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

講義課程大綱

講義課程v.s.演講內容

講義課程：
- ◆ 循序漸進，說明詳盡，可供自修研讀。
- ◆ 要好好用心做習題，才會有成效。
- ◆ 估計學習時間：約一個暑假或一學期。

講義課程v.s.演講內容

講義課程：
- ◆ 循序漸進，說明詳盡，可供自修研讀。
- ◆ 要好好用心做習題，才會有成效。
- ◆ 估計學習時間：約一個暑假或一學期。

演講內容：
- ◆ 重點整理數值模擬的基本概念。
- ◆ 強調從事數值模擬應該注意的事項。

什麼是數值模擬？

- ◆ 數值模擬就是寫個程式，利用電腦來做實驗，模擬非線性現象如何隨著時間演化。

什麼是數值模擬？

- ◆ 數值模擬就是寫個程式，利用電腦來做實驗，模擬非線性現象如何隨著時間演化。
  - 線性的問題，通常已有解析解，故不必勞師動眾用數值模擬來看它的演化情形。

什麼是數值模擬？

- 數值模擬就是寫個程式，利用電腦來做實驗，模擬**非線性現象**如何隨著**時間**演化。
  - 線性的問題，通常已有解析解，故不必勞師動眾用數值模擬來看它的演化情形。
  - 線性不穩定性的問題，常被用來檢驗數值模擬結果，是否值得信賴。

    不過，我們的經驗告訴我們：即使模擬結果與線性估算相符，仍不能保證，它在非線性部份的模擬結果一定正確！

什麼是數值模擬？

- 數值模擬就是利用一套**精準有效**的數值方法，解一組聯立的微分方程式（system ODEs or PDEs），其中，至少有一組方程式是**對時間微分**的方程式（範例）。
- 用微積分的基本概念來處理積分與微分，是一種以管窺天，以蠡測海，**奴役電腦的實驗**。

  誤差的累積：差之毫釐，失之千里。
  浮點運算的有效位數：機器誤差，使100%準確無法達成。
  基本方程式，略去了一些【微小項】，不是100%準確。
  蝴蝶效應：當【主要項】互相抵消達到平衡時，這些被略去的【微小項】，可能變得非常重要，舉足輕重，不可忽視。
  有限的電腦資源：RAM, HD, CPU, I/O速率
  有限的研究時間
- 一套精準有效的數值方法，不只要**穩**，還要**快**、要**準**！

要快：不要浪費時間

對時間微分的方程式

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

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

$$\frac{\partial y(x,t)}{\partial t} = f\left(t, y, \frac{\partial y}{\partial x}, \frac{\partial^2 y}{\partial x^2}, \ldots, \int y\,dx, \ldots\right) \qquad (3.3)$$

善用資源、能省則省、錙銖必較

Do your best to save

- Memory
  - 務必重複使用working Arrays
- CPU time
  - 避免重複計算：指數、對數、三角函數。
  - 重複計算部份，務必先建立Table。
- Real time
  - Watch out your I/O scheme

Bad and Good I/O Schemes

Very Very Slow:
Free Format ASCII data output
for Excel, IDL, PVWave to read

```
   DO I=1,N
   WRITE(1,*) X(I),VX(I)
   ENDDO
```

Fast:
Format ASCII data output
for Excel, IDL, PVWave to read

```
   WRITE(1,1) (X(I),VX(I),I=1,N)
1  FORMAT(1X, G15.8,1X,G15.8)
```

(For 8 significant digits: Choose 8+7=15)
e.g., Linux (Lahey f77): −0.12345678E-03
IBM Unix (xlf): −.12345678E-03

Good I/O Schemes for Massive Data Set

Fast and Save Disk Space:
Unformatted Binary data output
for Fortran, IDL, PVWave to read

```
   WRITE(1) N,(X(I),I=1,N)
   WRITE(2) N,(VX(I),I=1,N)
```

Fast and Save Disk Space:
Formatted Binary data output
for Fortran, IDL, PVWave to read

```
   WRITE(1,1) N,(X(I),I=1,N)
   WRITE(2,1) N,(VX(I),I=1,N)
1  FORMAT(I4,<N>A4)
```

為什麼要做數值模擬？

- 檢驗理論模式是否正確合理。
  - 理論模式通常可以簡化為一組聯立的微分方程式（system ODEs or PDEs）。
  - "To understand is to know how to calculate"
    - Direc
- 彌補實驗或觀測上之不足。
  - 實驗或觀測受到時空條件的限制，往往無法取得足夠精細的時空資訊。
- 藉由高時空解析度的模擬結果，了解非線性過程中的因果關係與主要物理機制。
  - "… the more I discover, the more I want to know"
    - Maeve Leakey
- 預測（預報）在不同初始條件與邊界條件下非線性過程的發展情形。

如何選擇電漿數值模擬碼

- 選擇模擬碼，要對症下藥：Choose a right simulation code for your problem
- 要隨時保有：宏觀的視野與微觀的警惕。知道自己的模擬碼，適用的底線在哪裡！
- 要先學過電漿物理，才會做正確的選擇！

電漿數值模擬碼的基本類型：

- Fluid Simulations
  - MHD code
  - Two-Fluid code
- Kinetic Simulations
  - Hybrid code (Fluid electrons and kinetic ions)
  - Full particle code
  - Test particle code
  - Vlasov code

The particle-code simulation is indeed a multiple-fluid simulation.

Test particle code

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 $\mathbf{B}(\mathbf{x})$

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

Full particle code

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:

$$\nabla \cdot \mathbf{E}(\mathbf{x},t) = \int \sum_\alpha \frac{e_\alpha}{\varepsilon_0} \delta(\mathbf{x}' - \mathbf{x}_\alpha) S(\mathbf{x} - \mathbf{x}') d\mathbf{x}'$$

$$\nabla \cdot \mathbf{B} = 0$$

$$\frac{\partial \mathbf{B}(\mathbf{x},t)}{\partial t} = -\nabla \times \mathbf{E}(\mathbf{x},t)$$

$$\nabla \times \mathbf{B}(\mathbf{x},t) = \mu_0 \int \sum_\alpha e_\alpha \frac{\mathbf{u}_\alpha(t)}{\sqrt{1+[u_\alpha(t)/c]^2}} \delta[\mathbf{x}' - \mathbf{x}_\alpha(t)] S(\mathbf{x} - \mathbf{x}') d\mathbf{x}' + \frac{1}{c^2}\frac{\partial \mathbf{E}(\mathbf{x},t)}{\partial t}$$

Vlasov code

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 $\nabla \cdot \mathbf{B} = 0$ $\nabla \cdot \mathbf{E} = \frac{e}{\varepsilon_0}[\iiint f_i d^3v - \iiint f_e d^3u]$

數值模擬要用到的數值方法

以下介紹
- 解System ODEs or PDEs要用到的基本數值方法
- 與電漿物理無關
- 但是Kinetic Plasma Simulation所遇到的函數相當不友善，因此模擬時可能遭遇的困境也最大，須要非常謹慎小心！

處理微分與積分常用的數值方法

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

Finite Differences

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

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

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

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

模擬碼中處理空間的微分與積分，除了用Finite Differences還可用

- FFT（適用於週期函數，非週期函數用FFT求微分與積分時，可能產生巨大誤差。）

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

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

- Cubic Spline
  - Piece-wise continuous cubic function
    f(x) = Ax³ + Bx² + Cx + D
  - 在連接點上，函數值、函數一次微分值、與函數二次微分值，均連續。

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 $h_k = x_{k+1} - x_k$.

處理時間積分的數值方法

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

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

Examples of Explicit Scheme

- Euler method
- Runge-Kutta method
- Adams' open Formula (Adams-Bashforth Formula)
- Lax-Wendroff scheme
  （只含奇次空間微分之時空微分方程式）

Euler method/ Runge-Kutta method

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 $y_{i+1} = y_i + \frac{h}{3}(\frac{3}{8} f_i + \frac{9}{8} f_{i+\frac{1}{3}} + \frac{9}{8} f_{i+\frac{2}{3}} + \frac{3}{8} f_{i+1}) + O(h^4 f''')$	

處理只含奇次空間微分之時空微分方程式可用Explicit Scheme：

Lax-Wendroff scheme

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

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

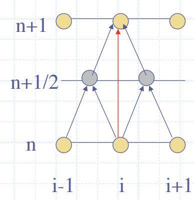
Step 1:

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

Step 2:

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

Richtmyer and Morton (1967)



Examples of Implicit Scheme

- Equatons of motion with gyro-type motion
- The diffusion equation
  （含偶次空間微分之時空微分方程式）
- Adams' close formula (Adams-Moulton formula)

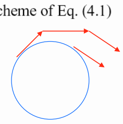
模擬打轉運動Gyro Motion 一定要用Implicit Scheme

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}.

模擬含偶次空間微分之時空微分方程式一定要用Implicit Scheme：

The Diffusion Equation

The diffusion equation

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

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

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

or

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

or

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

The Diffusion Equation

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

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

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:

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

Based on Runge-Kutta Method and Adams Formula

Predictor-Corrector Method

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}.

預測用Explicit Scheme：

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	$y^{n+1}=y^n+h[f^n]+O(h^2f')$
2nd	$y^{n+1}=y^n+h[\frac{3}{2}f^n-\frac{1}{2}f^{n-1}]+O(h^3f'')$ Cannot self-start
3rd	$y^{n+1}=y^n+h[\frac{23}{12}f^n-\frac{16}{12}f^{n-1}+\frac{5}{12}f^{n-2}]+O(h^4f''')$ Cannot self-start
4th	$y^{n+1}=y^n+h[\frac{55}{24}f^n-\frac{59}{24}f^{n-1}+\frac{37}{24}f^{n-2}-\frac{9}{24}f^{n-3}]+O(h^5f^{(4)})$ Cannot self-start
5th	$y^{n+1}=y^n+h[\frac{1901}{720}f^n-\frac{2774}{720}f^{n-1}+\frac{2616}{720}f^{n-2}-\frac{1274}{720}f^{n-3}+\frac{251}{720}f^{n-4}]+O(h^6f^{(5)})$ Cannot self-start
6th	$y^{n+1}=y^n+h[\frac{4277}{1440}f^n-\frac{7923}{1440}f^{n-1}+\frac{9982}{1440}f^{n-2}-\frac{7298}{1440}f^{n-3}+\frac{2877}{1440}f^{n-4}-\frac{475}{1440}f^{n-5}]+O(h^7f^{(6)})$ Cannot self-start

校正用Implicit Scheme：

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	$y^{n+1}=y^n+h[f^{n+1}]+O(h^2f')$
2nd	$y^{n+1}=y^n+h[\frac{1}{2}f^{n+1}+\frac{1}{2}f^n]+O(h^3f'')$
3rd	$y^{n+1}=y^n+h[\frac{5}{12}f^{n+1}+\frac{8}{12}f^n-\frac{1}{12}f^{n-1}]+O(h^4f''')$
4th	$y^{n+1}=y^n+h[\frac{9}{24}f^{n+1}+\frac{19}{24}f^n-\frac{5}{24}f^{n-1}+\frac{1}{24}f^{n-2}]+O(h^5f^{(4)})$
5th	$y^{n+1}=y^n+h[\frac{251}{720}f^{n+1}+\frac{646}{720}f^n-\frac{264}{720}f^{n-1}+\frac{106}{720}f^{n-2}-\frac{19}{720}f^{n-3}]+O(h^6f^{(5)})$
6th	$y^{n+1}=y^n+h[\frac{475}{1440}f^{n+1}+\frac{1427}{1440}f^n-\frac{798}{1440}f^{n-1}+\frac{482}{1440}f^{n-2}-\frac{173}{1440}f^{n-3}+\frac{27}{1440}f^{n-4}]+O(h^7f^{(6)})$

疊代法與機器誤差估算

◆Iteration scheme

If U is the relative error of 1,
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}.

疊代法與機器誤差估算

◆ Machine-dependent relative error U

◆ 當浮點運算時，
若電腦認為1+U還等於1時，
U就是相對於1的機器相對誤差（下一頁）

◆ 利用浮點運算的誤差，可用來設計
均勻分布之亂數產生器（補充教材）

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
```

數值模擬應注意與檢驗的事項

◆ Always use double precision in your simulation
◆ Normalize your equations可以幫助你選取模擬相關係數，如系統長度、時間間距、空間間距。
◆ 檢查系統總能量是否守恆
◆ 檢查是否滿足Courant condition: $dt * V_{max} < dx$

其他進一步的檢查： Make Sure

◆ 當系統長度增加一倍，模擬結果沒有重大改變。
◆ 當時間間距減半，模擬結果沒有重大改變。
◆ 當空間間距減半，模擬結果沒有重大改變。
◆ 當粒子數增加一倍，模擬結果沒有重大改變。
◆ 選用真實離子電子質量比，模擬結果沒有重大改變。

數值模擬的診斷分析

Display your simulation results

◆ 利用 Matlab或 IDL 或 PVWave等軟體，寫一個簡單繪圖檔（指令檔）自動處理大量數值模擬結果。
◆ 利用 Excel或Kaleidagraph等軟體，個別處理總結形式的圖片.（詳見朱旭新：數據製圖）
◆ Carefully trace the time evolution of all fluid variables.
◆ Carefully trace the phase-space trajectory of a *group* of simulation particles.

數值模擬的診斷分析

◆ 要用心分析你的模擬結果，不要浪費了辛苦算出來的模擬結果！

◆ 最好不要一邊模擬一邊做診斷繪圖。如果模擬一個Case所花費的時間，超過一天，最好把需要分析的資料存下來，用另外一台電腦慢慢的仔細分析。

◆ 範例一
◆ 範例二

總結與討論

◆ 能省則省，錙銖必較：Do your best to save memory, CPU time, and real time (Watch out your I/O scheme).
◆ 選擇模擬碼，要對症下藥：Choose a right simulation code for your problem
◆ 要隨時保有：宏觀的視野與微觀的警惕。知道自己的模擬碼，適用的底線在哪裡！
◆ 好的診斷（ good diagnostics ）可以幫助我們了解模擬現象背後的物理過程。

總結與討論

◆ 好的模擬研究，需要
- 好的硬體
- 好的軟體，
- 優秀的、有經驗的模擬人才（manpower）。

◆ 一個真正有經驗的模擬專家，不必用電腦，也可以大致預估模擬的結果。就好像貝多芬失聰後，仍然能作曲。

（電漿模擬專家Dawson 曾為電漿物理教科書中的非線性課題提供詳盡的定性說明。）

下課了！

感謝各位的捧場！

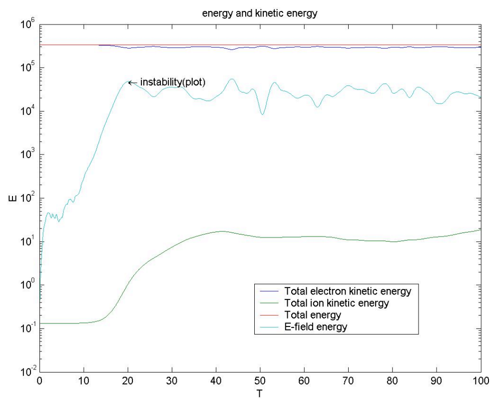
模擬結果範例一

作者：中央大學大氣科學系，大四學生王詠晶，太空電漿數值模擬期末作業

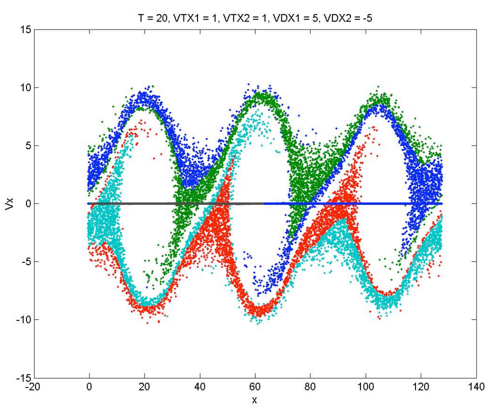
題目：全粒子數值模擬碼模擬雙流體不穩定

◆ 圖一：系統總能、粒子動能、電場能量隨時間變化圖

◆ 圖二：進入非線性飽和階段時，相空間粒子分布圖（將電子分為四群，離子兩群）



Time*Omega_pe



模擬結果範例 二

作者：中央大學太空所研究生蔡宗哲

◆ Vlasov Simulation模擬靜電激震波

non-periodic boundary condition

比較模擬結果phase space plots

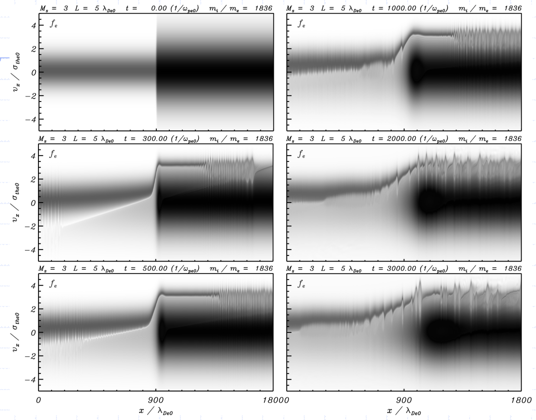
m_i/m_e=1836 : at $t\omega_{pe}$=0~3000

$t\omega_{pi}$=0~70

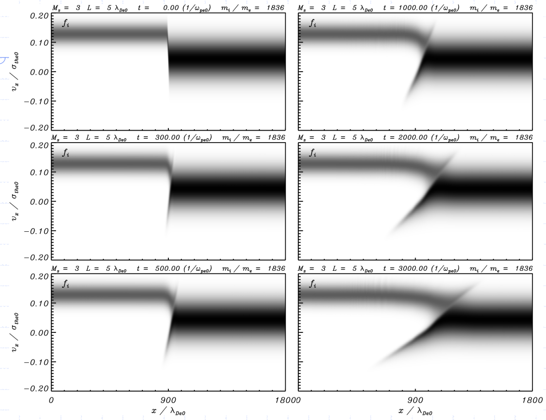
m_i/m_e =25 : at $t\omega_{pe}$=300

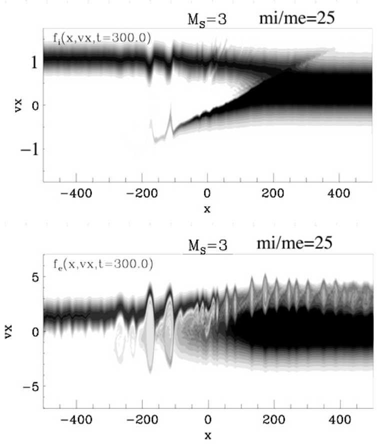
$t\omega_{pi}$=60

Electrons' phase space plots



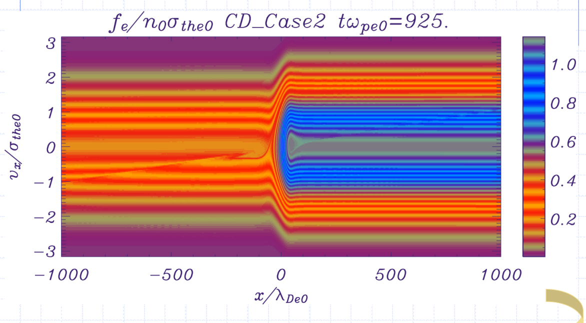
Ions' phase space plots





接觸部連續面數值模擬結果（作者：呂 ）

Color phase space plot can reveal characteristic curves in phase space



亂數產生器（補充教材）

常用於

- particle-code simulations
- Monte Carlo simulations

Random Number Generators for:

- Uniform Probability Function
- General Non-uniform Probability Function
- Normal Distribution Function

Uniform Probability Function

```
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
```

General Non-uniform Probability Function

```
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
```


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.