Numerical methods to solve differential equations

Enrico Baglione

Outline

Differential Equations

Numerical methods to solve differential equations:

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

Example:

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

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 $\Delta t = k$:

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

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 $v^0, v^1, \ldots, v^n$ 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**:

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

Obtaining then a recursive formula:

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

1. Starting from v^0, I evaluate $v^1 = v^0 + k f^0$ 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:

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

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

↙

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**

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}:

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

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

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

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

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

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

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 tme 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:

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

ANALYTICAL SOLUTION

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

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

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

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:

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

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

RECURSIVE STEP

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

Runge-Kutta 2 (RK2)

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 α_{ij} denote the intermediate coefficients and b_i the final weights.

[1]

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

[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]

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

[s]

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

[END]

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

BUTCHER TABLEAU

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

General RK Structure

Runge–Kutta Coefficient Matrix (Butcher Tableau)

$$
\begin{array}{c|ccccc}
c_1 & & & & & \\
c_2 & \alpha_{21} & & & & \\
c_3 & \alpha_{31} & \alpha_{32} & & & \\
\cdots & \cdots & \cdots & \ddots & & \\
c_s & \alpha_{s1} & \alpha_{s2} & \cdots & \alpha_{s,s-1} & \\
\hline
 & b_1 & b_2 & \cdots & b_{s-1} & b_s
\end{array}
$$

c_i: stage time coefficients

α_{ij}: intermediate weights, $i = 1, \ldots, s, j = 1, \ldots, i - 1$

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)

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

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

$$\begin{array}{c|c} 0 & \\ \hline & 1 \end{array} \quad \rightarrow \quad v^{n+1} = v^n + k f^n$$

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]

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

[Heun formula, 2nd order]

HEUN (RK3 VARIANT)

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

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)

$$v^{n+1} = v^n + k(b_1 f^n + b_2 f^{n+c_2})$$

GENERAL 2-STAGE RK

$$\begin{array}{c|cc} 0 & & \\ c_2 & \alpha_{21} & \\ \hline & b_1 & b_2 \end{array}$$

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)

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

(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)

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

STABILITY ANALYSIS

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

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

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

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

Two equations for three unknowns (b_1, b_2, α_{21}).

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

Consistency Contraints

General row-sum constraint (consistency):

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

$$\Rightarrow$$

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

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:

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

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.

Linear Multistep

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

$$\alpha_{n+1}v^{n+1} + \alpha_n v^n + \cdots + \alpha_{n-s+1}v^{n-s+1} =$$
$$= k\left(\beta_{n+1}f^{n+1} + \beta_n f^n + \cdots + \beta_{n-s+1}f^{n-s+1}\right)$$

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:

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

Integral Formulation

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

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

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:

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

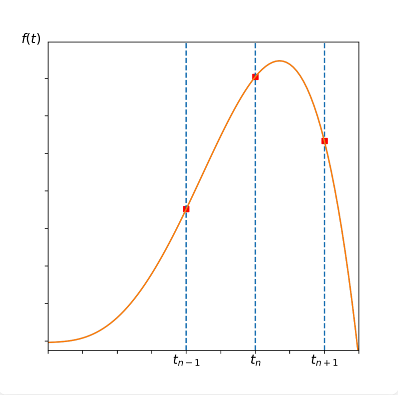
$$\Rightarrow \quad v^{n+1} = v^n + \int_{t_n}^{t_{n+1}} q(t) \, dt$$

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

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

$$\Rightarrow \quad v^{n+1} = v^n + kf^n$$

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

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

Depending on the points selected, q will be a polynomial of 1°, 2°, ... 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}):

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

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

Backward Differentiation – Order 1

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

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

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

(Explicit Euler)

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

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

(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:

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

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

$\Rightarrow$ $$\boxed{v^{n+1} = v^{n-1} + 2k f^n}$$ *(Explicit 2-step scheme)*

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

$\Rightarrow$ $$\boxed{v^{n+1} = -\tfrac{1}{3}v^{n-1} + \tfrac{4}{3}v^n + \tfrac{2}{3}k f^{n+1}}$$ *(Implicit 2-step scheme / BDF2)*

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 δ^- and δ^+. By defining $\delta^X = \delta^- \cdot \delta^+$, we obtain the second-order central difference:

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

Commutative Property:

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

The spatial analog is defined similarly:

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

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 δ_- 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:

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

This yields the central difference for the second derivative:

$\Rightarrow \boxed{\frac{\partial^2 u}{\partial x^2} = \delta_X v_j^n}$ *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).

UP-WIND (UW1)

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

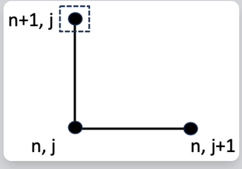
Starting from:

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

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

We obtain the update formula:

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



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

UP-WIND (UW2)

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

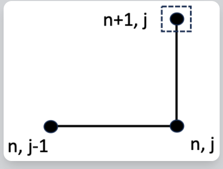
Starting from:

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

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

We obtain the update formula:

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



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: $u_t + cu_x = 0$.

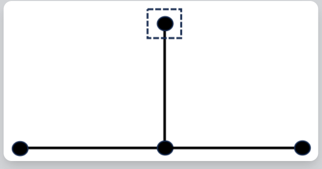
Starting from:

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

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

We obtain the update formula:

$$\boxed{v_j^{n+1} = v_j^n - \tfrac{1}{2}c\lambda\left(v_{j+1}^n - v_{j-1}^n\right)}$$



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

LEAP-FROG (LF)

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

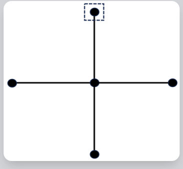
Starting from:

$$\boxed{\delta^0 v_j^n = -c\delta_0 v_j^n}$$

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

We obtain the update formula:

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



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:

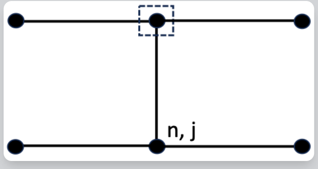
$$\boxed{\delta^+ v_j^n = -\tfrac{c}{2}\left(\delta_0 v_j^n + \delta_0 v_j^{n+1}\right)}$$

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

We obtain the update formula:

$$\boxed{v_j^{n+1} = v_j^n - \tfrac{1}{4}c\lambda\left(v_{j+1}^n - v_{j-1}^n + v_{j+1}^{n+1} - v_{j-1}^{n+1}\right)}$$

[implicit]



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

BACKWARD EULER (B-EU)

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

Starting from:

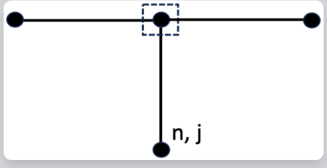
$$\boxed{\delta^+ v_j^n = -c\delta_0 v_j^{n+1}}$$

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

We obtain the update formula:

$$\boxed{v_j^{n+1} = v_j^n - \tfrac{1}{2}c\lambda\left(v_{j+1}^{n+1} - v_{j-1}^{n+1}\right)}$$

[implicit]



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

LAX-WENDROFF (LF)

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

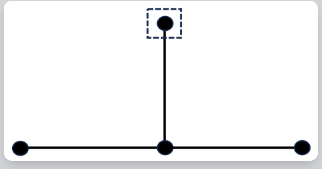
Starting from:

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

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

We obtain the update formula:

$$\boxed{v_j^{n+1} = v_j^n - \tfrac{1}{2}c\lambda\left(v_{j+1}^n - v_{j-1}^n\right) + \tfrac{1}{2}c^2\lambda^2\left(v_{j+1}^n - 2v_j^n + v_{j-1}^n\right)}$$



The Lax-Wendroff scheme adds numerical diffusion for stability.

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:

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

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:

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

Therefore:

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

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 ≤ 1 for stability.

Modulus of the Amplification Factor

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

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

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:

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

Stability Condition

Since we know that $1 - \cos\theta \geq 0$ for all θ, 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 $c\lambda$:

$$\boxed{0 \leq c\lambda \leq 1}$$

*This is the **CFL (Courant-Friedrichs-Lewy)** 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 Δt is:

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

> 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:

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

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: Characteristics

We consider the linear transport equation:

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

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:

$$\boxed{x' = x_j - c\Delta t}$$

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

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

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:

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

CFL principle:

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

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:

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

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 $x_j, j = 1, \ldots, N$, be grid points with spacing Δx.

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):

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

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

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

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:

$$\boxed{\Delta t \leq C \Delta x^2}$$