雷射電漿物理暑期學校
Laser-Plasma Interaction Summer School
7/4 ~ 7/9, 2005
中研院原分所浦大邦講堂

數值模擬的基本知識

呂凌霄

國立中央大學

太空科學研究所

課程大綱

1. 什麼是數值模擬？

Definition of Numerical Simulation

Numerical simulations use a set of numerical methods to solve a set of ordinary differential equations and/or partial differential equations, in which at least one time-derivative term is present in these equations. Thus, numerical simulation is a set of carefully planed numerical schemes to solve initial value problems numerically.

{Numerical Simulations $\subset$ Numerical Methods}

Comprehensive knowledge on numerical methods can help us to make a good diagnostics of the simulation results.

Objectives of Making Numerical Simulation Studies

The goal of using numerical simulations to study nonlinear phenomena includes:

- To test theoretical model to see if one can reproduce the observed phenomena based on the given theoretical model.
- To understand detail nonlinear evolution processes and to determine the complicated cause-and-result relationships.
- To make a prediction or forecasting of the observed phenomena (applications of numerical simulations).

2. 如何選擇電漿數值模擬碼

How to Choose a Suitable Simulation Code for Your Problem

Plasma consists of positive charged ions and negative charged electrons. Since $m_i >> m_e$, there are many intrinsic time scales in a plasma system even in a uniform background environment. For time scale less than the intrinsic time scales, it is hard for the plasma to reach a thermal dynamic equilibrium state. As a result, the kinetic effect might become important for time scale less than the intrinsic time scales of the plasma.

Present plasma simulation codes can be classified based on their phase space resolutions as listed in Table 2.1. Note that the so-called *particle-code simulation* is indeed a *multiple-fluid simulation*. A *simulation particle* in the particle-code simulation is indeed a *fluid element* in the phase space $(\mathbf{x}, \mathbf{v})$.

Basic equations of a relativistic test particle simulation code, a relativistic full-particle code, and a relativistic Vlasov simulation code are given in Tables, 2.2, 2.3, and 2.4, respectively.

Table 2.1. Classification of Plasma Simulation Codes

	Simulation Code	Phenomena scale length λ	Assuming thermal dynamic equilibrium?		
			e-e	i-i	e-i
Fluid Simulations	MHD code	$\lambda \geq 10^3 \lambda_i$	yes	yes	yes
	Two-Fuild code	$10^3 \lambda_i \geq \lambda \geq 10\lambda_i$	yes	yes	no
Kinetic Simulations	Hybrid code *fluid electrons & kinetic ions*	$10\lambda_i \geq \lambda \geq \lambda_i$	yes	no	no
	Full particle code	$\lambda_i \geq \lambda \geq \lambda_e$	no	no	no
	Test particle code	Strong magnetic field	n/a	n/a	n/a
	Vlasov Code	$\lambda_i \geq \lambda \geq \lambda_e$	no	no	no

Table 2.2. Equations of motion of a relativistic test particle α, with mass m_α and charge e_α, moving in a background electric field $\mathbf{E}(\mathbf{x})$ and magnetic field $\mathbf{B}(\mathbf{x})$

$\frac{d\mathbf{x}_\alpha(t)}{dt} = \frac{\mathbf{u}_\alpha(t)}{\sqrt{1+[u_\alpha(t)/c]^2}}$
$\frac{d\mathbf{u}_\alpha(t)}{dt} = \frac{e_\alpha}{m_\alpha}[\mathbf{E}(\mathbf{x}) + \frac{\mathbf{u}_\alpha(t)}{\sqrt{1+[u_\alpha(t)/c]^2}} \times \mathbf{B}(\mathbf{x})]_{\mathbf{x}=\mathbf{x}_\alpha(t)}$

Table 2.3. Governing equations of relativistic electromagnetic particle code simulation (the simulation particle is a finite-size particle with shape function S)

Equation of motion of simulation particles: $$\frac{d\mathbf{x}_\alpha(t)}{dt} = \frac{\mathbf{u}_\alpha(t)}{\sqrt{1+[u_\alpha(t)/c]^2}}$$ $$\frac{d\mathbf{u}_\alpha(t)}{dt} = \frac{e_\alpha}{m_\alpha}\int[\mathbf{E}(\mathbf{x},t) + \frac{\mathbf{u}_\alpha(t)}{\sqrt{1+[u_\alpha(t)/c]^2}} \times \mathbf{B}(\mathbf{x},t)]S[\mathbf{x}-\mathbf{x}_\alpha(t)]d\mathbf{x}$$
Maxwell's equations: $$\nabla\cdot\mathbf{E}(\mathbf{x},t) = \int\sum_\alpha \frac{e_\alpha}{\varepsilon_0}\delta(\mathbf{x}'-\mathbf{x}_\alpha)S(\mathbf{x}-\mathbf{x}')d\mathbf{x}'$$ $$\nabla\cdot\mathbf{B} = 0$$ $$\frac{\partial\mathbf{B}(\mathbf{x},t)}{\partial t} = -\nabla\times\mathbf{E}(\mathbf{x},t)$$ $$\nabla\times\mathbf{B}(\mathbf{x},t) = \mu_0\int\sum_\alpha e_\alpha \frac{\mathbf{u}_\alpha(t)}{\sqrt{1+[u_\alpha(t)/c]^2}}\delta[\mathbf{x}'-\mathbf{x}_\alpha(t)]S(\mathbf{x}-\mathbf{x}')d\mathbf{x}' + \frac{1}{c^2}\frac{\partial\mathbf{E}(\mathbf{x},t)}{\partial t}$$

Table 2.4. Governing equations of electromagnetic Vlasov simulation code with relativistic electrons and non-relativistic ions

$\frac{\partial f_e}{\partial t} = -\frac{\mathbf{u}}{\sqrt{1+(u/c)^2}}\cdot\frac{\partial f_e}{\partial\mathbf{x}} + \frac{e}{m_e}(\mathbf{E} + \frac{\mathbf{u}}{\sqrt{1+(u/c)^2}}\times\mathbf{B})\cdot\frac{\partial f_e}{\partial\mathbf{u}}$
$\frac{\partial f_i}{\partial t} = -\mathbf{v}\cdot\frac{\partial f_i}{\partial\mathbf{x}} - \frac{e}{m_i}(\mathbf{E}+\mathbf{v}\times\mathbf{B})\cdot\frac{\partial f_i}{\partial\mathbf{v}}$
$\frac{\partial\mathbf{B}}{\partial t} = -\nabla\times\mathbf{E}$
$\frac{\partial\mathbf{E}}{\partial t} = c^2\nabla\times\mathbf{B} - \frac{e}{\varepsilon_0}[\iiint\mathbf{v}f_i d^3v - \iiint\frac{\mathbf{u}}{\sqrt{1+(u/c)^2}}f_e d^3u]$
Initial conditions $\nabla\cdot\mathbf{B} = 0$ $\nabla\cdot\mathbf{E} = \frac{e}{\varepsilon_0}[\iiint f_i d^3v - \iiint f_e d^3u]$

A fluid simulation code can provide reasonable and quick simulation results when the kinetic effects are unimportant. The magnetohydrodynamic (MHD) simulation code is good to simulate large-scale nonlinear plasma phenomena. The Hall MHD simulation code and the two-fluid simulation code are good to simulate medium-scale nonlinear plasma phenomena, in which dispersion effect due to finite ion inertial length effect is important.

The kinetic effect becomes important when the non-uniformity scale length of the system is comparable to the characteristic scale length of a species (ions and/or electrons), or *when the wave speed observed in the center of mass frame of a species is approximately equal or less than the thermal speed of that species*. When the kinetic effect is important, we have to use a kinetic simulation code to study the nonlinear evolutions of wave-particle interactions in the phase space.

3. 處理微分與積分常用的數值方法

Numerical Methods for Differentiations and Integrations

As we have discussed in Section 1 that numerical simulation is a set of carefully planed numerical schemes to solve initial value problems numerically. Let us consider the following three types of differential equations,

$$\frac{dy(t)}{dt} = f(t) \tag{3.1}$$

$$\frac{dy(t)}{dt} = f(y,t) \tag{3.2}$$

$$\frac{\partial y(x,t)}{\partial t} = f(t, y, \frac{\partial y}{\partial x}, \frac{\partial^2 y}{\partial x^2}, ..., \int y dx, ...) \tag{3.3}$$

The numerical methods for time integration of these equations will be discussed in the next section (Section 4). Before we conduct the time integration, we need to determine the differentiations $\frac{\partial y}{\partial x}, \frac{\partial^2 y}{\partial x^2}, ...$ and integrations $\int y dx, ...$ on the left hand side of the equation (3.3) at each grid point.

In this section, we are going to discuss the following four types of numerical methods, which are commonly used in spatial differentiations and integrations.

- Finite Differences (based on Taylor's expansion)
- FFT (Fast Fourier Transform)
- Cubic Spline
- Cubic Spline with Corrections

3.1. Finite Differences

For convenience, we shall use the following notation in the rest of this lecture notes.

$$f_{ijk}^n = f(x = i\Delta x, y = j\Delta y, z = k\Delta z, t = n\Delta t) = f(x_i, y_j, z_k, t^n)$$

Using finite difference method, we can obtain derivatives of a tabulated function f.

i	:	1	2	$\cdots$	N
x_i	:	x_1	x_2	$\cdots$	x_N
f_i	:	f_1	f_2	$\cdots$	f_N

Table 3.1 lists examples of the first order finite difference expressions of $[df/dx]_{x=x_i}$, $[d^2 f/dx^2]_{x=x_i}$, and $[d^3 f/dx^3]_{x=x_i}$. Table 3.2 lists examples of the finite difference

expressions of $y = \int f\,dx$.

Table 3.1. The first order numerical differentiations based on finite difference method

Derivatives	Central Difference	Forward Difference	Backward Difference
$\left.\dfrac{df}{dx}\right\|_{x=x_i}$	$\delta f_i = \dfrac{f_{i+1} - f_{i-1}}{2\Delta x}$	$\Delta f_i = \dfrac{f_{i+1} - f_i}{\Delta x}$	$\nabla f_i = \dfrac{f_i - f_{i-1}}{\Delta x}$
$\left.\dfrac{d^2 f}{dx^2}\right\|_{x=x_i}$	$\delta^2 f_i = \dfrac{f_{i+1} - 2f_i + f_{i-1}}{(\Delta x)^2}$	$\Delta^2 f_i = \dfrac{f_{i+2} - 2f_{i+1} + f_i}{(\Delta x)^2}$	$\nabla^2 f_i = \dfrac{f_i - 2f_{i-1} + f_{i-2}}{(\Delta x)^2}$
$\left.\dfrac{d^3 f}{dx^3}\right\|_{x=x_i}$	$\delta^3 f_i = \dfrac{f_{i+2} - 2f_{i+1} + 2f_{i-1} - f_{i-2}}{2(\Delta x)^3}$	$\Delta^3 f_i = \dfrac{f_{i+3} - 3f_{i+2} + 3f_{i+1} - f_i}{(\Delta x)^3}$	$\nabla^3 f_i = \dfrac{f_i - 3f_{i-1} + 3f_{i-2} - f_{i-3}}{(\Delta x)^3}$

Exercise 3.1. Determine the second order and the forth order central differences expressions of $[df/dx]_{x=x_i}$, and $[d^2 f/dx^2]_{x=x_i}$, based on the Taylor expansions of the function f.

Exercise 3.2. Use the first order, the second order, and the forth order central differences expressions to determine the df/dx, and $d^2 f/dx^2$, an analytical function f with a fixed Δx. Determine the numerical errors in your results. Compare the numerical errors obtained from different finite differences expressions.

Table 3.2. The spatial integrations based on finite difference method

	$\dfrac{dy(x)}{dx} = f(x),\quad h = \Delta x$
1st order integration	$y_{i+1} = y_i + h f_i + O(h^2 f)$
2nd order integration Trapezoidal rule	$y_{i+1} = y_i + h\dfrac{f_{i+1} + f_i}{2} + O(h^3 f'')$
4th order integration Simpson's rule	$y_{i+1} = y_i + h(\frac{1}{6} f_i + \frac{4}{6} f_{i+(1/2)} + \frac{1}{6} f_{i+1}) + O(h^5 f^{(4)})$
3rd order integration Simpson's 3/8 rule	$y_{i+1} = y_i + \frac{h}{3}(\frac{3}{8} f_i + \frac{9}{8} f_{i+\frac{1}{3}} + \frac{9}{8} f_{i+\frac{2}{3}} + \frac{3}{8} f_{i+1}) + O(h^4 f''')$

補助教材：

Show that, for $y'(x) = f(x)$,

$$y_{i+1} = y_i + h f_i + O(h^2 f)$$

$$y_{i+1} = y_i + h\frac{f_{i+1} + f_i}{2} + O(h^3 f'')$$

$$y_{i+1} = y_i + h(\frac{1}{6} f_i + \frac{4}{6} f_{i+(1/2)} + \frac{1}{6} f_{i+1}) + O(h^5 f^{(4)})$$

Proof:

Since

$$y(x+\Delta x) = y(x) + \frac{\Delta x}{1!} y'(x) + \frac{(\Delta x)^2}{2!} y''(x) + \frac{(\Delta x)^3}{3!} y'''(x) + \cdots$$

Substituting $y'(x) = f(x)$ into above equation yields

$$\boxed{y_{i+1} = y_i + h f_i + O(h^2 f)}$$

or

$$y_{i+1} = y_{i+\frac{1}{2}} + \frac{h}{2} f_{i+\frac{1}{2}} + \frac{(h/2)^2}{2!} f'_{i+\frac{1}{2}} + \frac{(h/2)^3}{3!} f''_{i+\frac{1}{2}} + \frac{(h/2)^4}{4!} f'''_{i+\frac{1}{2}} + \cdots \qquad (3.1)$$

$$y_i = y_{i+\frac{1}{2}} - \frac{h}{2} f_{i+\frac{1}{2}} + \frac{(-h/2)^2}{2!} f'_{i+\frac{1}{2}} + \frac{(-h/2)^3}{3!} f''_{i+\frac{1}{2}} + \frac{(-h/2)^4}{4!} f'''_{i+\frac{1}{2}} + \cdots \qquad (3.2)$$

Keeping the 2nd order term and subtracting Eq. (3.2) from Eq. (3.1) yields

$$y_{i+1} - y_i = 0 + h f_{i+\frac{1}{2}} + 0 + O(h^3 f'')$$

or

$$\boxed{y_{i+1} = y_i + h\frac{f_{i+1} + f_i}{2} + O(h^3 f'')}$$

Keeping the 4th order term and subtracting Eq. (3.2) from Eq. (3.1) yields

$$y_{i+1} - y_i = 0 + h f_{i+\frac{1}{2}} + 0 + \frac{(h/2)^3}{3} f''_{i+\frac{1}{2}} + 0 + O(h^5 f^{(4)})$$

or

$$y_{i+1} - y_i = h f_{i+\frac{1}{2}} + \frac{h^3}{24} \frac{f_{i+1} - 2 f_{i+(1/2)} + f_i}{(h/2)^2} + O(h^5 f^{(4)})$$

It yields

$$\boxed{y_{i+1} = y_i + h(\frac{1}{6} f_i + \frac{4}{6} f_{i+(1/2)} + \frac{1}{6} f_{i+1}) + O(h^5 f^{(4)})}$$

Exercise 3.3.

Derive Simpson's 3/8 rule.

3.2. FFT (Fast Fourier Transform)

A function can be expanded by a complete set of sine and cosine functions. In the Fast Fourier Transform, the sine and cosine tables are calculated in advance to save the CPU time of the simulation.

For a periodic function f, one can use FFT to determine its *differentiations* and *integrations,* i.e.,

$$\frac{df}{dx} = FFT^{-1}\{ik[FFT(f)]\}$$

$$\int f dx = FFT^{-1}\{\frac{1}{ik}[FFT(f)]\} \quad \text{for} \quad k > 0 .$$

Exercise 3.4.

Use an FFT subroutine to determine the first derivatives of a periodic analytical function f. Determine the numerical errors in your results.

Exercise 3.5.

Use an FFT subroutine to determine the first derivatives of a non-periodic analytical function f. Determine the numerical errors in your results.

3.3. Cubic Spline

A tabulate function can be fitted by a set of piece-wise continuous functions, in which the first and the second derivatives of the fitting functions are continuous at each grid point. One need to solve a tri-diagonal matrix to determine the piece-wise continuous cubic spline functions. The inversion of the tri-diagonal matrix depends only on the position of grid points. Thus, for simulations with fixed grid points, one can evaluate the inversion of the tri-diagonal matrix in advance to save the CPU time of the simulation.

For a non-periodic function f, it is good to use the cubic spline method to determine its *differentiations* and *integrations* at each grid point. Results of *differentiations* obtained from the cubic spline show the same order of accuracy as the results obtained from the forth order finite differences scheme.

補助教材：

The piece-wise continuous function in the cubic spline can be written in the following form.

$$f(x_k \le x \le x_{k+1}) = \frac{f(x_k)(x-x_{k+1})}{(x_k - x_{k+1})} + \frac{f(x_{k+1})(x-x_k)}{(x_{k+1}-x_k)} + [a_k \frac{(x-x_k)}{(x_{k+1}-x_k)} + b_k]\frac{(x-x_k)(x-x_{k+1})}{(x_{k+1}-x_k)^2}$$

The constants $\{a_k, b_k, for k = 1 \to n-1\}$ are chosen such that the matching conditions for cubic spline can be fulfilled, i.e.,

$$\left.\frac{df(x_{k-1} \le x \le x_k)}{dx}\right|_{x=x_k} = \left.\frac{df(x_k \le x \le x_{k+1})}{dx}\right|_{x=x_k}$$

and

$$\left.\frac{d^2 f(x_{k-1} \le x \le x_k)}{dx^2}\right|_{x=x_k} = \left.\frac{d^2 f(x_k \le x \le x_{k+1})}{dx^2}\right|_{x=x_k}$$

One can obtain the following two types of recursion formula

$$f'(x_{k-1}) + f'(x_k)[2+2(\frac{h_{k-1}}{h_k})] + f'(x_{k+1})(\frac{h_{k-1}}{h_k}) = 3f_0'(x_{k-1}) + 3f_0'(x_k)(\frac{h_{k-1}}{h_k})$$

$$f''(x_{k-1}) + f''(x_k)[2+2(\frac{h_k}{h_{k-1}})] + f''(x_{k+1})(\frac{h_k}{h_{k-1}}) = \frac{6}{h_{k-1}}[f_0'(x_k) - f_0'(x_{k-1})]$$

where $f_0'(x_k) = \frac{f(x_{k+1}) - f(x_k)}{x_{k+1} - x_k}$ and $h_k = x_{k+1} - x_k$.

Exercise 3.6.

Use a Cubic Spline subroutine to determine the first derivatives of an analytical function f. Determine the numerical errors in your results.

3.4. Cubic Spline with Corrections

Numerical oscillation may take place when we use higher order finite difference scheme or Cubic Spline to determine the *differentiations* of a sharp-changed function f. An exponential correction can reduce this type of numerical errors. Adding a damping term, $\partial^2 f / \partial x^2$, in the simulation code can also reduce this type of numerical errors in the simulation.

4. 處理時間積分的數值方法

Schemes for Solving Initial Value Problems—Numerical Methods for Time Integrations

All the numerical time integrations are constructed based on finite difference numerical schemes. FFT and Cubic Spline become useless in the numerical time integration processes. The numerical time integration schemes can be classified into the following two categories:

- Explicit Scheme:
  - The future information are determined based on the present and the post information
  - Easy to program, easy to blowout!
  - To avoid blowout → Choose shorter time step → Require more CPU time

- Implicit Scheme:
  - The future information are determined based on the future, the present, and the post information
  - Difficult to program and/or Require more memory
  - Stable in large time step → Save CPU time

4.1. Examples of Explicit Scheme

Examples of explicit time integration schemes include Euler method, Runge-Kutta method, Adams' open Formula, which is also called Adams-Bashforth Formula, and Lax-Wendroff scheme. Table 4.1. lists the numerical schemes of Euler method and Runge-Kutta method. Table 4.2 lists Adams' open formulae at different orders of accuracy, which will be discussed in the subsection 4.3. The Lax-Wendroff scheme to be discussed in this subsection is commonly used in fluid simulations.

Table 4.1. Explicit time integrations and their corresponding spatial integrations

The spatial integrations based on finite differences scheme	The explicit time integrations
$\frac{dy(x)}{dx} = f(x)\,, \quad h = \Delta x$	$\frac{dy(t)}{dt} = f(t, y)\,, \quad h = \Delta t$
1st order integration $y_{i+1} = y_i + h f_i + O(h^2 f)$	1st order explicit scheme: Euler method $y^{n+1} = y^n + h f(t^n, y^n) + O(h^2 f)$
2nd order integration Trapezoidal rule $y_{i+1} = y_i + h\frac{f_{i+1} + f_i}{2} + O(h^3 f'')$	2nd order Runge-Kutta method (an explicit scheme) $(y^*)^{n+1} = y^n + h f(t^n, y^n)$ $y^{n+\frac{1}{2}} = \frac{y^n + (y^*)^{n+1}}{2}$ $y^{n+1} = y^n + h f(t^{n+\frac{1}{2}}, y^{n+\frac{1}{2}}) + O(h^3 f'')$
4th order integration Simpson's rule $y_{i+1} = y_i$ $+ h(\frac{1}{6} f_i + \frac{4}{6} f_{i+(1/2)} + \frac{1}{6} f_{i+1})$ $+ O(h^5 f^{(4)})$	4th order Runge-Kutta method (an explicit scheme) $(y^*)^{n+1} = y^n + h f(t^n, y^n)$ $(y^*)^{n+\frac{1}{2}} = \frac{y^n + (y^*)^{n+1}}{2}$ $(y^{**})^{n+1} = y^n + h f(t^{n+\frac{1}{2}}, (y^*)^{n+\frac{1}{2}})$ $(y^{**})^{n+\frac{1}{2}} = \frac{y^n + (y^{**})^{n+1}}{2}$ $(y^{***})^{n+1} = y^n + h f(t^{n+\frac{1}{2}}, (y^{**})^{n+\frac{1}{2}})$ $y^{n+1} = y^n + h[\frac{1}{6} f(t^n, y^n)$ $+ \frac{2}{6} f(t^{n+\frac{1}{2}}, (y^*)^{n+\frac{1}{2}})$ $+ \frac{2}{6} f(t^{n+\frac{1}{2}}, (y^{**})^{n+\frac{1}{2}})$ $+ \frac{1}{6} f(t^{n+1}, (y^{***})^{n+1})] + O(h^5 f^{(4)})$
3rd order integration Simpson's $\frac{3}{8}$ rule $y_{i+1} = y_i$ $+ \frac{h}{3}(\frac{3}{8} f_i + \frac{9}{8} f_{i+\frac{1}{3}} + \frac{9}{8} f_{i+\frac{2}{3}} + \frac{3}{8} f_{i+1})$ $+ O(h^4 f''')$	

Exercise 4.1.

Solve proton's trajectory in a uniform magnetic field $\mathbf{B} = \mathbf{e}_z B_0$ and electric field $\mathbf{E} = \mathbf{e}_y E_0$ by means of (i) Euler method, (ii) 2nd order Runge-Kutta method, and (iii) 4th order Runge-Kutta method, where $\mathbf{e}_y$ and $\mathbf{e}_z$ are the unit vectors along the y and z directions, respectively. Solve this problem for 100 gyro periods with the following three different initial conditions. Plot proton's trajectory in both x-y space and in v_x-v_y space. Compare your numerical results with the analytical solutions.

Case 1: $\mathbf{x}(t=0)=0$ and $\mathbf{v}(t=0)=0$

Case 2: $\mathbf{x}(t=0)=0$ and $\mathbf{v}(t=0)=(2.5E_0/B_0)\mathbf{e}_x$

Case 3: $\mathbf{x}(t=0)=0$ and $\mathbf{v}(t=0)=(0.5E_0/B_0)\mathbf{e}_x$

The Lax-Wendroff scheme is an explicit scheme. It is good for solving fluid equations with absence of diffusion or dissipation terms. A set of one-dimensional fluid equations, without dissipation or diffusion terms, can be written in the following conservative form

$$\frac{\partial \mathbf{U}}{\partial t} + \frac{\partial \mathbf{F}(\mathbf{U})}{\partial x} = 0$$

which can be solved numerically by the second order Lax-Wendroff scheme.

Step 1:

$$\mathbf{U}_{i+\frac{1}{2}}^{n+\frac{1}{2}} = \frac{\mathbf{U}_{i+1}^n + \mathbf{U}_i^n}{2} - \frac{\Delta t}{2\Delta x}\left[\mathbf{F}(\mathbf{U}_{i+1}^n) - \mathbf{F}(\mathbf{U}_i^n)\right]$$

Step 2:

$$\mathbf{U}_i^{n+1} = \mathbf{U}_i^n - \frac{\Delta t}{\Delta x}[\mathbf{F}(\mathbf{U}_{i+\frac{1}{2}}^{n+\frac{1}{2}}) - \mathbf{F}(\mathbf{U}_{i-\frac{1}{2}}^{n+\frac{1}{2}})]$$

Additional examples and advanced discussion on using Lax-Wendroff scheme to solve fluid equations can be found in the book by Richtmyer and Morton (1967).

Exercise 4.2.

Using the second order Lax-Wendroff scheme to solve Korteweg-deVries (KdV) equation

$$\frac{\partial V}{\partial t} + (C_0 + V)\frac{\partial V}{\partial x} + \alpha\frac{\partial^3 V}{\partial x^3} = 0$$

with uniform boundary condition and a given initial profile $V(x, t=0)$ with a bump at center of the simulation domain. Plot evolutions of spatial profile $V(x,t)$. You can normalize your velocity field by C_0. Study the following two cases: one for $\alpha > 0$, and the other for $\alpha < 0$.

4.2. Examples of Implicit Scheme

Consider a charge particle moving in a uniform strong magnetic field. Momentum equation of this charge particle is

$$\frac{d\mathbf{v}(t)}{dt} = \frac{q}{m}\mathbf{v}(t) \times \mathbf{B}_0 \qquad (4.1)$$

The following numerical scheme is an implicit scheme of Eq. (4.1)

$$\mathbf{v}^{n+1} = \mathbf{v}^n + \Delta t \frac{q}{m}\frac{\mathbf{v}^n + \mathbf{v}^{n+1}}{2} \times \mathbf{B}_0 \qquad (4.2)$$

Exercise 4.3.

Solve Eq. (4.2) to obtain v_x^{n+1}, v_y^{n+1} and v_z^{n+1} for a given set of v_x^n, v_y^n, v_z^n, B_{0x}, B_{0y}, and B_{0z}.

Exercise 4.4.

Solve proton's trajectory in Exercise 4.1 by means of the implicit scheme discussed this section.

In addition to the gyro motion, the diffusion equation is another type of differential equation, which *should* be solved by an implicit scheme.

補助教材：

The diffusion equation

$$\frac{\partial T}{\partial t} = \kappa \frac{\partial^2 T}{\partial x^2} \qquad (4.3)$$

can be solved numerically by one of the following implicit schemes.

$$\frac{T_i^{n+1} - T_i^n}{\Delta t} = \frac{\kappa}{(\Delta x)^2}\frac{1}{2}[(T_{i+1}^n - 2T_i^n + T_{i-1}^n) + (T_{i+1}^{n+1} - 2T_i^{n+1} + T_{i-1}^{n+1})] \qquad (4.4)$$

or

$$\frac{T_i^{n+1} - T_i^n}{\Delta t} = \frac{\kappa}{(\Delta x)^2}(T_{i+1}^{n+1} - 2T_i^{n+1} + T_{i-1}^{n+1}) \qquad (4.5)$$

or

$$\frac{T_i^{n+1}-T_i^n}{\Delta t}=\frac{\kappa}{(\Delta x)^2}[(1-\lambda)(T_{i+1}^n-2T_i^n+T_{i-1}^n)+\lambda(T_{i+1}^{n+1}-2T_i^{n+1}+T_{i-1}^{n+1})] \tag{4.6}$$

where $0<\lambda<1$. For $\lambda=1/2$, Eq. (4.6) is reduced to Eq. (4.4). For $\lambda=1$, Eq. (4.6) is reduced to Eq. (4.5). Eq. (4.4) can be written as

$$-\alpha T_{i-1}^{n+1}+(1+2\alpha)T_i^{n+1}-\alpha T_{i+1}^{n+1}=\alpha T_{i-1}^n+(1-2\alpha)T_i^n+\alpha T_{i+1}^n \tag{4.7}$$

where $\alpha=\dfrac{\kappa\Delta t}{2(\Delta x)^2}$

For given boundary conditions $T(x=0)=T_0$, and $T(x=N_x\Delta x)=T_{N_x}$, Eq. (4.7) can be rewritten in the following tri-diagonal matrix form:

$$\begin{pmatrix} (1+2\alpha) & -\alpha & 0 & \cdots & 0 \\ -\alpha & (1+2\alpha) & -\alpha & & \vdots \\ 0 & & \ddots & & 0 \\ \vdots & & -\alpha & (1+2\alpha) & -\alpha \\ 0 & \cdots & 0 & -\alpha & (1+2\alpha) \end{pmatrix} \begin{pmatrix} T_1^{n+1} \\ T_2^{n+1} \\ \vdots \\ T_{N_x-2}^{n+1} \\ T_{N_x-1}^{n+1} \end{pmatrix} = \begin{pmatrix} 2\alpha T_0+(1-2\alpha)T_1^n+\alpha T_2^n \\ \alpha T_1^n+(1-2\alpha)T_2^n+\alpha T_3^n \\ \vdots \\ \alpha T_{N_x-3}^n+(1-2\alpha)T_{N_x-2}^n+\alpha T_{N_x-1}^n \\ \alpha T_{N_x-2}^n+(1-2\alpha)T_{N_x-1}^n+2\alpha T_{N_x} \end{pmatrix}$$

Exercise 4.5.

Write a subroutine, using Gauss elimination method to solve ($x_1,x_2,\cdots,x_n$) in the following tri-diagonal set of equations. Limit number of arrays used in your program. There should be no more than five $n\times 1$ arrays used in your program.

$$\begin{pmatrix} b_1 & c_1 & 0 & \cdots & 0 \\ a_2 & b_2 & c_2 & & \vdots \\ 0 & & \ddots & & 0 \\ \vdots & & a_{n-1} & b_{n-1} & c_{n-1} \\ 0 & \cdots & 0 & a_n & b_n \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_{n-1} \\ x_n \end{pmatrix} = \begin{pmatrix} r_1 \\ r_2 \\ \vdots \\ r_{n-1} \\ r_n \end{pmatrix}$$

Examples on how to solve tri-diagonal set of equations numerically can be found at Press et al. (1988).

Exercise 4.6.

Write a program to solve diffusion equation (4.3) for a given initial condition and boundary conditions. Plot evolution of spatial profile $T(x,t)$.

Adams' close formula, which is also called Adams-Moulton formula, is also an implicit scheme. Table 4.3 lists Adams' close formulae at different orders of accuracy, which will be discussed in the next subsection 4.3.

4.3. Predictor-Corrector Method Based on Adams Formula

Predictor-Corrector method is an easy-to-program implicit scheme, but require more memory than the corresponding explicit scheme. We use the Adams' formula to construct the Predictor-Corrector simulation scheme (e.g., Shampine and Gordon, 1975; Press et al., 1988). Tables 4.2 and 4.3 list the Adams' open and close formulae, respectively, at different orders of accuracy. Proof of these Adams' formulae can be found in advanced mathematics textbooks (e.g., Hildebrand, 1976). A 4th order Predictor-Corrector method (e.g., Shampine and Gordon, 1975) is summarized in Table 4.4.

Exercise 4.7.

Using the forth order Predictor-Corrector method described in Table 4.4 to solve Korteweg-deVries (KdV) equation in Exercise 4.2 and proton’s trajectory in Exercise 4.4.

Table 4.2. Adams' Open Formulae (also called Adams-Bashforth Formula)

Order of Accuracy	Solving $dy/dt = f$ or $\partial y/\partial t = f$ explicitly with $h = \Delta t$
1st	$y^{n+1} = y^n + h[f^n] + O(h^2 f')$
2nd	$y^{n+1} = y^n + h[\frac{3}{2} f^n - \frac{1}{2} f^{n-1}] + O(h^3 f'')$
3rd	$y^{n+1} = y^n + h[\frac{23}{12} f^n - \frac{16}{12} f^{n-1} + \frac{5}{12} f^{n-2}] + O(h^4 f''')$
4th	$y^{n+1} = y^n + h[\frac{55}{24} f^n - \frac{59}{24} f^{n-1} + \frac{37}{24} f^{n-2} - \frac{9}{24} f^{n-3}] + O(h^5 f^{(4)})$
5th	$y^{n+1} = y^n$ $+ h[\frac{1901}{720} f^n - \frac{2774}{720} f^{n-1} + \frac{2616}{720} f^{n-2} - \frac{1274}{720} f^{n-3} + \frac{251}{720} f^{n-4}] + O(h^6 f^{(5)})$
6th	$y^{n+1} = y^n$ $+ h[\frac{4277}{1440} f^n - \frac{7923}{1440} f^{n-1} + \frac{9982}{1440} f^{n-2} - \frac{7298}{1440} f^{n-3} + \frac{2877}{1440} f^{n-4} - \frac{475}{1440} f^{n-5}]$ $+ O(h^7 f^{(6)})$

Table 4.3. Adams' Close Formulae (also called Adams-Moulton Formula)

Order of Accuracy	Solving $dy/dt = f$ or $\partial y/\partial t = f$ implicitly with $h = \Delta t$
1st	$y^{n+1} = y^n + h[f^{n+1}] + O(h^2 f')$
2nd	$y^{n+1} = y^n + h[\frac{1}{2} f^{n+1} + \frac{1}{2} f^n] + O(h^3 f'')$
3rd	$y^{n+1} = y^n + h[\frac{5}{12} f^{n+1} + \frac{8}{12} f^n - \frac{1}{12} f^{n-1}] + O(h^4 f''')$
4th	$y^{n+1} = y^n + h[\frac{9}{24} f^{n+1} + \frac{19}{24} f^n - \frac{5}{24} f^{n-1} + \frac{1}{24} f^{n-2}] + O(h^5 f^{(4)})$
5th	$y^{n+1} = y^n + h[\frac{251}{720} f^{n+1} + \frac{646}{720} f^n - \frac{264}{720} f^{n-1} + \frac{106}{720} f^{n-2} - \frac{19}{720} f^{n-3}] + O(h^6 f^{(5)})$
6th	$y^{n+1} = y^n + h[\frac{475}{1440} f^{n+1} + \frac{1427}{1440} f^n - \frac{798}{1440} f^{n-1} + \frac{482}{1440} f^{n-2} - \frac{173}{1440} f^{n-3} + \frac{27}{1440} f^{n-4}] + O(h^7 f^{(6)})$

Table 4.4. Procedure of the 4th order Predictor-Corrector Method

Initial Steps	Using 4th order Runge-Kutta method to obtain y^1, y^2, and y^3 from y^0.
Predicting Step	Using 4th order Adams Open Formula to predict y^4 from y^0, y^1, y^2, and y^3.
Correcting Steps	Using 4th order Adams Close Formula to correct y^4 from y^1, y^2, y^3, and the predicted y^4 (or corrected y^4 of the last iteration).
	Repeat the correcting step for several times or until *the iteration converges*. [The condition of convergence in an iteration scheme will be discussed in the next section (Section 5).]
….	Repeat the *Predicting* and *Correcting Steps* to advance y from y^n to y^{n+1}.

5. 疊代法與機器誤差估算

Condition of Convergence in an Iteration Scheme and Estimation of Machine Errors

If U is the relative error of 1, then an iteration scheme is convergent when

$$\left|{}^{k+1}y^{n+1} - {}^{k}y^{n+1}\right| < U\left|{}^{k}y^{n+1}\right|.$$

where ${}^{k}y^{n+1}$ is the kth iteration result of y^{n+1}.

Machine-dependent relative error U can be obtained from the following program (e.g., Shampine and Gordon, 1975).

```
C   This subroutine determines machine-dependent relative error
C     relative to 1.
        Subroutine DGETU(U)
        Implicit double precision (a-h,o-z)
        A1=1.d0        !for double precision
        AH=0.5d0       !for double precision
C         A1=1.         !for single precision program
C         AH=0.5        !for single precision program
        U=A1
        UU=U
      1   CONTINUE
        UU=UU*AH
        UT=U+UU
        IF(UT.GT.U) GO TO 1
        U=UU*2
        RETURN
        END
```

6. 亂數產生器簡介（補充教材）

Random Number Generators

Random number generators are commonly used in particle-code simulations and in Monte Carlo simulations (e.g., Snell, 1975).

6.1. Uniform Random Number Generator

Uniform random number generator is a machine-dependent function. The following function is modified from the subroutine RANDU in IBM/SSP (Scientific Subroutine Package). In general, one should be able to find a machine-dependent intrinsic function of "uniform random number generator" from a given FORTRAN complier. However, for convenience, one can always use the following function to generate a uniform random number in a 32-bits machine.

```
C    This function obtains a machine-dependent uniform random number.
C    This function is good for 32 bits computer.
C    This function is modified from IBM/SSP subroutine RANDU
C    Constants used in this functions include
C    2147483648=2**31
C    0.4656613E-9=2.**(-31)
C    65539=65536+3=(2**16)+3
C
      function ran(ix)
      iy=ix*65539
      if(iy)5,6,6
    5   iy=iy+2147483647+1
    6   yfl=iy
      yfl=yfl*.4656613E-9
      ran=yfl
      ix=iy
      return
      end
```

Exercise 6.1.

Write a program to generate more than 1000 uniform random numbers. Plot distribution of these random numbers and compare it with the profile of uniform distribution function.

6.2. Random Number Generator of a General Non-uniform Probability Function

From the uniform random number generator, one can obtain random number of any given probability function. The following subroutine is an example of random number generator of any given probability function FUNC, in which random number between XL and XR are obtained. FMAX is the maximum of the probability function FUNC(x). A main program need provide an external probability function in order to use the following subroutine to generate a random number.

```
C    This subroutine obtains a random number of a given function FUNC(x).
C    The resulting random number is in the range of (XL, XR)
C    Fmax>0 is the maximum of the function FUNC(x) for XL<x<XR
C    ix is the seed of uniform random number.
C    program stop if fmax .le.0, or XR.LE.XL, or FUNC(x)<0 for XL<x<XR,
C
      subroutine ranfunc(x,FUNC,XL,XR,Fmax,ix)
      external FUNC
C--------------
      if(Fmax.le.0.) then
      print *,'program stop because Fmax<0, Fmax=', Fmax
      stop
      endif
      if(XR.le.XL) then
      print *,'program stop because XR.LE.XL, XR,XL=', XR,XL
      stop
      endif
C--------------
    1 continue
      x=ran(ix)*(XR-XL)+XL
      y=ran(ix)*Fmax
      y0= FUNC(x)
C--------------
      if(y0.lt.0.) then
      print *,'program stop because FUNC(x)<0, FUNC(x)=', y0
      stop
      endif
C--------------
      if(y.gt.y0) go to 1
      return
      end
```

Exercise 6.2.

Write a program to generate more than 1000 random numbers of a given function. Plot distribution of these random numbers and compare it with the profile of the given distribution function.

6.3. Random Number Generator of a Normal Distribution Function

According to *Law of Large Numbers*, and *Central Limit Theorem* (e.g., Snell, 1975), random number of normal distribution function can be obtained from uniform random number generator. According to subroutine GAUSS in IBM/SSP (Scientific Subroutine Package), random number of normal distribution function, with mean equal to zero and standard deviation equal to one, can be obtained from

$$Y = \frac{(\sum_{i=1}^{K} x_i) - \frac{K}{2}}{\sqrt{K/12}} \tag{6.1}$$

where all x_i are obtained from a uniform random number generator. Y is a random number of normal distribution with mean equal to zero and standard deviation equal to one.

Exercise 6.3.

Varify Eq. (6.1) based on *Law of Large Numbers*, and *Central Limit Theorem*.

Hint: According to Law of Large Numbers, if the mean of x_i is μ, and variance of x_i is σ^2 then the mean and variance of $S = x_1 + x_2 + \cdots + x_K$ will be $K\mu$ and $K\sigma^2$, respectively. For uniform random number generator, the corresponding probability function is $f(x) = x$ where $0 \le x \le 1$. It can be easily shown that the mean of this uniform probability function is 1/2 and the variance of this uniform probability function is 1/12. Thus, if $S = x_1 + x_2 + \cdots + x_K$ and x_i are obtained from uniform random number generator, the mean of S should be $K/2$ and variance of S should be $K/12$.

Exercise 6.4.

Based on Eq. (6.1), write a program to generate more than 1000 random numbers of a normal distribution function with $K = 12$ and $K = 6$. Plot distributions of these two sets of random numbers and compare them with the profile of normal distribution function. Compare CPU time used in generating random numbers in Exercise 6.2 and in this Exercise.

7. 數值模擬的診斷分析

Diagnostics of Simulation Results

We need to make a good diagnostics to understand detail nonlinear evolution processes and to determine the complicated cause-and-result relationships from the simulation results. Making good diagnostics is as important as choosing a good simulation scheme.

Guideline for making a *correct* simulation and good diagnostics

- Check your simulation results
  - Check and make sure the total energy is conserved.
  - Check and make sure that your simulation results satisfy the *Courant condition*. That is, the maximum speed v (which is equal to the largest possible wave speed plus the maximum flow speed or particle speed) multiplying one time step Δt is less than one grid size Δ_x. Indeed, we recommend that $v\Delta t < 0.1\Delta_x$.
  - Check and make sure that your simulation results are almost unchanged when the simulation system length is doubled, or when the simulation time step is reduced in half, or when the simulation grid size is reduced in half, or when the number of simulation particles is doubled, or when the real ion-electron mass ratio is used.
  - Always use double precision in your simulation.
- Display your simulation results
  - Carefully trace the time evolution of fields and fluid variables.
  - Carefully trace the phase-space trajectories of a *group* of simulation particles.
  - Use Matlab, IDL, or PV Wave to display massive simulation results automatically.
  - Use Excel or Kaleidagraph to display a single frame of a summary plot.
- Analysis and understand your simulation results
  - Explain your simulation results based on simple theoretical models.
    - Wave-wave interferences
    - Wave-particle interactions
    - Doppler shift effects
    - Instabilities
    - …

8. 總結與討論

Summary and Discussion

If you want to use numerical simulation to study nonlinear plasma phenomena, you should

(a) choose a right simulation code for your problem,

(b) do your best to save CPU time (simulation scheme) and real time (I/O),

(c) *always keep a macroscopic vision and a microscopic alert in your mind*,

(d) make a good diagnostics for your simulation results.

Prospective of Numerical Simulations

To build up a good simulation group, we need good hardwares, good softwares, and *scientists* with good experiences in doing different types of plasma simulations.

Beethoven can compose a symphony after he lost his hearing ability.

A simulation expert can predict simulation results even without a computer.

參考文獻

References

Hildebrand, F. B., *Advanced Calculus for Applications, 2nd edition,* Prentice-Hall, Inc., Englewood, Cliffs, New Jersey, 1976.

Press, W. H., B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling, *Numerical Recipes (in C or in FORTRAN and Pascal), Cambridg*e University Press, Cambridge, 1988.

Richtmyer, R. D., and K. W. Morton, *Difference Methods for Initial-Value Problems,* 2nd edition, John Wiley & Sons, Inc., 1967.

Shampine, L. F., and M. K. Gordon, *Computer Solution of Ordinary Differential Equation: the Initial Value Problem*, W. H. Freeman and Company, San Francisco, 1975.

Snell, J. L., *Introduction to Probability Theory With Computing,* Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1975.

System/360 Scientific Subroutine Package Version III, Programmer's Manual, 5th edition, IBM, New York, 1970.