Numerical Simulation for Fluid Dynamics

Takayuki Aoki

Global Scientific Information and Computing Center

Tokyo Institute of Technology

Table of Contents

2nd Order Linear Partial Differential Equation

$$A\frac{\partial^2 f}{\partial x^2}+B\frac{\partial^2 f}{\partial x\partial y}+C\frac{\partial^2 f}{\partial y^2}+D\frac{\partial f}{\partial x}+E\frac{\partial f}{\partial y}+Ff+G=0$$

Elliptic	$B^2-4AC<0$
Parabolic	$B^2-4AC=0$
Hyperbolic	$B^2-4AC>0$

Characteristics of the partial differential equation is determined by the highest order derivative term.

Typical Equations

- **Elliptic Equation**

  Poisson Eq.

  (A = C = 1, B = 0)

$$\frac{\partial^2 f}{\partial x^2}+\frac{\partial^2 f}{\partial y^2}=\rho$$

- **Parabolic Equation**

  Diffusion Eq.

  （A=κ, B = C = 0）

$$\frac{\partial f}{\partial t}=\kappa\frac{\partial^2 f}{\partial x^2}$$

- **Hyperbolic Equation**

  Wave Eq.

  (A = $-c^2$, B = 0, C = 1)

$$\frac{\partial^2 f}{\partial t^2}-c^2\frac{\partial^2 f}{\partial x^2}=0$$

Partial Differential Equation in space

・Boundary Problems (Boundary Conditions)

1D Poisson Eq. $$\frac{d^2 f}{dx^2} = \rho = const \quad (0 \le x \le 1)$$

Boundary Condition : $f(0) = 0, \quad f(1) = 0$

2D Poisson Eq.

$$\frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} = \rho = const \quad (x \in S)$$

Boundary Conditions : f is given at the surface - S

Differential Equation in time

・Initial Condition

Time goes in one way

Newton Eq. : $$\frac{d^2 \boldsymbol{r}}{dt^2} = \boldsymbol{F} \quad (t_0 \le t \le t_1)$$

Initial Condition: $\boldsymbol{r}(t_0) = \boldsymbol{r_0}, \quad \dot{\boldsymbol{r}}(t_0) = \dot{\boldsymbol{r}}_0$

: $$\frac{d\boldsymbol{v}}{dt} = \boldsymbol{F}, \quad \frac{d\boldsymbol{r}}{dt} = \boldsymbol{v} \quad (t_0 \le t \le t_1)$$

Initial Condition: $\boldsymbol{r}(t_0) = \boldsymbol{r_0}, \quad \boldsymbol{v}(t_0) = \boldsymbol{v_0}$

Partial Differential Equation in time and space

GSIC

- Initial Boundary Problem

One-dimensional diffusion equation:

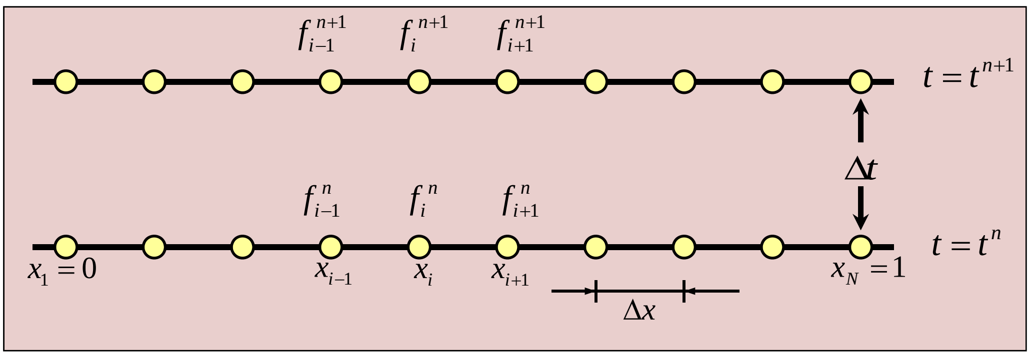
$$\frac{\partial f}{\partial t} = \kappa \frac{\partial^2 f}{\partial x^2} \qquad (0 \le x \le 1)$$

Initial Condition: $f(x,0) = f_0(x) \qquad (0 \le x \le 1)$

Boundary Condition: $f(0,t) = 0$, $f(1,t) = 0$

Discretization (Notation)

GSIC

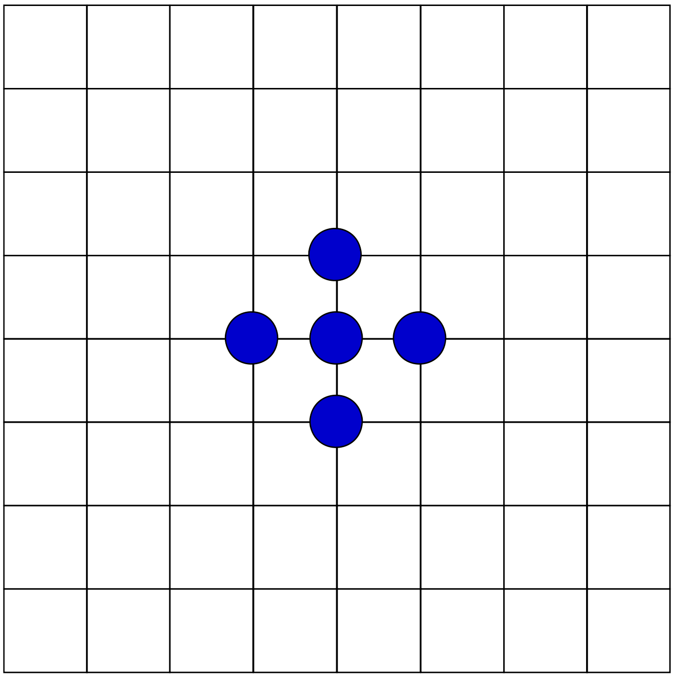


$f(x_i, t_n) = f_i^n$

$f(x_i + \Delta x,\ t_n + \Delta t) = f_{i+1}^{n+1}$

Game

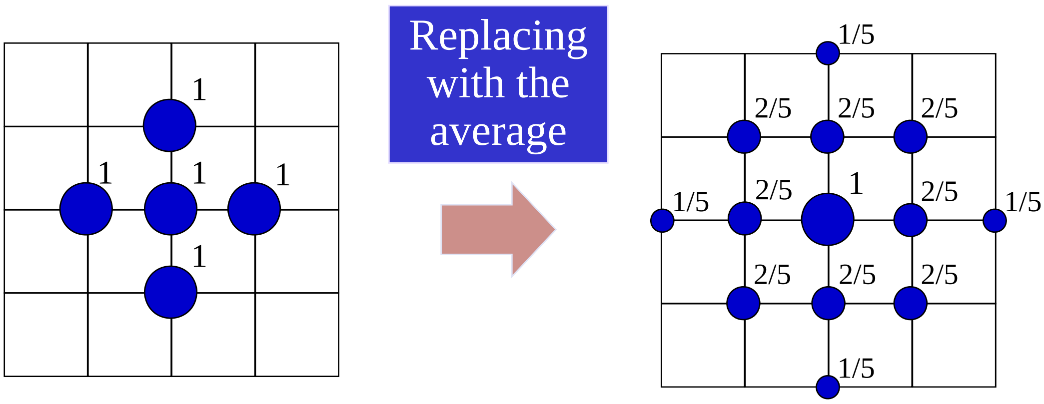
Rule: get a new value by replacing the average about surroundings



Initial

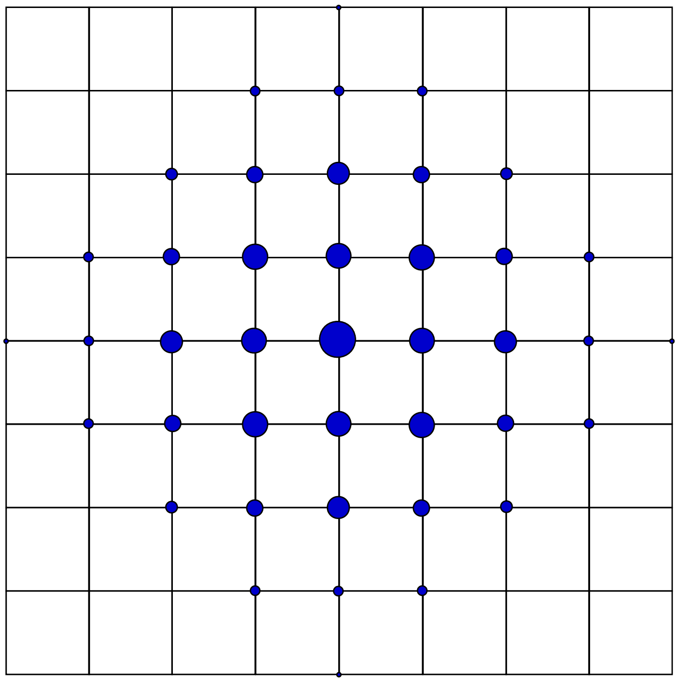
Game

Rule: get a new value by replacing the average about surroundings

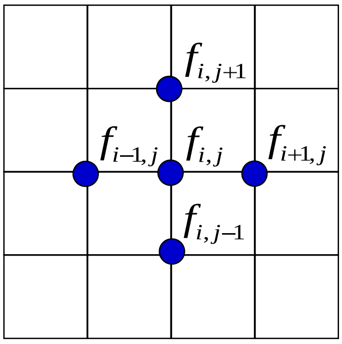


Game

Rule: get a new value by replacing the average about surroundings



Game



i : grid index in the x-direction

j : grid index in the y-direction

This average process is

$$f^*_{i,j} = \frac{f_{i,j} + f_{i+1,j} + f_{i-1,j} + f_{i,j+1} + f_{i,j-1}}{5}$$

Game

$$f_{i,j}^{n+1} = \frac{f_{i,j}^{n} + f_{i+1,j}^{n} + f_{i-1,j}^{n} + f_{i,j+1}^{n} + f_{i,j-1}^{n}}{5}$$

n : present value

n+1 : the value after average

By subtracting $f_{i,j}$ from both side

$$f_{i,j}^{n+1} - f_{i,j}^{n} = \frac{f_{i+1,j}^{n} - 2f_{i,j}^{n} + f_{i-1,j}^{n} + f_{i,j+1}^{n} - 2f_{i,j}^{n} + f_{i,j-1}^{n}}{5}$$

Game

When we regard $\Delta t = 1.0$, $\Delta x = \Delta y = 1.0$, $\kappa = 1/5$,

$$\frac{f_{i,j}^{n+1} - f_{i,j}^{n}}{\Delta t} = \kappa \frac{f_{i+1,j}^{n} - 2f_{i,j}^{n} + f_{i-1,j}^{n}}{\Delta x^2} + \kappa \frac{f_{i,j+1}^{n} - 2f_{i,j}^{n} + f_{i,j-1}^{n}}{\Delta y^2}$$

2-dimendional diffusion equation

$$\frac{\partial f}{\partial t} = \kappa \left(\frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} \right)$$

Parabolic Equation

GSIC

1-dimeisional diffusion eq.

$$\frac{\partial \phi}{\partial t} = \kappa \frac{\partial^2 \phi}{\partial x^2}$$

κ : diffusion coefficient

Image of diffusion equation: spreading distribution with fading out.

→ **Increasing entropy**

Particle Collision Process from the microscopic view

Thermal Conduction:	**Electron Collision**
Viscosity:	**Ion Collision**
Nuclear Reactor:	**Neutron Collision**

1-dimensional
Diffusion Equation

GSIC

$$\frac{\partial \phi}{\partial t} = \kappa \frac{\partial^2 \phi}{\partial x^2}$$

κ : diffusion coefficient

Applying the forward finite difference to the time derivative term and the center finite difference to the spatial difference,

$$\frac{\phi_j^{n+1} - \phi_j^n}{\Delta t} = \kappa \frac{\phi_{j+1}^n - 2\phi_j^n + \phi_{j-1}^n}{\Delta x^2}$$

We can reduce to

$$\phi_j^{n+1} = \phi_j^n + \mu\left(\phi_{j+1}^n - 2\phi_j^n + \phi_{j-1}^n\right)$$

where $\mu = \frac{\kappa \Delta t}{\Delta x^2}$

Sample Program 1

```
#include "xwin.h"
#define    N   101

int  main() {
      double f[N],  fn[N],  x[N],  mu = 0.25;
      int    j,  icnt = 0;

      while(icnt < 100) {
                for(j=1; j < N-1; j++) {
                      fn[j] = f[j] + mu*(f[j+1] - 2.0*f[j] + f[j-1]);
                }
                for(j=0; j < N; j++) f[j] = fn[j];     /* updating */
      }
}
```

Source Code

Stability Analysis (1/3)

von Neumann's Method:

Assuming the perturbation $\phi_j^n = \delta\phi^n \, e^{ik\cdot j\Delta x}$

Where the notation *i* is the imaginary, *k* is the wave number, *j* is grid index, *j*Δ*x* is the grid position.

Substituting into $\phi_j^{n+1} = \phi_j^n + \mu\left(\phi_{j+1}^n - 2\phi_j^n + \phi_{j-1}^n\right)$

$$\delta\phi^{n+1} e^{ik\cdot j\Delta x} = \delta\phi^n e^{ik\cdot j\Delta x} + \mu\left(\delta\phi^n e^{ik\cdot (j+1)\Delta x} - 2\delta\phi^n e^{ik\cdot j\Delta x} + \delta\phi^n e^{ik\cdot (j-1)\Delta x}\right)$$

Stability Analysis (2/3)

$n+1$ step／n step : amplitude ratio

$$\delta\phi_j^{n+1} / \delta\phi_j^n = 1 + \mu\left(e^{-ik\Delta x} - 2 + e^{ik\Delta x}\right)$$
$$= 1 - 2\mu(1 - \cos k\Delta x)$$

$$\cos k\Delta x = \frac{e^{ik\Delta x} + e^{-ik\Delta x}}{2}$$

$\left|\delta\phi^{n+1} / \delta\phi^n\right| < 1$: The amplitude of the perturbation decrease in time.

→ The calculation is stable.

Stability Analysis (3/3)

Amplitude ratio: $\delta\phi_j^{n+1} / \delta\phi_j^n = 1 - 2\mu(1 - \cos k\Delta x)$

$\mu < 0$:	unstable,
$0 < \mu < 1/2$:	stable
$1/2 < \mu$:	unstable depending on the value k

We consider only the case of $0 < \kappa$,

$$\mu < \frac{1}{2} \quad \Rightarrow \quad \Delta t < \frac{1}{2} \frac{\Delta x^2}{\kappa}$$

We have to choose Δt satisfying the condition, but Δt should be decrease propotionally to Δx^2 with decrease of Δx.

Hyperbolic Equation

Wave equation (Typical hyperbolic equation)

$$\frac{\partial^2 \phi}{\partial t^2} - u^2 \frac{\partial^2 \phi}{\partial x^2} = 0$$

By factorizing as $\left(\frac{\partial}{\partial t} - u\frac{\partial}{\partial x}\right)\left(\frac{\partial}{\partial t} + u\frac{\partial}{\partial x}\right)\phi = 0$

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

The advection equation is the simplest hyperbolic equation.

Advection Equation

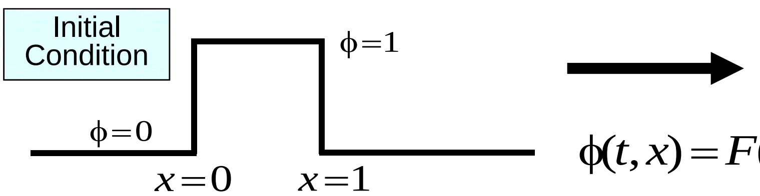
$$\frac{\partial \phi}{\partial t} + u\frac{\partial \phi}{\partial x} = 0$$

Analytic Solution：

$$\phi = F(x - ut)$$

$F(x)$ is an arbitrary function of x .
Confirming $f(t,x) = F(t\text{-}\Delta t, x\text{-}u\Delta t)$, we can understand the Profile $F(x)$ moves with the speed u.

● Let's solve the advection equation as an initial boundary problem.



$$\phi(t,x) = F(t - \Delta t, x - u\Delta t)$$

Simple
Finite Difference Method

At the time $t = t^n$ and the position $x = x_i$

■Time derivative $\frac{\partial \phi}{\partial t} \rightarrow \frac{\phi_i^{n+1} - \phi_i^n}{\Delta t}$ (Forward Difference)

■Spatial derivative $\frac{\partial \phi}{\partial x} \rightarrow \frac{\phi_{i+1}^n - \phi_{i-1}^n}{2\Delta x}$ (Center Difference)

Discretized form:

$$\frac{\phi_i^{n+1} - \phi_i^n}{\Delta t} = -u \frac{\phi_{i+1}^n - \phi_{i-1}^n}{2\Delta x}$$

CFL Number

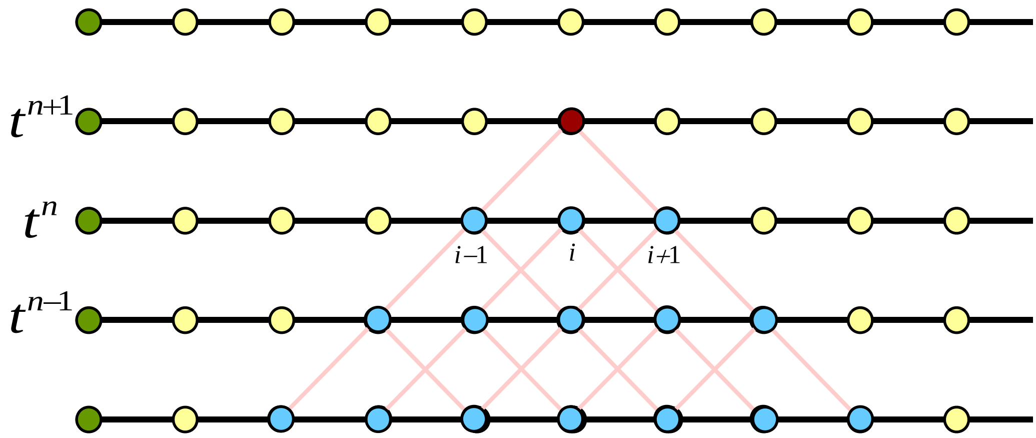
$$\phi_i^{n+1} = \phi_i^n - u\Delta t \frac{\phi_{i+1}^n - \phi_{i-1}^n}{2\Delta x}$$

$$= \phi_i^n - \frac{1}{2} C \left(\phi_{i+1}^n - \phi_{i-1}^n \right)$$

$C = u\Delta t / \Delta x$ is called CFL (Courant-Friedrich-Levy) number.

$$C = \frac{\text{Physical traveling distance } u\Delta t}{\text{Numerical traveling distance } \Delta x}$$

Time and Space Cone for Information travel



If C > 1, the scheme does not have enough information to obtain the value of the next time

Requirement:

The condition $C_{(CFL\ \ number)} \leq 1$ must be satisfied.

Sample Program 2

```
#include "program1.h"
#define    N   101

int  main() {

      double f[N],  fn[N],  x[N],  cfl = 0.5;
      int    j,  icnt = 0;

      while(icnt < 100) {
               for(j=1; j < N-1; j++) {
                    fn[j] = f[j] - cfl*0.5*(f[j+1] - f[j-1]);
               }
               for(j=0; j < N; j++) f[j] = fn[j];              /* updating */
      }
}
```

Source Code

For the advection equation

Stability Analysis (1/2)

von Neumann's Method

Assuming the perturbation $\phi_j^n = \delta\phi^n \, e^{ik \cdot j\Delta x}$

Substituting into $\phi_j^{n+1} = \phi_j^n - \frac{1}{2} C\left(\phi_{j+1}^n - \phi_{j-1}^n\right)$

C : CFL number

$$\delta\phi^{n+1} e^{ik \cdot j\Delta x} = \delta\phi^n e^{ik \cdot j\Delta x} - \frac{1}{2} C\left(\delta\phi^n e^{ik \cdot (j+1)\Delta x} - \delta\phi^n e^{ik \cdot (j-1)\Delta x}\right)$$

For the advection equation

Stability Analysis (2/2)

$$\delta\phi^{n+1} \big/ \delta\phi^n = 1 - \frac{1}{2} C\left(e^{ik\Delta x} - e^{-ik\Delta x}\right)$$
$$= 1 - C\, i \sin k\Delta x$$

$$\left|\delta\phi^{n+1} / \delta\phi^n\right| = \sqrt{1 + C^2 \sin^2 k\Delta x}$$

Analytically $\left|\delta\phi^{n+1} / \delta\phi^n\right| = 1$

The above formula shows $\left|\delta\phi^{n+1} / \delta\phi^n\right| \geq 1$

It is unstable to apply the forward difference in time and the center difference in space to the advection equation.

Lax Scheme（1/2）

$$\phi_j^{n+1} = \phi_j^n - \frac{1}{2} C \left(\phi_{j+1}^n - \phi_{j-1}^n \right)$$

$$\phi_j^n \to \frac{1}{2} \left(\phi_{j+1}^n + \phi_{j-1}^n \right)$$

$$\phi_j^{n+1} = \frac{1}{2} \left(\phi_{j+1}^n + \phi_{j-1}^n \right) - \frac{1}{2} C \left(\phi_{j+1}^n - \phi_{j-1}^n \right)$$

ϕ_j^n is replaced by the average of ϕ_{j+1}^n and ϕ_{j-1}^n

The result is expected to be diffusive.

Sample Program 3

```
#include "program1.h"
#define    N   101

int  main() {

      double f[N],  fn[N],  x[N],  cfl = 0.5;
      int     j,  icnt = 0;

      while(icnt < 100) {
            for(j=1; j < N-1; j++) {
                  fn[j] = 0.5*(f[j+1] – f[j-1])
                        - cfl*0.5*(f[j+1] - f[j-1]);
            }
            for(j=0; j < N; j++) f[j] = fn[j];     /* updating */
      }
}
```

Source Code

Lax Scheme（2/2）

von Neumann's stability analysis:

$$\delta\phi^{n+1} / \delta\phi^{n} = \cos k\Delta x - iC \sin k\Delta x$$

$$\left|\delta\phi^{n+1} / \delta\phi^{n}\right| = \sqrt{\cos^2 k\Delta x + C^2 \sin^2 k\Delta x}$$

It is understood that the scheme is stable for $C \leq 1$

Rewriting to

$$\phi_j^{n+1} = \phi_j^n - \frac{1}{2} C\left(\phi_{j+1}^n - \phi_{j-1}^n\right) + \frac{1}{2}\left(\phi_{j+1}^n - 2\phi_j^n + \phi_{j-1}^n\right)$$

Lax-Wendroff Scheme（1/2）

Taylor expansion in time:

$$\phi_j^{n+1} = \phi_j^n + \Delta t \frac{\partial \phi}{\partial t} + \frac{1}{2}\Delta t^2 \frac{\partial^2 \phi}{\partial t^2} + O(\Delta t^3)$$

By using the relations of the advection equation,

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

$$\phi_j^{n+1} = \phi_j^n - u\Delta t \frac{\partial \phi}{\partial x} + \frac{1}{2}(u\Delta t)^2 \frac{\partial^2 \phi}{\partial x^2} + O(\Delta t^3)$$

We apply the center difference to the spatial derivative term, we can reduce

Lax-Wendroff Scheme（2/2）

$$\phi_j^{n+1} = \phi_j^n - \frac{1}{2}C\left(\phi_{j+1}^n - \phi_{j-1}^n\right) + \frac{1}{2}C^2\left(\phi_{j+1}^n - 2\phi_j^n + \phi_{j-1}^n\right)$$

The 2nd center finite difference term stabilizes the scheme by working diffusively.

von Neumann's stability analysis:

$$\left|\delta\phi_j^{n+1} / \delta\phi_j^n\right| = \sqrt{1 - C^2(1-\cos k\Delta x)^2 + C^2 \sin^2 k\Delta x} \leq 1$$

The scheme is the second order in both time and space.

Sample Program 4

```
#include "program1.h"
#define    N    101

int  main() {

      double f[N],  fn[N],  x[N],  cfl = 0.5;
      int     j,  icnt = 0;

      while(icnt < 100) {
                for(j=1; j < N-1; j++) {
                      fn[j] = f[j] – 0.5*cfl*(f[j+1] - f[j-1])
                           + 0.5*cfl*cfl*(f[j+1] - f[j] + f[j-1]);
                }
                for(j=0; j < N; j++) f[j] = fn[j];      /* updating */
      }
}
```

Source Code

1st order upwind scheme

Since the solution of the advection equation is the profile moving with the velocity u from the upwind to the down direction, it is natural that we apply the backward finite difference to the advection term.

$$\phi_j^{n+1} = \phi_j^n - u\frac{\phi_j^n - \phi_{j-1}^n}{\Delta x}\Delta t$$
$$= \phi_j^n - C\left(\phi_j^n - \phi_{j-1}^n\right)$$

Sample Program 5

```
#include "program1.h"
#define    N   101

int  main() {
      double f[N],  fn[N],  x[N],  cfl = 0.5;
      int     j,  icnt = 0;

      while(icnt < 100) {
                for(j=1; j < N-1; j++) {
                     fn[j] = f[j] – cfl*(f[j] - f[j-1]);
                }
                for(j=0; j < N; j++) f[j] = fn[j];      /* updating */
      }
}
```

Source Code

1st order upwind scheme

$$\phi_j^{n+1} = \phi_j^n - C\left(\phi_j^n - \phi_{j-1}^n\right)$$
$$= \phi_j^n - \frac{C}{2}\left(\phi_{j+1}^n - \phi_{j-1}^n\right) + \frac{C}{2}\left(\phi_{j+1}^n - 2\phi_{j-1}^n + \phi_{j-1}^n\right)$$

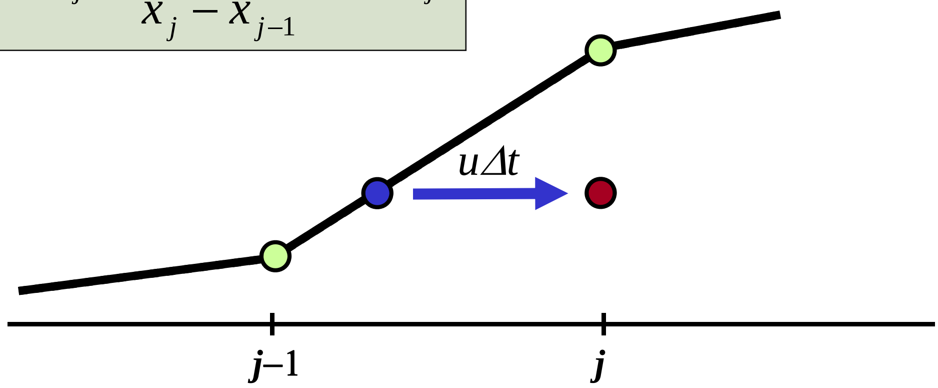
von Neumann's stability analysis:

$$\left|\delta\phi_j^{n+1} / \delta\phi_j^n\right| = \sqrt{\left(1 - C + C\cos k\Delta x\right)^2 + C^2 \sin^2 k\Delta x}$$
$$\leq 1$$

Other Interpretation

Linear interpolation for the upwind region $x_{j-1} \leq x \leq x_j$

$$F^n(x) = \phi_j^n + \frac{\phi_j^n - \phi_{j-1}^n}{x_j - x_{j-1}}(x - x_j)$$



At the time $t^n + \Delta t$, $f_j^{n+1} = F^n(x_j - u\Delta t)$ $x = x_j - u\Delta t$