2007 Laser-Plasma Summer School
中研院原分所浦大邦講堂
July 16, 2007

數值方法

呂凌霄
中央大學　太空科學研究所

大綱

數值模擬運算與分析要用到的數值方法

- 差分法與數值誤差的估算
- 空間的微分與積分
- 時間的積分
- *認識電腦運算的極限*
- 疊代法與機器誤差估算
- 亂數產生器
- 求根
- 內差與分散
- 總結

差分法與數值誤差估算

- 差分法：
  - 中央差分法、前差分法、後差分法
  - 中央差分法：一般空間的微分與積分
  - 前差分法：快速流問題
  - 後差分法：對時間的積分
- 差分法的數值誤差估算：
  - 直覺的看法：微積分的定義
  - 進一步的探究：Taylor series expansion

微積分的定義

$$\frac{df(x)}{dx} = \lim_{\Delta x \to 0} \frac{f(x+\Delta x) - f(x)}{\Delta x}$$ 前差分法

$$= \lim_{\Delta x \to 0} \frac{f[x+(\Delta x/2)] - f[x-(\Delta x/2)]}{\Delta x}$$ 中央差分法

$$= \lim_{\Delta x \to 0} \frac{f(x) - f(x-\Delta x)}{\Delta x}$$ 後差分法

利用差分法求空間微分

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

常用差分符號的定義： $f^n_{ijk} = f(x = i\Delta x, y = j\Delta y, z = k\Delta z, t = n\Delta t)$
$= f(x_i, y_j, z_k, t^n)$

Q:算算看各項係數和是多少？

求積分

Given $y'(x)$ and $y_i = y(x_i)$, find $y_{i+1} = y(x_{i+1})$

第一類型，比較簡單：求曲線下的面積，曲線函數為f(x)

If $y'(x) = f(x)$ then $y(x_{i+1}) = y(x_i) + \int_{x_i}^{x_{i+1}} f(x)\,dx$

第二類型，比較難：trace a curve of a given field f(x,y)

If $y'(x) = f[x, y(x)]$ then $y(x_{i+1}) = y(x_i) + ?$

•以下將用第一類型的積分問題，說明如何估算數值誤差以及計算效率。
•第二類型的微分方程，為數值模擬中，最常遇到的問題形態。

差分法之準確度與誤差估算
Taylor series expansion

Given $y'(x) = f(x)$ and y_i, find $y_{i+1} = y_i + \int_{x_i}^{x_{i+1}} f(x)\,dx$

將 y_{i+1} 對 x_i 這點展開，可得下式，其中 $h = \Delta x = x_{i+1} - x_i$

$$y_{i+1} = y_i + hf_i + O(h^2 f')$$ （長條圖積分：一階準確）

先將 y_i 對 $x_{i+(1/2)}$ 這點展開，再將 y_{i+1} 對 $x_{i+(1/2)}$ 這點展開，兩式結果相減， 保留三項，其餘為餘數，可得

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

$$y_{i+1} = y_i + h\frac{f_{i+1} + f_i}{2} + O(h^3 f'')$$ （梯形法積分：二階準確）

差分法之準確度與誤差估算
Taylor series expansion

先將 y_i 對 $x_{i+(1/2)}$ 這點展開，再將 y_{i+1} 對 $x_{i+(1/2)}$ 這點展開，兩式結果相減，保留五項，其餘為餘數，可得

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

將 $f''_{i+(1/2)} = (f_{i+1} - 2f_{i+(1/2)} + f_i)/(h/2)^2$ 代入上式，得

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

（The 4th order Simpson's rule 積分法：四階準確）

計算效率評估

- 考慮：長條圖積分（一階）與 The 4th order Simpson's rule 積分法
- 每一個積分步驟： 四階積分法 比 一階積分法 多3倍
- 欲達到準確度 s： 一階積分法 比 四階積分法 多走$s^{(-3/4)}$ 倍
- 故一階積分法 比 四階積分法 慢 $s^{(-3/4)}/3$ 倍
- For s=0.01, $s^{(-3/4)}/3$ is about 10
  - 如果容許百分之一以下的誤差， 四階積分法 比 一階積分法 快 10 倍
- For s=0.0001, $s^{(-3/4)}/3$ is about 300
  - 如果只容許萬分之一以下的誤差， 四階積分法 比 一階積分法 快 300 倍
- 計算效率評估： 高階差分法遠高於低階差分法
  - 但是如果只算一次，用高階差分法運算，寫程式所用的時間，可能比運算所省下來的時間還多！

求積分（曲線下面積）

Table 3.2. The spatial integrations based on finite difference method

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

Q:算算看各項係數和是多少？

非差分法之
空間微分與積分計算法

- Life is more than the finite differences
- FFT （Fast Fourier Transform，fast:預先建立好轉換table）
  - $df(x)/dx = F^{-1}\{ ik\, F[f(x)] \}$ $\quad$ $\text{Int}[f(x)] = F^{-1}\{F[f(x)] / ik\}$
  - 適用於求週期性函數或數值模擬的微分與積分
  - 波動現象的微分與積分，FFT比起差分法，準確度高又有效率
  - 非週期函數用 FFT 求微分與積分時，可能產生巨大誤差。
- Fast Cubic Spline （fast:預先建立好反矩陣table）
  - 分段連續之三次多項式 $f(x) = Ax^3 + Bx^2 + Cx + D$
  - 在連接點上，相鄰兩段區間之函數值、函數一次微分值、與函數二次微分值，均連續。
  - 適用於求週期性或非週期性函數的微分與積分
  - 準確度與三階差分法相似（高階差分法的微分見補充教材）

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

係數為 tri-diagonal matrix!

補充教材：
微分之高階中央差分表示式

- 假設微分結果可寫成鄰近網格點處 y 值的線性組合。

$$y_0^{(s)} = \frac{1}{h^s}\left(\sum_{k=-N}^{N} a_k y_k\right) + R$$

- 將 $\{ y_{-N} \sim y_N \}$ 相對 x_0 這點做Taylor expansion然後代入上式得

$$y_0^{(s)} = \frac{1}{h^s}\left[\sum_{k=-N}^{N} a_k \left(y_0 + khy_0^{(1)} + \frac{(kh)^2}{2} y_0^{(2)} + \frac{(kh)^3}{6} y_0^{(3)} + \ldots\right)\right] + R$$

- 比較 $y_0^{(n)}$ 項的係數，忽略 $n > 2N$ 之高階餘數項，可得 $2N+1$ 個 $\{ a_{-N} \sim a_N \}$ 之一階聯立方程式。求解 $\{ a_{-N} \sim a_N \}$ ，即得 $y_0^{(s)}$ 之 $2N$ 階中央差分表示式。

第二類型積分問題

- Trace a curve of a given field $f(x,y)$

$$\frac{dy(x)}{dx} = f[x, y(x)] \quad \text{find} \quad y_{i+1} = y_i + ?$$

- $y(t)$ 對時間的積分

$$\frac{dy(t)}{dt} = f(t,y) \quad \text{find} \quad y^{n+1} = y^n + ?$$

- $y(x,t)$ 對時間的積分

$$\frac{\partial y(x,t)}{\partial t} = f\left(t, y, \frac{\partial y}{\partial x}, \frac{\partial^2 y}{\partial x^2}, \ldots, \int y\,dx, \ldots\right) \quad \text{find} \quad y_i^{n+1} = y_i^n + ?$$

- 其中$y(x,t)$對空間的偏微分，可由前述差分法、FFT、Cubic Spline、或其他高階中央差分求得
- 以下將以 $dy(t)/dt = f(t,y)$ 為範例，進行討論

時間積分

- Solve $y(t)$ for $dy(t)/dt = f(t,y)$ with a given $y^0 = y(t=0)$
- Explicit scheme
  - 利用過去與現在的資訊，估算時間積分之值，以預測未來的結果。
  - 運算比較快、 程式比較好寫，但是誤差會不斷地累積。
- Implicit scheme
  - 利用過去、現在、與未來的資訊，估算時間積分之值。其中未來的資訊，可藉由聯立方程式求解，或利用疊代法求解。
  - 運算費時、程式難寫，但是誤差不會累積。

The explicit time integrations

The 1st order Euler method

$$y^{n+1} = y^n + hf(t^n, (y^*)^n) + O(h^2 f^{(1)})$$

The 2nd order Runge-Kutta method

$$(y^*)^{n+1} = y^n + hf(t^n, y^n)$$

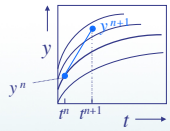
$$(y^*)^{n+(1/2)} = \frac{y^n + (y^*)^{n+1}}{2}$$

$$y^{n+1} = y^n + hf(t^{n+(1/2)}, (y^*)^{n+(1/2)}) + O(h^3 f^{(2)})$$

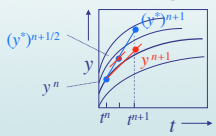
$$\frac{dy(t)}{dt} = f[t, y(t)]$$

圖解法

- **The 1st order Euler method**



- **The 2nd order Runge-Kutta method**



The 4th order Runge-Kutta method

$$(y^*)^{n+1} = y^n + hf(t^n, y^n)$$

$$(y^*)^{n+(1/2)} = \frac{y^n + (y^*)^{n+1}}{2}$$

$$(y^{**})^{n+1} = y^n + hf(t^{n+(1/2)}, (y^*)^{n+(1/2)})$$

$$(y^{**})^{n+(1/2)} = \frac{y^n + (y^{**})^{n+1}}{2}$$

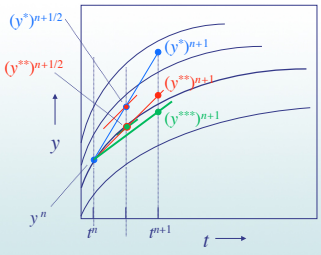
$$(y^{***})^{n+1} = y^n + hf(t^{n+(1/2)}, (y^{**})^{n+(1/2)})$$

$$y^{n+1} = y^n + h\left[\frac{1}{6} f(t^n, y^n) + \frac{2}{6} f(t^{n+(1/2)}, (y^*)^{n+(1/2)}) + \frac{2}{6} f(t^{n+(1/2)}, (y^{**})^{n+(1/2)}) + \frac{1}{6} f(t^{n+1}, (y^{***})^{n+1})\right] + O(h^5 f^{(4)})$$

$$\frac{dy(t)}{dt} = f[t, y(t)]$$

圖解法

- **The 4nd order Runge-Kutta method**



The implicit time integrations

- 如果可能，直接解線性聯立方程式。
  - 當問題為三元一次聯立方程式
  - 當問題為n元一次聯立方程式，且係數矩陣為tri-diagonal matrix
  - Examples:
- 當聯立方程式為非線性方程式，或雖為線性方程式，但是係數矩陣太大、太複雜
  - 用疊代法求解
  - 例如，predictor-corrector method

當問題為三元一次聯立方程式

帶電粒子在磁場中運動：

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

The implicit scheme:

$$\frac{\mathbf{v}(t+\Delta t) - \mathbf{v}(t)}{\Delta t} = \frac{q}{m}\frac{\mathbf{v}(t+\Delta t) + \mathbf{v}(t)}{2} \times \mathbf{B}$$

解以上三元一次聯立方程式，可積分求得

$$\{v_x(t+\Delta t), v_y(t+\Delta t), v_z(t+\Delta t)\}$$

當問題為n元一次聯立方程式，且係數矩陣為tri-diagonal matrix

- The diffusion equation $\frac{\partial T}{\partial t} = \kappa \frac{\partial^2 T}{\partial x^2}$
- The implicit scheme

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

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

where $\alpha = \frac{\kappa \Delta t}{2(\Delta x)^2}$ $T(x=0) = T_0$ $T(x = N_x \Delta x) = T_{N_x}$

結論：
要先學會解 tri-diagonal matrix的問題

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

怎麼解 $\{x_1 \sim x_n\}$ ？
這是你們的功課，今天沒時間介紹了！

疊代法形式之 implicit time integration scheme：
Predictor-Corrector Method

基本概念：

$$\frac{dy(t)}{dt} = f[t, y(t)]$$

- 先用explicit scheme 根據過去與現在的 y 值，預測下一步可能之 y 值
- 根據預測所得之 y 值，以及過去與現在的 y 值，重新預測下一步可能之 y 值
- 重複以上第二個步驟（疊代iteration），直到最新的 y 值與前一次疊代求出的 y 值相同（小於機器誤差），就可停止疊代，回到第一個步驟，進行下一個 time step 的積分。
- 範例：The 4th order Predictor-Corrector Method
  （Shampine and Gordon, 1975）

The 4th Order Predictor-Corrector Method
（Shampine and Gordon, 1975）

Solving $dy/dt = f$ or $\partial y/\partial t = f$ with $f = f(y,t)$ and $h = \Delta t$

- 預測步驟：利用 y^n, y^{n-1}, y^{n-2}, y^{n-3}, 預測y^{n+1}之值 (the 4th order Adams' open formula)

$$y^{n+1} = y^n + h[\frac{55}{24}f^n - \frac{59}{24}f^{n-1} + \frac{37}{24}f^{n-2} - \frac{9}{24}f^{n-3}] + O(h^5 f^{(4)})$$

- 校正（疊代）步驟：利用前一次預測所得的 y^{n+1} ，以及原有之 y^n, y^{n-1}, y^{n-2}, 重新預測 y^{n+1} 之值 (the 4th order Adams' close formula)

$$y^{n+1} = y^n + h[\frac{9}{24}f^{n+1} + \frac{19}{24}f^n - \frac{5}{24}f^{n-1} + \frac{1}{24}f^{n-2}] + O(h^5 f^{(4)})$$

- 重複校正（疊代）數次，直到收斂，即可進行下一步的預測
- 因為當 $n < 4$ 時， the 4th order Adams' open formula 不適用，故利用 the 4th order Runge-Kutta method 來求 y^1, y^2, y^3之值

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	$y^{n+1} = y^n + h[f^n] + O(h^2 f')$
2nd	$y^{n+1} = y^n + h[\frac{3}{2}f^n - \frac{1}{2}f^{n-1}] + O(h^3 f'')$ Cannot self-start
3rd	$y^{n+1} = y^n + h[\frac{23}{12}f^n - \frac{16}{12}f^{n-1} + \frac{5}{12}f^{n-2}] + O(h^4 f''')$ Cannot self-start
4th	$y^{n+1} = y^n + h[\frac{55}{24}f^n - \frac{59}{24}f^{n-1} + \frac{37}{24}f^{n-2} - \frac{9}{24}f^{n-3}] + O(h^5 f^{(4)})$ Cannot self-start
5th	$y^{n+1} = y^n + h[\frac{1901}{720}f^n - \frac{2774}{720}f^{n-1} + \frac{2616}{720}f^{n-2} - \frac{1274}{720}f^{n-3} + \frac{251}{720}f^{n-4}] + O(h^6 f^{(5)})$ 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	$y^{n+1} = y^n + h[f^{n+1}] + O(h^2 f')$
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	$y^{n+1} = y^n + h[\frac{251}{720}f^{n+1} + \frac{646}{720}f^n - \frac{264}{720}f^{n-1} + \frac{106}{720}f^{n-2} - \frac{19}{720}f^{n-3}] + O(h^6 f^{(5)})$

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}.

電腦運算的極限

- 疊代法在implicit time integration 中，扮演著重要的角色！
- 可是判斷疊代過程是否收斂，也就是判斷 the kth iteration 結果是否等於 the (k+1)th iteration 結果卻是一門大學問！
- 因此在說明如何判斷疊代收斂之前我們須要先介紹一下電腦一般整數與浮點運算的極限。
- 事實上，在撰寫模擬程式時，如果能隨時隨地考慮電腦運算上的極限，可以大幅減少模擬結果的數值誤差。

電腦運算的極限

- 電腦系統中整數與浮點運算的極限。
- 整數運算：
  - 最大值與最小值的極限（32-bit or 64-bit system?）
  - 超出一般整數運算最大值與最小值的計算，可透過Integer Array記錄更高位數的整數值，來進行毫無誤差的計算。（試寫一個程式求階層123456789!之完整數值）
- 浮點運算：
  - 有效位數的極限、二進位法與十進位法的轉換
  - 最大值與最小值的極限

疊代法與機器誤差估算

- Iteration
  - Interations are commonly used in solving Laplace equation, Poison equation, or use implicit scheme to solve differential equation(s).
  - 疊代法通常需要藉由相對誤差或絕對誤差來判斷是否可以停止疊代，找到解了。
- Machine error (relative error)
  - Let U be the relative error of 1, then
    - The computer cannot distinguish the differences between U+1 and 1
    - The relative error of a real number A will be $U*A$
  - Example: (Shampine and Gordon, 1975)

估算1的相對誤差

```
C--SUB. GETRERR---------------------------------
C To obtain the relative error with respect to 1
      SUBROUTINE GETRERR
      IMPLICIT DOUBLE PRECISION (A-H, O-Z)
      COMMON /GET_RERR/ RELERR,ABSERR
      A1=1
      AH=0.5D00
      U=A1
    1 CONTINUE
      U=U*AH
      TEMP=A1-U
      IF(TEMP.LT.A1) GO TO 1
      RELERR=U*2
      ABSERR=U*200
      RETURN
      END
```


亂數產生器

- 電腦系統的機器誤差，對使用者唯一的正面貢獻，就是可以製造隨機亂數！
- 0~1 uniformly distributed random number generator
  - 通常由電腦系統直接提供
  - 是建構其他亂數產生器的basic element
- Normal-distributed random number generator
  - 大數法則 與 中央極限定理 的最佳應用
  - 效率高
- General random number generator of a given probability function f(x)
  - 標準作業流程，適用於任何機率函數
  - 效率低

Uniform Random Number Generator

- 通常每一個電腦系統，都會根據它自己的硬體架構與浮點運算方式，提供專屬此一電腦的隨機亂數產生器的相關內建函數（intrinsic function)
- 根據電腦整數運算與浮點運算的running error，可得0到1之間均勻分布之隨機亂數。
- Example (參考 IBM/SSP, for 32-bit computer)

0~1之間均勻分布之隨機亂數產生器

```
C Fun. ran: 0~1 Uniform Random Generator
C Argument 'ix' is the seed of
C  the random number generating function.
C In order to get really 'random' numbers,
C  initial value of ix must be an odd number.
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
```


Random Number Generator of the Normal Distribution Function

- 大數法則 + 中央極限定理：
  - 任何一種機率分布函數，重覆取樣無窮多次之後，其總和結果都呈現常態分布。
  - 同一種機率分布函數， 重覆取樣 n 次之後，其總和結果之機率函數的期望值與變異數將分別為原機率函數的期望值與變異數的 n 倍。
- Example: (參考 IBM/SSP)
  - 重複選取0~1均勻分布之隨機亂數的和，可近似為常態分布的亂數。
  - 0~1均勻分布的隨機亂數，它的期望值為0.5，變異數為1/12。(會證明嗎？)

平均值為0，標準差為1之常態分布亂數產生器

```
C
C==RANORM=======================================
C
      FUNCTION RANORM(A)
      COMMON IRR    ! seed of random numbers
      A=0.
      DO 999 II=1,16
  999 A=RAN(IRR)+A
      A = A-8.0     ! A=A-(nsum/2) where nsum=16
      A = A*0.866   ! A=A*sqrt(12./nsum)
      RANORM=A
      RETURN
      END
```


General Random Number Generator (Snell, 1975)

- 考慮 $y=f(x)$ ，決定 y_{max}, y_{min}, x_{max}, x_{min}
- 先選取一個 $0\sim1$ 均勻分布的隨機亂數 a
- 再選取一個 $0\sim1$ 均勻分布的隨機亂數 b
- 令 $x_a=a*(x_{max}-x_{min})+x_{min}$
- 令 $y_b=b*(y_{max}-y_{min})+y_{min}$

- 若 $y_b \le f(x_a)$ 則接受亂數 x_a 。 若 $y_b > f(x_a)$ 則放棄 x_a ，重新選取亂數 a 與 b 。
- 重複以上步驟，直到滿足成功條件 $y_b < f(x_a)$ 為止。這樣所產生的亂數 x_a ，即為所求。

求根

- 二分逼近法
  - 適用於單一變數（一維空間）的問題
  - 此法重根時，失效
- 牛頓法（切線法）
  - 可適用於單一變數與多變數聯立方程式的問題（多變數牛頓法，需要具有空間曲線的概念，才能選對方向做切線）
  - 此法重根時，失效
- 解析解
  - 例如：1～4 次多項式或多元一次聯立方程式（求反矩陣）
- 將求根問題轉換為解微分方程式的問題
  - 把已知解拿來當作初始條件，設計一個（組）微分方程式，可快速求得求其他參數狀況下的解。

利用 解微分方程式 求根

- 範例一：
  - 求 $f(x,y)=0$ 之根
  - 把 y 想成是 x 的函數，於是

$$\frac{\partial f}{\partial x}+\frac{\partial f}{\partial y}\frac{dy}{dx}=0 \quad \Rightarrow \quad \frac{dy}{dx}=-\frac{\partial f/\partial x}{\partial f/\partial y}$$

  - 只要先求出一組 (x_0, y_0) 之解，就可輕易求出其他(x_i, y_i) 之解。
  - 適用條件： $\partial f/\partial y \neq 0$

應用解微分方程式的方式求根

- 範例二：（Homotopy method, 芝加哥大學李天岩教授想出的方法）
  - 求 $f(x_1, x_2, x_3, \ldots x_n)=0$ 之根
  - 增加一個參數 t ， 再藉由一個已知解析解的函數 h $(x_1, x_2 \ldots x_n)=0$ ，來建構一個新函數

$$g(x_1, x_2, \ldots x_n, t)=(1-t)*h(x_1, x_2 \ldots x_n)+t*f(x_1, x_2, \ldots x_n)$$

  - 當 $t=0$ 時， $g=0$ 之解，即 $h=0$ 之解
  - 利用範例一的方式，透過解微分方程式的方法，逐漸增加 t 到 $t=1$ ，則 $g=0$ 之解，即 $f=0$ 之解。

內差法與分散法

- Interpolation method（內差法）
  - 利用網格點上的物理量，推算非網格點上的物理量。
- inverse-interpolation method（分散法）
  - 將非網格點上的物理量，適當的分散到四周的網格點上。

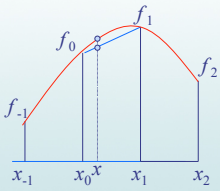
內差法與分散法之應用

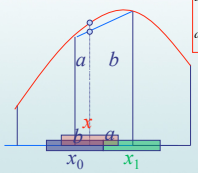
- 內差法：
  內差法可幫助我們，根據網格點上的電場、磁場、速度場之值，來trace電場線、磁場線、或流線。
- 內差法與分散法：
  - 利用內差法與分散法，可將粒子數值模擬計算的N^2計算步驟，藉由網格點的幫助，化簡為$N*M$計算步驟，其中 N 為粒子數， M 為網格數。且$N >> M$。
- 分散法：
  - 利用分散法，可求得粒子在相空間phase space的分布函數$f(x,v)$。藉此可獲得 $f(x,v)$ 的灰階分布圖。

一度空間之 一階與三階內差／分散法

- 一階：$f(x)=(\frac{x_1-x}{x_1-x_0})f_0+(\frac{x-x_0}{x_1-x_0})f_1+O(h^2f^{(2)})=b\,f_0+a\,f_1+O(h^2f^{(2)})$
- 三階：$f(x)=a_{-1}f_{-1}+a_0f_0+a_1f_1+a_2f_2+O(h^4f^{(4)})$

$$a_{-1}=\frac{(x-x_0)(x-x_1)(x-x_2)}{(x_{-1}-x_0)(x_{-1}-x_1)(x_{-1}-x_2)}$$
$$a_0=\frac{(x-x_{-1})(x-x_1)(x-x_2)}{(x_0-x_{-1})(x_0-x_1)(x_0-x_2)}$$
$$a_1=\frac{(x-x_{-1})(x-x_0)(x-x_2)}{(x_1-x_{-1})(x_1-x_0)(x_1-x_2)}$$
$$a_2=\frac{(x-x_{-1})(x-x_0)(x-x_1)}{(x_2-x_{-1})(x_2-x_0)(x_2-x_1)}$$



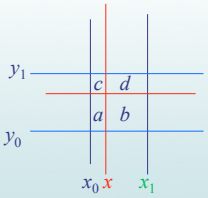


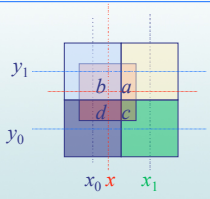
二度空間之 一階內差／分散法

- 一階：

以下Particle-in-Cell的看法，看似好懂，其實不易推廣到高階

$$f(x)=d\,f_{00}+c\,f_{10}+b\,f_{01}+a\,f_{11}$$
$$=(\frac{x_1-x}{x_1-x_0})(\frac{y_1-y}{y_1-y_0})f_{00}+(\frac{x-x_0}{x_1-x_0})(\frac{y_1-y}{y_1-y_0})f_{10}$$
$$+(\frac{x_1-x}{x_1-x_0})(\frac{y-y_0}{y_1-y_0})f_{01}+(\frac{x-x_0}{x_1-x_0})(\frac{y-y_0}{y_1-y_0})f_{11}$$





二度空間之 一階與三階內差／分散法

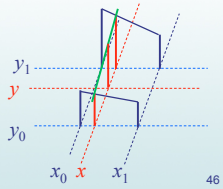
- 一階：

$$f(x)=(\frac{x_1-x}{x_1-x_0})(\frac{y_1-y}{y_1-y_0})f_{00}+(\frac{x-x_0}{x_1-x_0})(\frac{y_1-y}{y_1-y_0})f_{10}$$
$$+(\frac{x_1-x}{x_1-x_0})(\frac{y-y_0}{y_1-y_0})f_{01}+(\frac{x-x_0}{x_1-x_0})(\frac{y-y_0}{y_1-y_0})f_{11}+O(h^2f^{(2)})$$

同理可得三階的內差／分散公式

- 三階：$f(x)=\sum_{i=-1}^{2}\sum_{j=-1}^{2}a_{i,j}f_{i,j}$

係數$a_{i,j}$=？，留待各位回家當作業練習。
另外，三度空間的差分／分散法，
也可以如法炮製，求得。



一階&三階之內差／分散法 效率評估

每次計算參與格點數

	一階	三階
1-D	2	4
2-D	4	16
3-D	8	64

相同準確度下系統網格點數

	一階	三階
1-D	A/s	A/s$^{1/3}$
2-D	A/s^2	A/s$^{2/3}$
3-D	A/s^3	A/s

一階與三階之內差／分散法 效率評估

每次計算相同準確度下運算時間比值（一階／三階）

1st/3rd	s	s=0.01	s=0.001
1-D	1/(2s$^{2/3}$)	>10	50
2-D	1/(4s$^{4/3}$)	>100	2500
3-D	1/(8s^2)	1250	125000

Summary

- 高階優於低階（Shampine and Gordon, 1975）：
  - 高階運算的程式，雖然難寫，但是大量運算時，可以花費較少的CPU時間，獲得相同準確度的結果，因此解題效率反而高於低階運算的程式。
  - 花費的人力越多，可以節省的電腦運算資源也越多！

Summary

- 針對問題的特性，選擇最佳的數值方法
  - 不能只會一種數值方法，妄想一劍走天下！
  - 簡單的問題，要用簡單的方法解決，不要動不動就–殺雞用牛刀！浪費電腦與你自己的寶貴時間！
  - 有些問題，就是那麼複雜，無法簡單化。若想偷懶，用簡單的數值方法解決它，小心得不償失！
- 數值方法的效率與準確度
  - 寫程式時，隨時考慮機器誤差與數值誤差，才不會grabage in, garbage out (GIGO) !
  - 只算一次的計算，或許可以忽略數值方法的效率問題
  - 要重複計算千萬次的模擬計算，一定要仔細考慮所採用之數值方法的效率與準確度問題（錙銖必較）！

參考文獻

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.