Numerical Differentiation

Computational Space Plasma Physics

Outline

- Differential Equations in Space Plasma Physics
- Computational Methods for solving
- Finite difference approximation
- A simple Matlab code

Differential Equations in Space Plasmas

Fluid Description

$$\frac{\partial}{\partial t}(n_\alpha) + \nabla \cdot (n_\alpha \mathbf{u}_\alpha) = S_\alpha$$

Mass conservation

Momentum conservation

$$\frac{\partial}{\partial t}(n_\alpha \mathbf{u}_\alpha) + \nabla \cdot (n_\alpha \mathbf{u}_\alpha \mathbf{u}_\alpha + \mathbf{P}_\alpha) - \frac{n_\alpha q_\alpha}{m_\alpha}(\mathbf{E} + \mathbf{u}_\alpha \times \mathbf{B}) = \mathbf{R}_\alpha$$

Thermal Dynamics

Maxwell's equation

$$\frac{\partial p_\alpha}{\partial t} + \mathbf{u}_\alpha \cdot \nabla p_\alpha + \gamma p_\alpha \nabla \cdot \mathbf{u}_\alpha = Q_\alpha \qquad \frac{\partial \mathbf{B}}{\partial \mathbf{t}} + \nabla \times \mathbf{E} = 0$$

$$\frac{\partial}{\partial t} : \ \text{time derivative}$$

$$\nabla : \ \text{spatial derivative}$$

$$= \frac{\partial}{\partial x}\hat{\mathbf{x}} + \frac{\partial}{\partial y}\hat{\mathbf{y}} + \frac{\partial}{\partial z}\hat{\mathbf{z}}$$

So the first key element of computation space plasma physics is to approximate these derivatives

Kinetic Description

Boltzmann equation

$$\frac{\partial f_s}{\partial t} + \nabla_\mathbf{x} \cdot (\mathbf{v} f_s) + \nabla_\mathbf{v} \cdot (\mathbf{F}_s f_s) = \left(\frac{\delta f_s}{\partial t}\right)_c$$

$$\mathbf{F}_s = q_s / m_s (\mathbf{E} + \mathbf{v} \times \mathbf{B})$$

Lorentz Force

$$\frac{\partial \mathbf{B}}{\partial \mathbf{t}} + \nabla \times \mathbf{E} = 0$$

Maxwell's equations

$$\mu_0 \epsilon_0 \frac{\partial \mathbf{E}}{\partial t} - \nabla \times \mathbf{B} = -\mu_0 \sum_s q_s \int_{-\infty}^{+\infty} v f_s d^3 v$$

Particle Description

$$m_s n_s \frac{d\mathbf{v}_s}{dt} = q n_s (\mathbf{E} + \mathbf{v_s} \times \mathbf{B})$$

Equation of motion

$$\frac{\partial \mathbf{B}}{\partial \mathbf{t}} + \nabla \times \mathbf{E} = 0$$

Maxwell's equations

$$\mu_0 \epsilon_0 \frac{\partial \mathbf{E}}{\partial t} - \nabla \times \mathbf{B} = -\mu_0 \sum_s q_s n_s \mathbf{v}_s$$

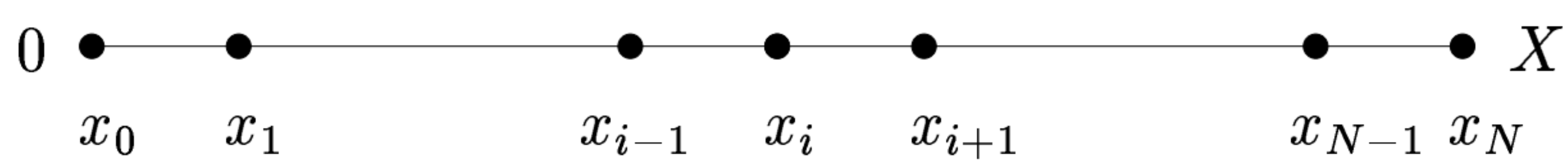
The most important equation for CPD

Taylor's expansion

Either the kinetic Vlasov equations of the fluid equations are solved numerically on a discrete set of spatial and temporal "grid points", this is called numerical discretization:

$$\text{1D:} \qquad \Omega = (0, X), \qquad u_i \approx u(x_i), \qquad i = 0, 1, \ldots, N$$

$$\text{grid points} \quad x_i = i\Delta x \qquad \text{mesh size} \quad \Delta x = \frac{X}{N}$$



Recall the definition of derivatives:

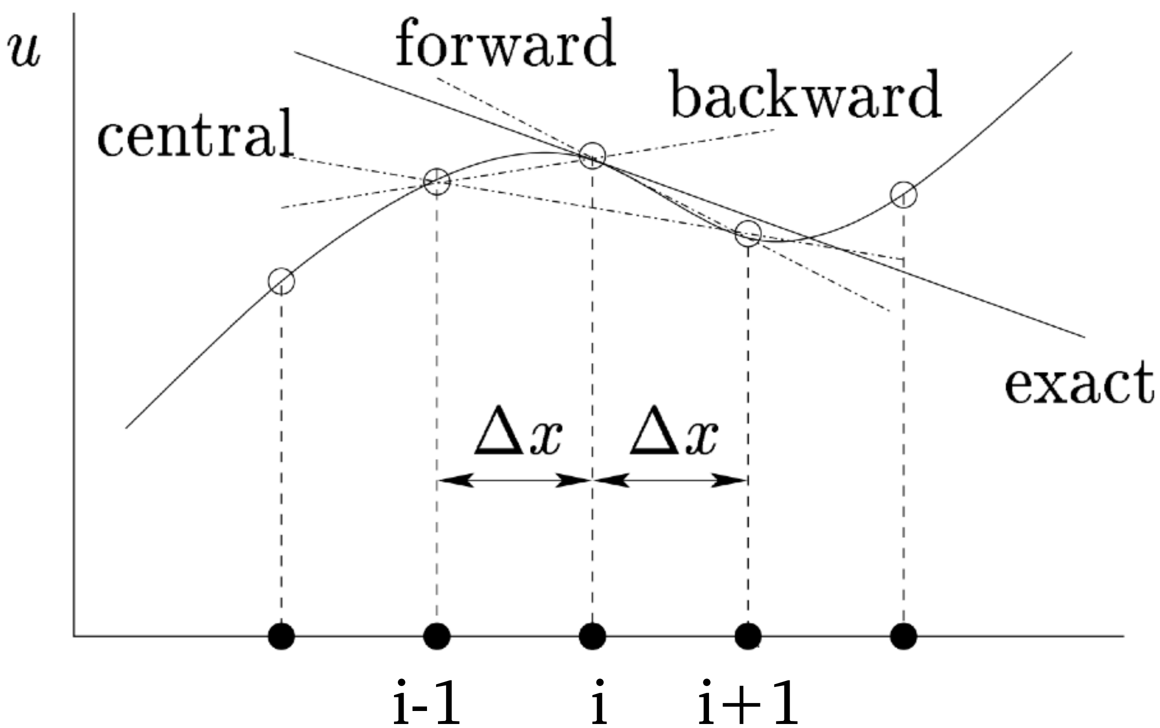
$$\left.\frac{dQ}{dx}\right|_{x=x_i} = \lim_{\Delta x \to 0} \frac{Q(x_i + \Delta x) - Q(x_i)}{\Delta x} = \lim_{\Delta x \to 0} \frac{Q(x_i) - Q(x_i - \Delta x)}{\Delta x}$$

$$= \lim_{\Delta x \to 0} \frac{Q(x_i + \Delta x) - Q(x_i - \Delta x)}{2\Delta x}$$

Numerical differentiation

Approximation of first-order derivatives

Geometric interpretation



$$\left(\frac{\partial u}{\partial t}\right)_i \approx \frac{u_{i+1} - u_i}{\Delta x}$$ Forward difference

$$\left(\frac{\partial u}{\partial t}\right)_i \approx \frac{u_i - u_{i-1}}{\Delta x}$$ Backward difference

$$\left(\frac{\partial u}{\partial t}\right)_i \approx \frac{u_{i+1} - u_{i-1}}{2\Delta x}$$ Central difference

Recall Tayler series expansion: $u(x) = \sum_{n=0}^{\infty} \frac{(x - x_i)^n}{n!} \left(\frac{\partial^n u}{\partial x^n}\right)_i$

Let's try two expansions around x_i

T1: $$u_{i+1} = u_i + \Delta x \left(\frac{\partial u}{\partial x}\right)_i + \frac{1}{2}\Delta x^2 \left(\frac{\partial^2 u}{\partial x^2}\right)_i + \frac{1}{6}\Delta x^3 \left(\frac{\partial^3 u}{\partial x^3}\right)_i + \dots$$

T2: $$u_{i-1} = u_i - \Delta x \left(\frac{\partial u}{\partial x}\right)_i + \frac{1}{2}\Delta x^2 \left(\frac{\partial^2 u}{\partial x^2}\right)_i - \frac{1}{6}\Delta x^3 \left(\frac{\partial^3 u}{\partial x^3}\right)_i + \dots$$

Numerical differentiation

Truncation errors

Using the Taylor series expansion, we can evaluate the accuracy of the finite difference

T1: $$u_{i+1} = u_i + \Delta x \left(\frac{\partial u}{\partial x}\right)_i + \frac{1}{2}\Delta x^2 \left(\frac{\partial^2 u}{\partial x^2}\right)_i + \frac{1}{6}\Delta x^3 \left(\frac{\partial^3 u}{\partial x^3}\right)_i + \dots$$

$$\Rightarrow \left(\frac{\partial u}{\partial x}\right)_i = \underbrace{\frac{u_{i+1} - u_i}{\Delta x}}_{\text{Forward difference approximation}} \underbrace{- \frac{1}{2}\Delta x \left(\frac{\partial^2 u}{\partial x^2}\right)_i - \frac{1}{6}\Delta x^2 \left(\frac{\partial^3 u}{\partial x^3}\right)_i + \dots}_{\text{Truncation Error, leading term } \mathcal{O}(\Delta x)}$$

Using T2, we can get the backward difference approximation of the first-derivative

$$\Rightarrow \left(\frac{\partial u}{\partial x}\right)_i = \underbrace{\frac{u_i - u_{i-1}}{\Delta x}}_{\text{Backward difference approximation}} \underbrace{+ \frac{1}{2}\Delta x \left(\frac{\partial^2 u}{\partial x^2}\right)_i - \frac{1}{6}\Delta x^2 \left(\frac{\partial^3 u}{\partial x^3}\right)_i + \dots}_{\text{Truncation Error, leading term } \mathcal{O}(\Delta x)}$$

Numerical differentiation

Truncation errors - central difference

Using the Taylor series expansion, we can evaluate the accuracy of the finite difference

T1: $$u_{i+1} = u_i + \Delta x \left(\frac{\partial u}{\partial x}\right)_i + \frac{1}{2}\Delta x^2 \left(\frac{\partial^2 u}{\partial x^2}\right)_i + \frac{1}{6}\Delta x^3 \left(\frac{\partial^3 u}{\partial x^3}\right)_i + \dots$$

T2: $$u_{i-1} = u_i - \Delta x \left(\frac{\partial u}{\partial x}\right)_i + \frac{1}{2}\Delta x^2 \left(\frac{\partial^2 u}{\partial x^2}\right)_i - \frac{1}{6}\Delta x^3 \left(\frac{\partial^3 u}{\partial x^3}\right)_i + \dots$$

T1 - T2:

$$\Rightarrow \left(\frac{\partial u}{\partial x}\right)_i = \frac{u_{i+1} - u_{i-1}}{2\Delta x} - \frac{1}{6}\Delta x^2 \left(\frac{\partial^3 u}{\partial x^3}\right)_i + \dots$$

Central difference approximation

Truncation Error, leading term $\mathcal{O}(\Delta x^2)$

Definition of leading truncation error term:

$$\epsilon_\tau = \alpha_m(\Delta x)^m + alpha_{m+1}(\Delta x)^{m+1} + \dots \approx \alpha_m(\Delta x)^m \rightarrow \mathcal{O}(\Delta x^m)$$

Numerical differentiation

Second-order derivatives

Using the Taylor series expansion, we can eliminate the first order derivative

T1: $$u_{i+1} = u_i + \Delta x \left(\frac{\partial u}{\partial x}\right)_i + \frac{1}{2}\Delta x^2 \left(\frac{\partial^2 u}{\partial x^2}\right)_i + \frac{1}{6}\Delta x^3 \left(\frac{\partial^3 u}{\partial x^3}\right)_i + \dots$$

T2: $$u_{i-1} = u_i - \Delta x \left(\frac{\partial u}{\partial x}\right)_i + \frac{1}{2}\Delta x^2 \left(\frac{\partial^2 u}{\partial x^2}\right)_i - \frac{1}{6}\Delta x^3 \left(\frac{\partial^3 u}{\partial x^3}\right)_i + \dots$$

T1 + T2:

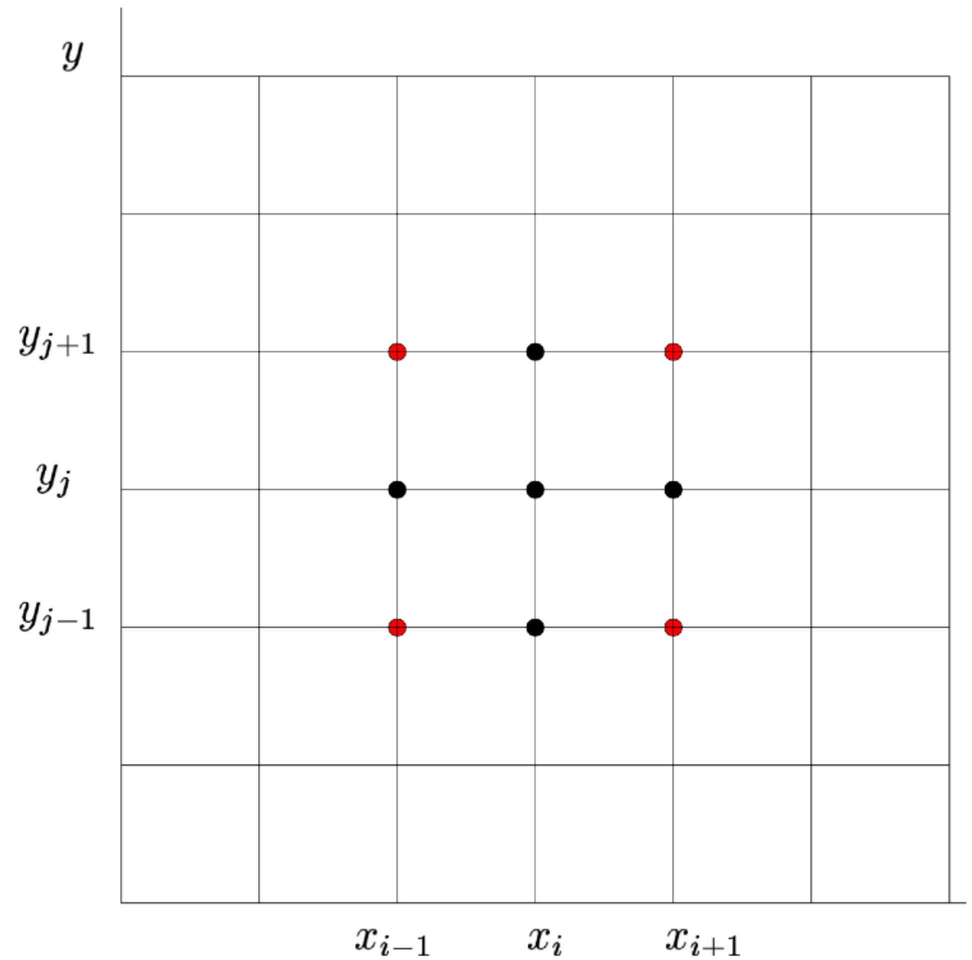
$$\Rightarrow \left(\frac{\partial^2 u}{\partial x^2}\right)_i = \frac{u_{i+1} - 2u_i + u_{i-1}}{\Delta x^2} - \frac{1}{6}\Delta x^2 \left(\frac{\partial^3 u}{\partial x^3}\right)_i + \dots$$

$$= \underbrace{\frac{u_{i+1} - 2u_i + u_{i-1}}{\Delta x^2}}_{\text{Central difference approximation}} + \underbrace{\mathcal{O}(\Delta x^2)}_{\text{Truncation Error}}$$

Alternatively: $$\left(\frac{\partial^2 u}{\partial x^2}\right)_i = \left[\frac{\partial}{\partial x}\left(\frac{\partial u}{\partial x}\right)_i\right] \approx = \lim_{\Delta x \to 0} \frac{\left(\frac{\partial u}{\partial x}\right)_{i+\frac{1}{2}} - \left(\frac{\partial u}{\partial x}\right)_{i-\frac{1}{2}}}{\Delta x}$$

Numerical differentiation

Mixed derivatives



$$\frac{\partial^2 u}{\partial x \partial y} = \frac{\partial}{\partial x}\left(\frac{\partial u}{\partial y}\right) = \frac{\partial}{\partial y}\left(\frac{\partial u}{\partial x}\right)$$

$$\left(\frac{\partial^2 u}{\partial x \partial y}\right)_{i,j} = \frac{\left(\frac{\partial u}{\partial y}\right)_{i+1,j} - \left(\frac{\partial u}{\partial y}\right)_{i-1,j}}{2\Delta x} + \mathcal{O}(\Delta x^2)$$

In a 2-D mesh, we know that

$$\left(\frac{\partial u}{\partial y}\right)_{i+1,j} = \frac{u_{i+1,j+1} - u_{i+1,j-1}}{2\Delta y} + \mathcal{O}(\Delta y^2)$$

$$\left(\frac{\partial u}{\partial y}\right)_{i-1,j} = \frac{u_{i-1,j+1} - u_{i-1,j-1}}{2\Delta y} + \mathcal{O}(\Delta y^2)$$

So the second-order mix derivative is

$$\left(\frac{\partial^2 u}{\partial x \partial y}\right)_{i,j} = \frac{u_{i+1,j+1} - u_{i+1,j-1} - u_{i-1,j+1} + u_{i-1,j-1}}{4\Delta x \Delta y} + \mathcal{O}(\Delta x^2) + \mathcal{O}(\Delta y^2)$$

Numerical differentiation

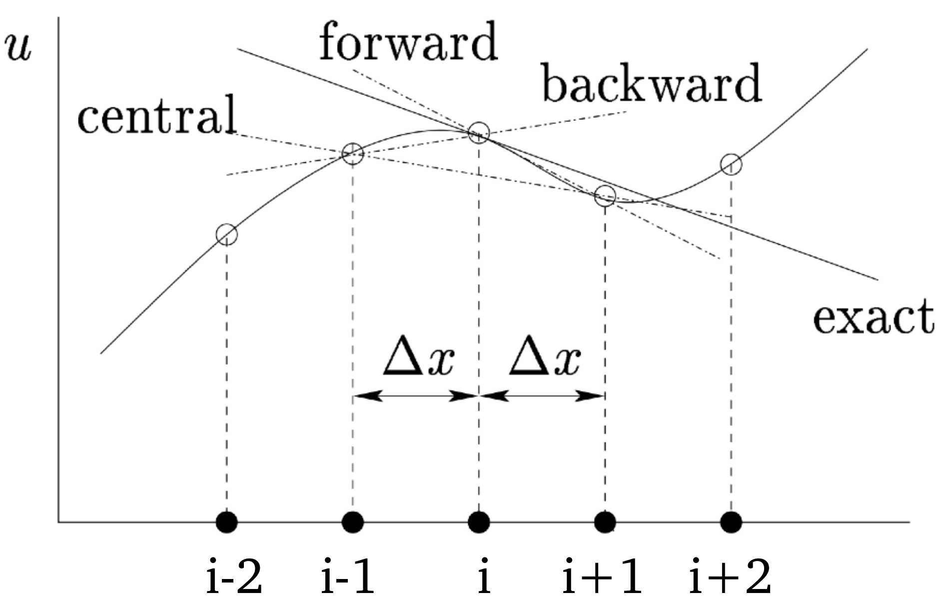
High-order approximations

Central difference $\Rightarrow \left(\frac{\partial u}{\partial x}\right)_i = \frac{u_{i+1} - u_{i-1}}{2\Delta x} + \mathcal{O}(\Delta x^2)$

We used two neighboring points to get a second-order accuracy, can we go higher?

The trick is in the Taylor series of expansion

$$u_{i+1} = u_i + \Delta x \left(\frac{\partial u}{\partial x}\right)_i + \frac{1}{2}\Delta x^2 \left(\frac{\partial^2 u}{\partial x^2}\right)_i + \frac{1}{6}\Delta x^3 \left(\frac{\partial^3 u}{\partial x^3}\right)_i + \dots$$



$$u_{i+1} \approx u_i + \Delta x \left(\frac{\partial u}{\partial x}\right)_i + \frac{1}{2}\Delta x^2 \left(\frac{\partial^2 u}{\partial x^2}\right)_i + \frac{1}{6}\Delta x^3 \left(\frac{\partial^3 u}{\partial x^3}\right)_i + \frac{1}{24}\Delta x^4 \left(\frac{\partial^4 u}{\partial x^4}\right)_i$$

$$u_{i-1} \approx u_i - \Delta x \left(\frac{\partial u}{\partial x}\right)_i + \frac{1}{2}\Delta x^2 \left(\frac{\partial^2 u}{\partial x^2}\right)_i - \frac{1}{6}\Delta x^3 \left(\frac{\partial^3 u}{\partial x^3}\right)_i + \frac{1}{24}\Delta x^4 \left(\frac{\partial^4 u}{\partial x^4}\right)_i$$

$$u_{i+2} \approx u_i + 2\Delta x \left(\frac{\partial u}{\partial x}\right)_i + \frac{1}{2}(2\Delta x)^2 \left(\frac{\partial^2 u}{\partial x^2}\right)_i + \frac{1}{6}(2\Delta x)^3 \left(\frac{\partial^3 u}{\partial x^3}\right)_i + \frac{1}{24}(2\Delta x)^4 \left(\frac{\partial^4 u}{\partial x^4}\right)_i$$

$$u_{i-2} \approx u_i - 2\Delta x \left(\frac{\partial u}{\partial x}\right)_i + \frac{1}{2}(2\Delta x)^2 \left(\frac{\partial^2 u}{\partial x^2}\right)_i - \frac{1}{6}(2\Delta x)^3 \left(\frac{\partial^3 u}{\partial x^3}\right)_i + \frac{1}{24}(2\Delta x)^4 \left(\frac{\partial^4 u}{\partial x^4}\right)_i$$

Numerical differentiation

High-order approximations

Now we get four equations with four unknown:

$$u_{i+1} \approx u_i + \Delta x\left(\frac{\partial u}{\partial x}\right)_i + \frac{1}{2}\Delta x^2\left(\frac{\partial^2 u}{\partial x^2}\right)_i + \frac{1}{6}\Delta x^3\left(\frac{\partial^3 u}{\partial x^3}\right)_i + \frac{1}{24}\Delta x^4\left(\frac{\partial^4 u}{\partial x^4}\right)_i$$

$$u_{i-1} \approx u_i - \Delta x\left(\frac{\partial u}{\partial x}\right)_i + \frac{1}{2}\Delta x^2\left(\frac{\partial^2 u}{\partial x^2}\right)_i - \frac{1}{6}\Delta x^3\left(\frac{\partial^3 u}{\partial x^3}\right)_i + \frac{1}{24}\Delta x^4\left(\frac{\partial^4 u}{\partial x^4}\right)_i$$

$$u_{i+2} \approx u_i + 2\Delta x\left(\frac{\partial u}{\partial x}\right)_i + \frac{1}{2}(2\Delta x)^2\left(\frac{\partial^2 u}{\partial x^2}\right)_i + \frac{1}{6}(2\Delta x)^3\left(\frac{\partial^3 u}{\partial x^3}\right)_i + \frac{1}{24}(2\Delta x)^4\left(\frac{\partial^4 u}{\partial x^4}\right)_i$$

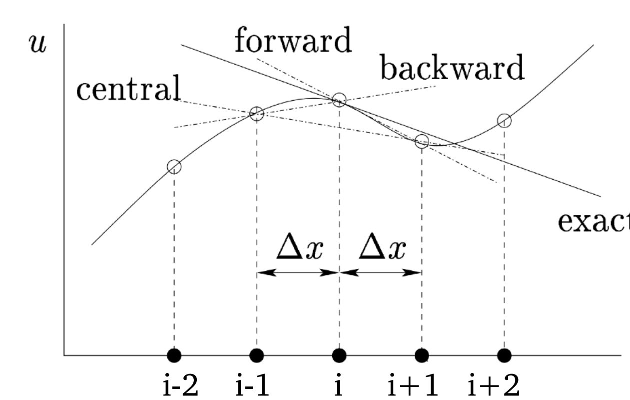
$$u_{i-2} \approx u_i - 2\Delta x\left(\frac{\partial u}{\partial x}\right)_i + \frac{1}{2}(2\Delta x)^2\left(\frac{\partial^2 u}{\partial x^2}\right)_i - \frac{1}{6}(2\Delta x)^3\left(\frac{\partial^3 u}{\partial x^3}\right)_i + \frac{1}{24}(2\Delta x)^4\left(\frac{\partial^4 u}{\partial x^4}\right)_i$$

Eliminate the second, third and forth derivatives from the Taylor series, we have

$$\Rightarrow \left(\frac{\partial u}{\partial x}\right)_i \approx \frac{-u_{i+2} + 8u_{i+1} - 8u_{i-1} + u_{i-2}}{12\Delta x} + \mathcal{O}(\Delta x^4)$$

Pros and Cons of high-order differentiation:

- ⊖ more grid points, fill-in, considerable overhead cost
- ⊕ high resolution, reasonable accuracy on coarse grids



A simple example

Linear Advection Equation

Time derivative

$$\frac{\partial}{\partial t}(n_\alpha) + \nabla \cdot (n_\alpha \mathbf{u}_\alpha) = 0 \xrightarrow[\nabla = \frac{\partial}{\partial x}\hat{\mathbf{x}} \qquad \mathbf{u}_\alpha = u_0 \hat{\mathbf{x}}]{\textbf{1-D}} \frac{\partial n_\alpha}{\partial t} + u_0 \frac{\partial n_\alpha}{\partial x} = 0$$

Spatial derivative

u_0 is a constant, this equation is simply a linear PDE about n_\alpha

Now let's define a new notation for discretized n in both space and time

$$n_\alpha(t,x) \xrightarrow[\text{call it } Q]{\text{ignore } \alpha} Q(t,x) \xrightarrow{t = t_n \; x = x_i} Q(t = t_n, x = x_i) \xrightarrow{\text{define}} Q_i^n$$

the n^{th} timestep

the i^{th} grid point

Then let's take a look how to discretize the differential equation

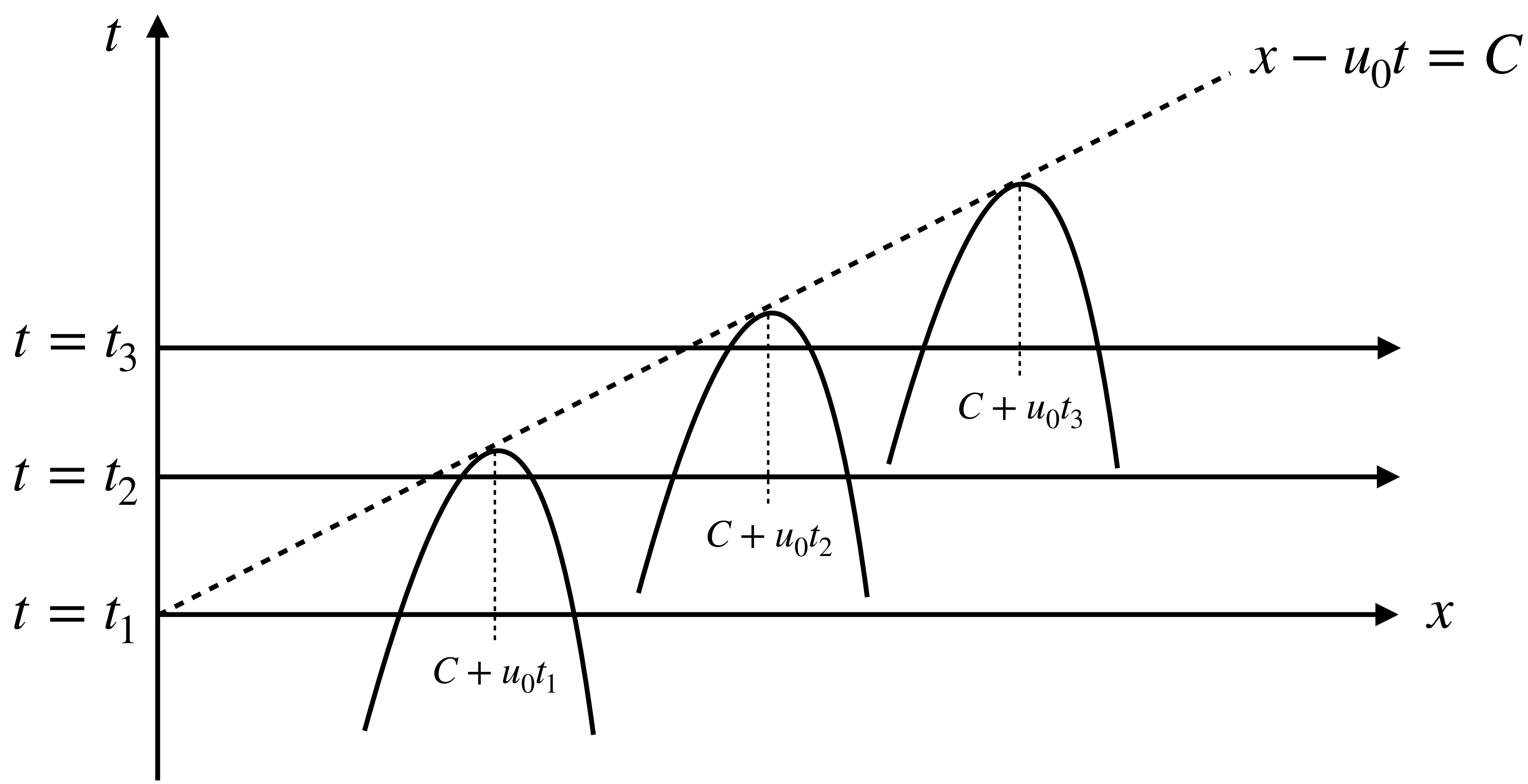
$$\frac{\partial Q}{\partial t} + u_0 \frac{\partial Q}{\partial x} = 0$$

first let's take a look at what the solution look like from the mathematical aspect

What does the linear advection equation do?

$$\frac{\partial Q(x,t)}{\partial t} + u_0 \frac{\partial Q(x,t)}{\partial x} = 0$$

The solution goes like $Q(x,t) \sim f(x - u_0 t)$



A simple wave propagation towards +x direction:

- When $u_0 > 0$, to keep $(x-u_0t)$ constant, x ***increases*** with t -> wave propagates towards right
- When $u_0 < 0$, to keep $(x-u_0t)$ constant, x ***decreases*** with t -> wave propagates towards left

A simple example

$$\left(\frac{\partial u}{\partial t}\right)_i \approx \frac{u_{i+1} - u_i}{\Delta x}$$

Backward difference

Linear Advection Equation

$$\frac{\partial Q}{\partial t} + u_0 \frac{\partial Q}{\partial x} = 0$$

Forward difference

Forward difference

Euler Time-Stepping

$$\left.\frac{\partial Q}{\partial t}\right|_{t=t_n} = \frac{Q_i^{n+1} - Q_i^n}{\Delta t} + \mathcal{O}(\Delta t)$$

$$\left.\frac{\partial Q}{\partial t}\right|_{x=x_i} = \frac{Q_{i+1}^n - Q_i^n}{\Delta x} + \mathcal{O}(\Delta x)$$

Combine the two numerical derivatives

$$\frac{Q_i^{n+1} - Q_i^n}{\Delta t} = u_0 \frac{Q_{i+1}^n - Q_i^n}{\Delta x} + \mathcal{O}(\Delta x) + \mathcal{O}(\Delta t) \longrightarrow$$

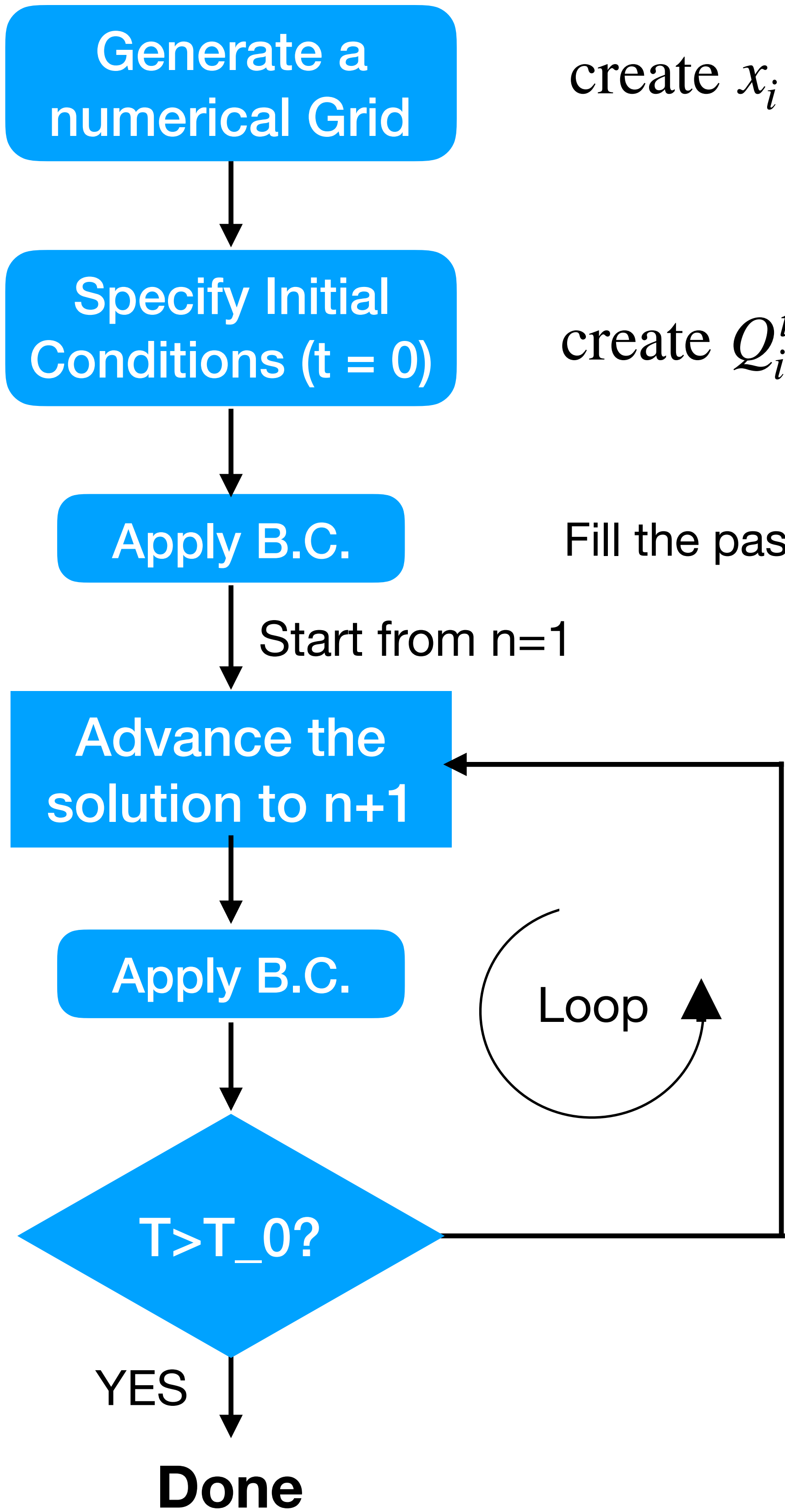
Forward Euler method

$$\boxed{Q_i^{n+1} \approx Q_i^n + \frac{u_0 \Delta t}{\Delta x}(Q_{i+1}^n - Q_i^n)}$$

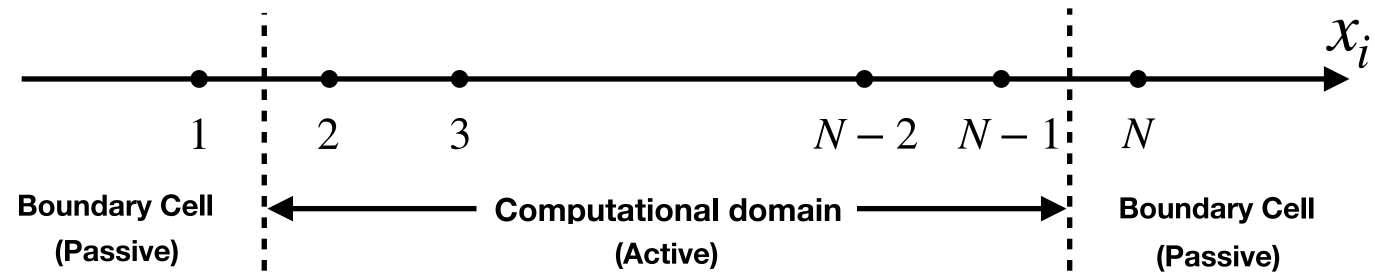
Values on step n (known)

If we know all the Q values at time t = t_n, then we can calculate the Q values at t = t_n+1. This is known as an explicit scheme of first-order accuracy

How to convert an algorithm to a Code



create x_i



create $Q_i^{t=0}$ e.g., $Q_i^{t=0} = e^{-\frac{(x_i - x_0)^2}{\sigma^2}}$

Fill the passive cells for derivatives at the end points

$$Q_i^{n+1} \approx Q_i^n + \frac{u_0 \Delta t}{\Delta x}(Q_{i+1}^n - Q_i^n)$$

Fill the passive cells for derivatives at the end points

Determine whether reaching the end time (or other criteria)

A simple Code

```
% explicit linear advection code 1-D

x = linspace(0,1,101); % create a computational grid between 0 and 1 with 100 grid points
% alternatively you can do this: dx = 0.01; x = 0:dx:100
dx = x(2)-x(1);
Ni = length(x);

u0 = 1; % advection speed;
sigma = 0.05;
Q = exp(-(x-0.5).^2/sigma^2); % initial profile of Q
Q_init = Q; % save the initial profile for the final plot

% impose boundary conditions here - periodic
Q(1) = Q(Ni-1);
Q(Ni) = Q(2);

dt = 0.01; % FIXME: time step should follow wave propagation
Time = 0; % keep track on the time step

figure('position',[442   668   988   280]) % create a blank figure to show the advection results

for n=1:200 % the time stepping loop - let's try advect 200 steps maximum

    Time = Time + dt; % the current time

    for i=2:Ni-1 % index stops at Ni-1 because we need i+1
        Q(i) = Q(i) + u0*dt/dx.*(Q(i+1)-Q(i)); % our first forward-difference scheme!
    end

    % This is called a periodic boundary condition
    Q(1) = Q(Ni-1);
    Q(Ni) = Q(2);

    % plot the Q profile as a function of x
    plot(x,Q);
    xlabel('x'),ylabel('Q'),title(['Simulation Time = ',num2str(Time)]);
    set(gca,'fontsize',14); % make the font better for visualization
    pause(0.1)

    if(Time > 1)
        break; % stop the time-loop if simulation time reaches 1
    end

end
```

Grid Generation x_i

Initialization $Q_i^{t=0}$

Boundary Conditions

$$Q_i^{n+1} \approx Q_i^n + \frac{u_0 \Delta t}{\Delta x}(Q_{i+1}^n - Q_i^n)$$

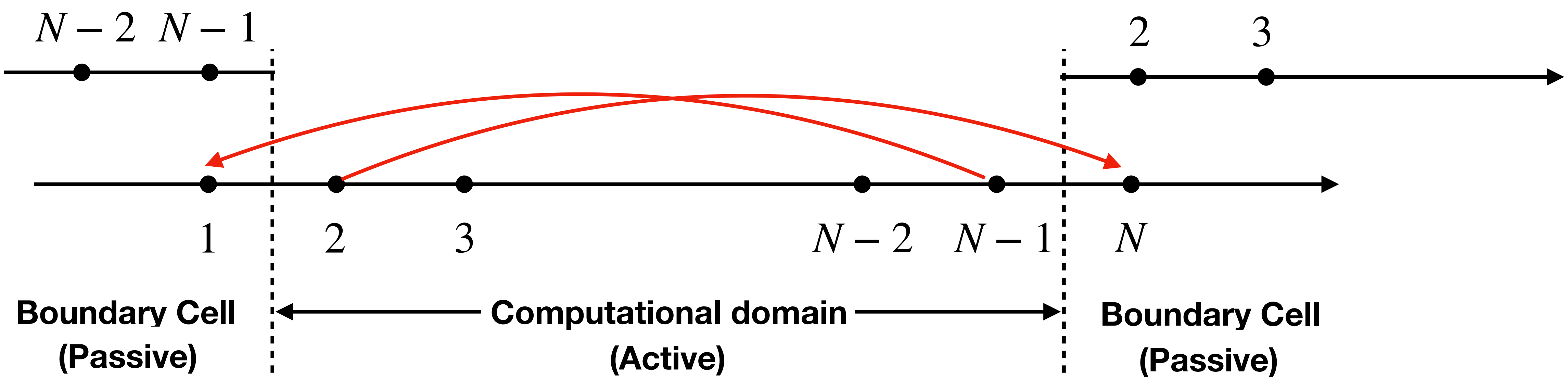
Forward Difference scheme

Boundary Conditions

The time stepping loop

Visualization

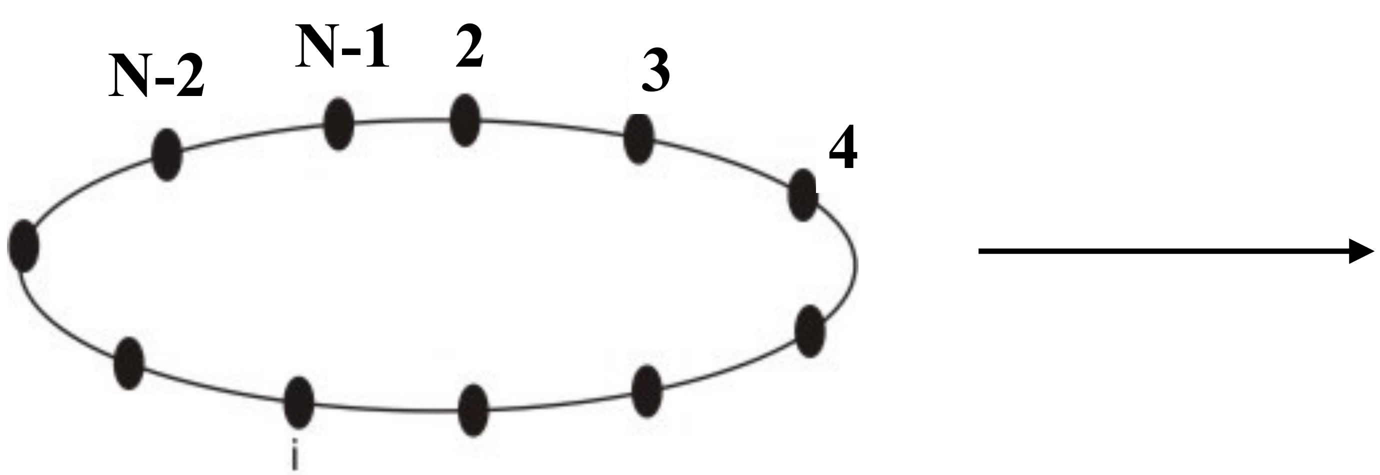
Periodic Boundary condition



Boundary cells are need for computing the spatial derivatives $\left.\frac{\partial Q}{\partial t}\right|_{x=x_i} = \frac{Q^n_{i+1} - Q^n_i}{\Delta x} + \mathcal{O}(\Delta x)$

How the boundary cells are filled is called the "boundary conditions" (b.c.)

In the sample code we used a special type of boundary condition called periodic b.c.



$$Q(1) = Q(N-1)$$

$$Q(N) = Q(2)$$

What if we use other numerical derivatives?

Linear Advection Equation

$$\left(\frac{\partial u}{\partial t}\right)_i \approx \frac{u_i - u_{i-1}}{\Delta x}$$

Backward difference

$$\frac{\partial Q}{\partial t} + u_0 \frac{\partial Q}{\partial x} = 0$$

Forward difference

$$\left.\frac{\partial Q}{\partial t}\right|_{t=t_n} = \frac{Q_i^{n+1} - Q_i^n}{\Delta t} + \mathcal{O}(\Delta t)$$

Forward difference

$$\left.\frac{\partial Q}{\partial t}\right|_{x=x_i} = \frac{Q_i^n - Q_{i-1}^n}{\Delta x} + \mathcal{O}(\Delta x)$$

Now use the backward spatial difference:

$$\frac{Q_i^{n+1} - Q_i^n}{\Delta t} = u_0 \frac{Q_i^n - Q_{i-1}^n}{\Delta x} + \mathcal{O}(\Delta x) + \mathcal{O}(\Delta t) \longrightarrow$$

Backward Euler method

$$\boxed{Q_i^{n+1} \approx Q_i^n + \frac{u_0 \Delta t}{\Delta x}(Q_i^n - Q_{i-1}^n)}$$

What about the Central Euler Method? High-Order Euler Method?

Do they work?