# CFL Condition

From the domains of dependence:

$$x_j - c\Delta t \geq x_{j-1} = x_j - \Delta x$$

This implies  $c\Delta t \leq \Delta x$ , or equivalently:

$$c\lambda \leq 1, \quad \lambda = \frac{\Delta t}{\Delta x}$$

*This is the CFL condition for the upwind scheme.*

# Remark on the CFL Condition

The CFL condition is a **necessary** condition for stability, but in general it is **not sufficient**.

- Some unstable schemes satisfy CFL (e.g. FTCS for transport)
- Stability must be verified by von Neumann analysis

The CFL condition ensures that numerical information propagates at least as fast as physical information.

# Physical Interpretation

The CFL number  $c\lambda = c\frac{\Delta t}{\Delta x}$  represents the distance traveled by a wave during one time step, measured in units of the spatial grid size.

- $c\lambda < 1$ : wave travels less than one cell per time step
- $c\lambda = 1$ : wave travels exactly one cell
- $c\lambda > 1$ : numerical scheme cannot capture the propagation

# Boundary Conditions

In addition to initial conditions, boundary conditions are required. Let  $x_j, j = 1, \dots, N$ , be grid points with spacing  $\Delta x$ .

Boundary conditions must be imposed only where characteristics **enter** the computational domain.

# Direction of Propagation

Consider again the transport equation:  $u_t + cu_x = 0$ .

- **If  $c > 0$ :** information travels from left to right
- **If  $c < 0$ :** information travels from right to left

Boundary conditions are needed only at the **inflow boundary**.

# Upwind Boundary Conditions

**Case  $c > 0$  (right-moving wave):**

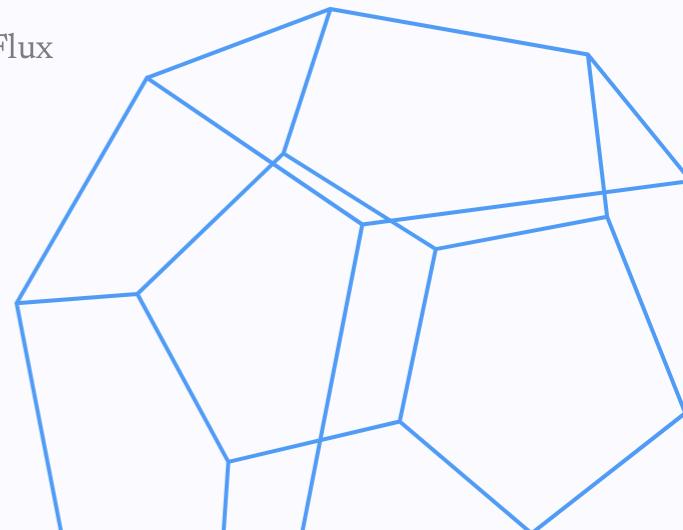
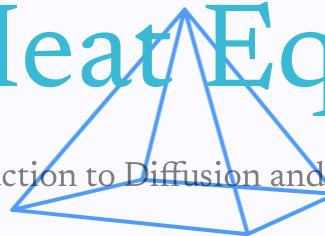
$$\begin{cases} v_1^n = g(t_n) & \text{(inflow)} \\ v_j^{n+1} = v_j^n - c\lambda(v_j^n - v_{j-1}^n), & j \geq 2 \end{cases}$$

**Case  $c < 0$  (left-moving wave):**

$$\begin{cases} v_N^n = g(t_n) & \text{(inflow)} \\ v_j^{n+1} = v_j^n - c\lambda(v_{j+1}^n - v_j^n), & j \leq N-1 \end{cases}$$

# The Heat Equation

An Introduction to Diffusion and Thermal Flux



# Physical Motivation

## What is Heat Diffusion?

The Heat Equation describes how temperature  $u(x, t)$  distributes itself in a medium over time. It is based on two fundamental physical principles:

### 1. Fourier's Law

Heat flux  $\vec{q}$  is proportional to the negative gradient of temperature:

$$\vec{q} = -k \nabla u$$

*Heat flows from hot to cold.*

### 2. Conservation of Energy

The change in internal energy in a volume must equal the heat flux through its boundaries:

$$\rho c_p \frac{\partial u}{\partial t} = -\nabla \cdot \vec{q}$$

*Energy is not created or destroyed.*

# The Mathematical Model

## From Physics to PDE

By combining Fourier's Law and Energy Conservation, we derive the **Parabolic** PDE:

The Standard Form:

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

$$u_t = Du_{xx}$$

Where:

- $u(x, t)$  is the Temperature
- $t$  is Time
- $x$  is Space
- $D$  is the **Thermal Diffusivity** ( $D = \frac{k}{\rho c_p}$ )

*The second spatial derivative acts as a "smoothing" operator.*

# Properties of the Heat Equation

## Qualitative Behavior

The Heat Equation is "Nature's Blur Tool." It has distinct mathematical properties:

### 1. Smoothing Effect

Even if the initial condition  $u_0(x)$  is "sharp" (like a box or a spike), for any  $t > 0$ , the solution becomes **infinitely differentiable ( $C^\infty$ )**.

### 2. Maximum Principle

The temperature inside the domain will never exceed the maximum temperature of the initial state or the boundaries. *No "spontaneous" hot spots.*

**Infinite Speed of Propagation:** A disturbance at one point is felt (theoretically) everywhere else instantly, though the magnitude decreases exponentially with distance.

# Boundary Conditions (BCs)

## Defining the Environment

To solve the Heat Equation on a finite domain  $[0, L]$ , we must specify what happens at the ends:

Type	Name	Physical Meaning
<b>Dirichlet</b>	$u(0, t) = T_0$	Fixed Temperature (e.g., end in ice water)
<b>Neumann</b>	$u_x(0, t) = 0$	Insulated End (no heat flux)
<b>Robin</b>	$u_x + \alpha u = g$	Convection (heat exchange with surroundings)

# Stability of the Heat Equation

Consider the heat equation:  $u_t = b^2 u_{xx}$ .

*This equation has **no characteristics**, since information propagates instantaneously.*

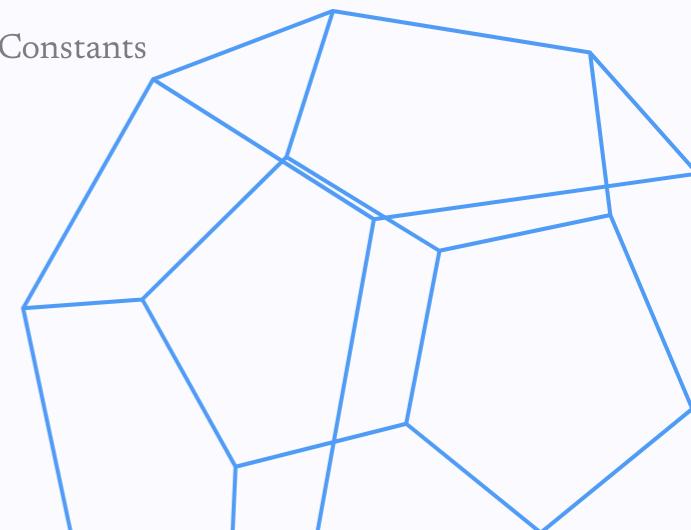
- The CFL condition cannot be derived via characteristics
- Stability must be studied purely by von Neumann analysis

Explicit schemes will impose a restrictive stability condition of the form:

$$\Delta t \leq C \Delta x^2$$

# The Diffusion Equation Solution

Solving with Fourier Transforms & Dimensional Constants



# Introduction: The Diffusion Model

We consider the one-dimensional diffusion equation with a diffusion coefficient  $D > 0$ .

The Governing System:

$$\begin{cases} u_t = Du_{xx}, & x \in \mathbb{R}, t > 0 \\ u(x, 0) = u_0(x), & x \in \mathbb{R} \end{cases}$$

The solution evolves as a convolution with the \*\*Gaussian Kernel\*\*:

$$G(x, t) = \frac{1}{\sqrt{4\pi Dt}} e^{-\frac{x^2}{4Dt}}$$

*Spreading and smoothing effects are controlled by the coefficient  $D$ .*

# General Solution

## Inverse Transform & Convolution

Returning to the physical domain via the Inverse Fourier Transform:

Substituting  $\hat{u}(k, t)$  into  $u(x, t) = \frac{1}{2\pi} \int_{-\infty}^{\infty} \hat{u}(k, t) e^{ikx} dk$ :

$$u(x, t) = \frac{1}{\sqrt{4\pi Dt}} \int_{-\infty}^{\infty} u_0(y) e^{-\frac{(x-y)^2}{4Dt}} dy$$

The solution is a **convolution** of the initial state  $u_0(x)$  with the Gaussian kernel. As  $t \rightarrow \infty$ , the initial features are smoothed out.

# Case 1: Dirac Delta $\delta(x)$

## The Fundamental Solution

If the initial condition is a concentrated pulse at the origin:  $u_0(x) = \delta(x)$ .

$$u(x, t) = \frac{1}{\sqrt{4\pi D t}} e^{-\frac{x^2}{4Dt}}$$

*Key Property:*

$$\text{Variance } \sigma^2 = 2Dt$$

Using the sifting property  $\int \delta(y) f(y) dy = f(0)$ , the Gaussian kernel **is** the solution for a point source.

## Case 2: The Box Function

### Diffusion of a Rectangular Pulse

Consider the initial condition  $u_0(x) = U$  for  $-L \leq x \leq L$ , and 0 elsewhere.

Solving the integral  $u(x, t) = \frac{U}{\sqrt{4\pi Dt}} \int_{-L}^L e^{-\frac{(x-y)^2}{4Dt}} dy$ :

$$u(x, t) = \frac{U}{2} \left[ \operatorname{erf}\left(\frac{x+L}{\sqrt{4Dt}}\right) - \operatorname{erf}\left(\frac{x-L}{\sqrt{4Dt}}\right) \right]$$

Where the Error Function is defined as:  $\operatorname{erf}(z) = \frac{2}{\sqrt{\pi}} \int_0^z e^{-t^2} dt$

*The corners of the box are immediately rounded as time progresses.*