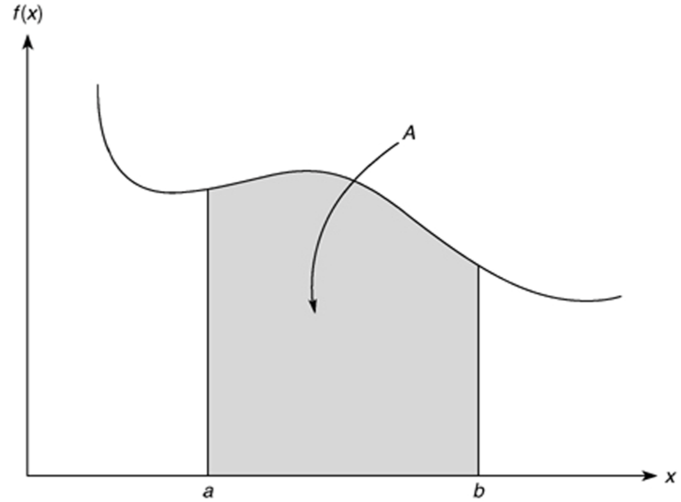
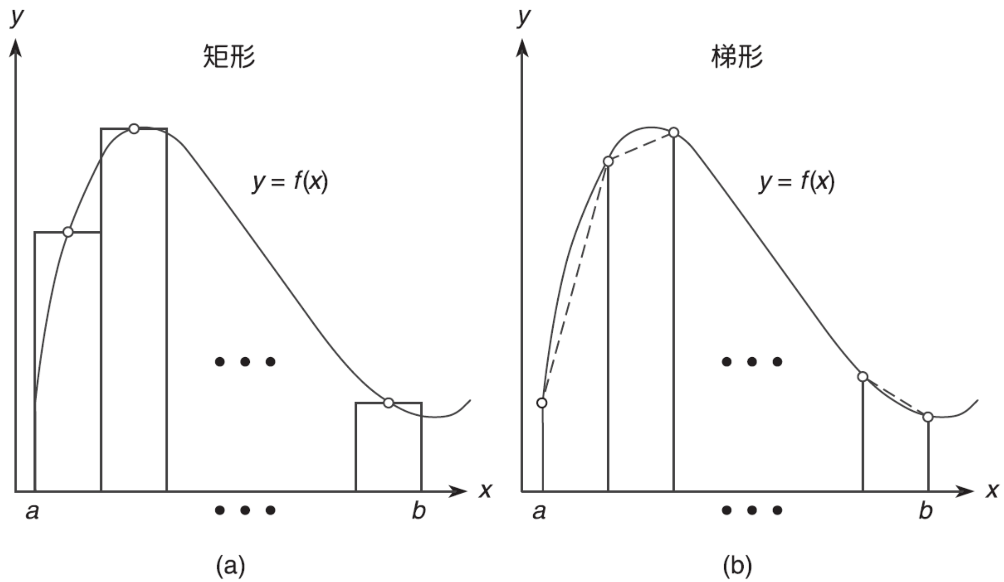
MATLAB
數值微積分與微分方程式求解

數值積分

- $\int_a^b f(x)dx$ 等於由界限範圍 $x = a$ 到 $x = b$ 之間曲線 $f(x)$ 底下的面積





(a) 矩形以及 **(b)** 梯形數值積分的圖解說明

數值積分

- 已知數據點的積分，不知函數 *f*(*x*)：**trapz**

$\boxed{\mathbf{I = trapz(x, y)}}$ （梯形積分法）

x：數據點之 *x* 值所構成的向量
y：數據點之 *f*(*x*) 值所構成的向量

Ex:

```
>>x=[0 10  20  30  40];
>>y=[0.5  0.7  0.9  0.6  0.4];
>>area=trapz(x,y)    %梯形法
area =
    26.5000
```

數值積分

- 已知函數 *f(x)* 之形式：**quad , quadl**

$\boxed{\textbf{I = quad(@fun, a, b)}}$ （適應性辛普森法）

$\boxed{\textbf{I = quadl(@fun, a, b)}}$ （羅伯特二次式）

fun：定義函數的 function m-file 檔名

a ：積分下限

b ：積分上限

數值積分

- **Ex:** $\int_0^1 e^{-x}\cos(x)\,dx$

1. edit fun.m

```
function  y=fun(x)
y=exp(-x).*cos(x);
```

2. 求積分(回到*Matlab Command Window*)

```
area=quadl(@fun,0,1)
```

亦可使用

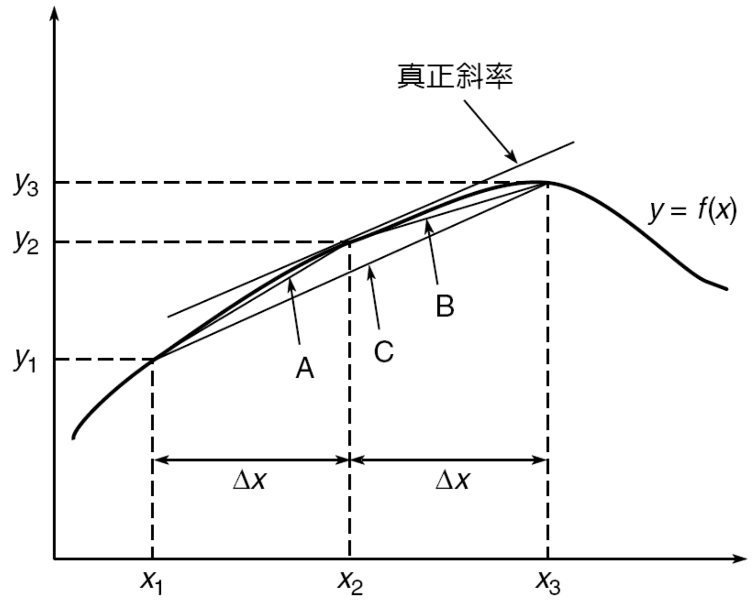
```
area=quadl('exp(-x).*cos(x)',0,1)
```

NOTE: 函數內之數學運算必須使用向量個別元素之運算 **(.* ./ .^)**

(註：比較此結果與利用trapz指令計算之結果)

數值微分

- 已知數據點的微分



- 在 x_2 之微分

$$\text{Definition of Derivative:} \quad \frac{dy}{dx} = \lim_{\Delta x \to 0} \frac{\Delta y}{\Delta x}$$

$$\text{Backward Difference:} \quad m_A = \frac{y_2 - y_1}{x_2 - x_1} = \frac{y_2 - y_1}{\Delta x}$$

$$\text{Forward Difference:} \quad m_B = \frac{y_3 - y_2}{x_3 - x_2} = \frac{y_3 - y_2}{\Delta x}$$

$$\text{Central Difference:} \quad m_C = \frac{1}{2}\left(\frac{y_3 - y_2}{\Delta x} + \frac{y_2 - y_1}{\Delta x}\right) = \frac{y_3 - y_1}{2\Delta x}$$

數值微分

- 可利用 **diff** 函數

$$\begin{aligned} d &= \text{diff}(x) \\ &= [x(2)-x(1), x(3)-x(2), \ldots, x(n)-x(n-1)] \end{aligned}$$

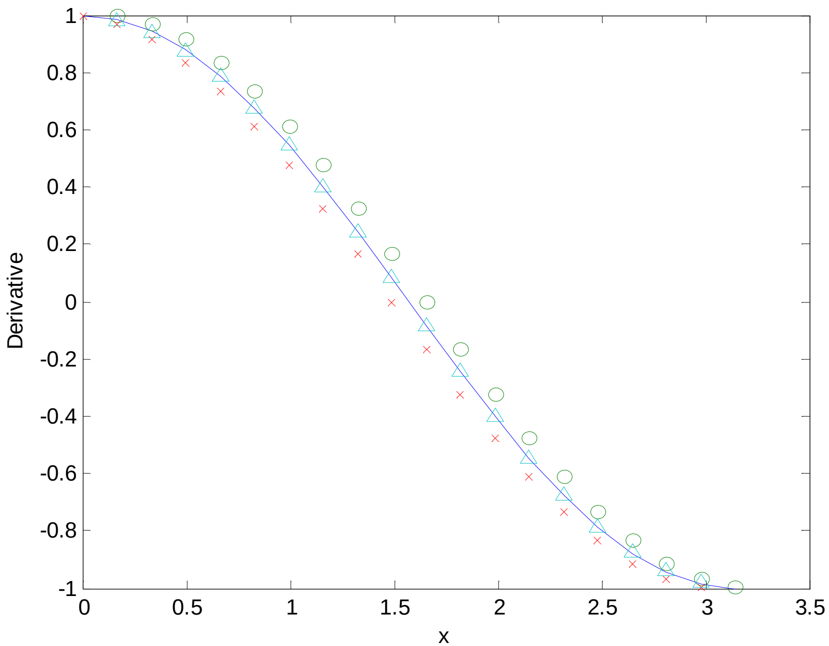
Ex:

```
>>x=0:0.1:1;
>>y=[0.5 0.6 0.7 0.9 1.2 1.4 1.7 2.0 2.4 2.9 3.5];
>>dx=diff(x);
>>dy=diff(y);
>>dydx=diff(y)./diff(x)
```

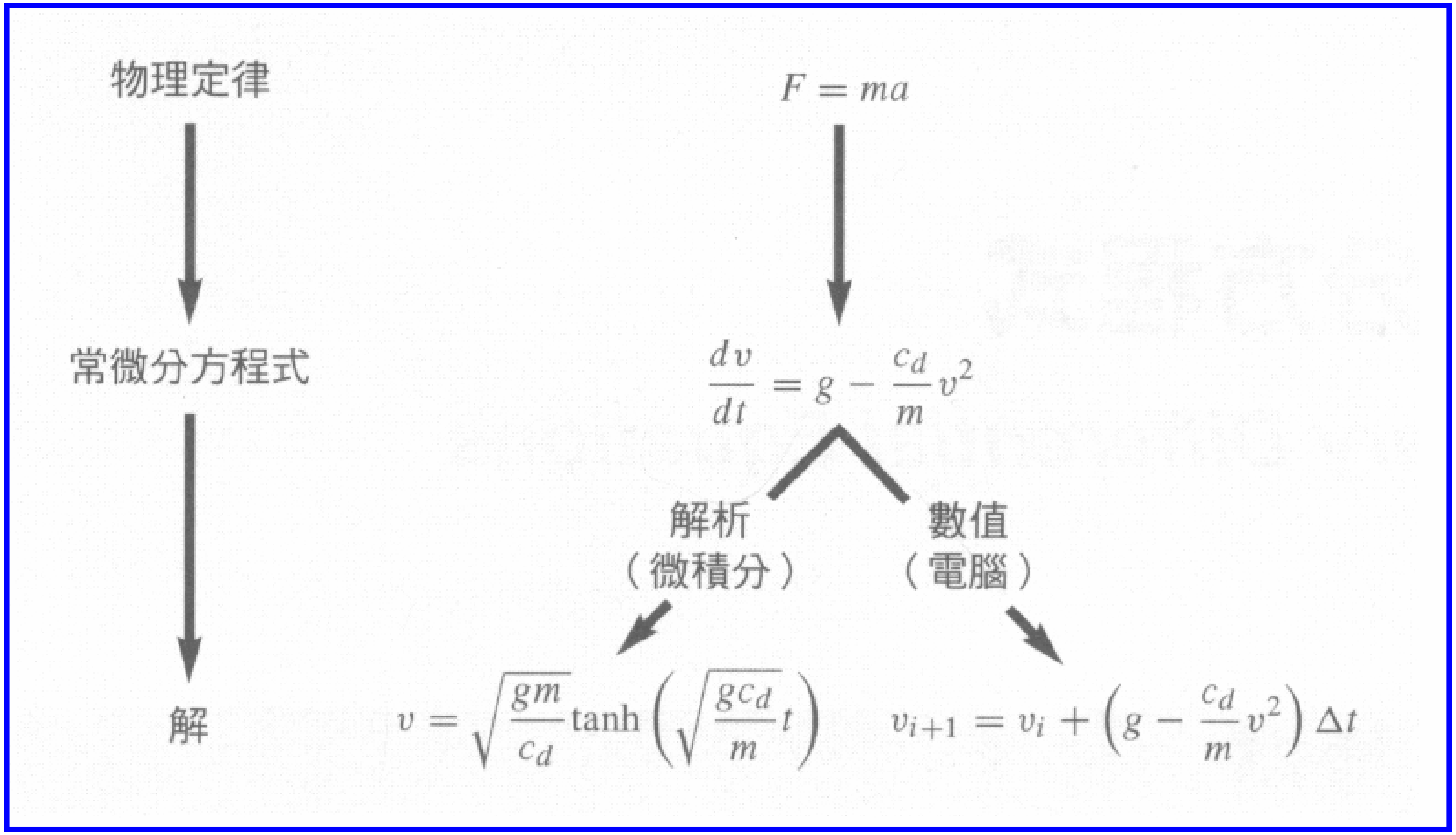
數值微分

Ex: $f(x)=\sin(x),\ \ f'(x)=?\quad x\in[0,\pi]$

```
>> x = linspace(0,pi,20);
>> y = sin(x);
>> d = diff(y)./diff(x);    % backward or forward difference
>> dc = (y(3:end)-y(1:end-2))./(x(3:end)-x(1:end-2));    % central difference
>> dy = cos(x);    % 實際微分值
>> plot(x, dy, x(2:end), d,'o', x(1:end-1), d,'x', x(2:end-1), dc,'^')
>> xlabel('x'); ylabel('Derivative')
```



工程問題中常微分方程式的解



常微分方程式

- 常微分方程式之形式：

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

- 一般解之形式：

$$y_{i+1} = y_i + \phi h$$

ϕ 是斜率或增量函數 (increment function)，被用來自舊值y_i外插到新值y_{i+1}，h為步長大小 (step size)。

- 此方法稱為單步方法 (one-step method)，因為增量函數的值是根據單一點 i 的資訊。

歐拉法

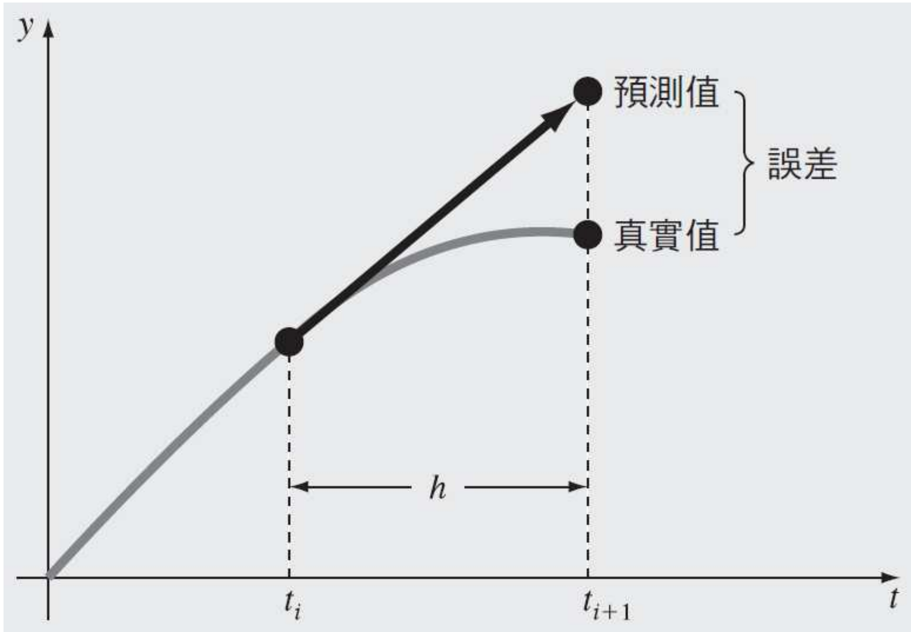
- 在t_i處的一次微分形式提供了直接的斜率估計：

$$\left.\frac{dy}{dt}\right|_{t_i} = f(t_i, y_i)$$

$$\phi = f(t_i, y_i)$$

$$y_{i+1} = y_i + f(t_i, y_i)h$$

- 此公式即稱為歐拉法 (Euler's method)。新的值y利用斜率（等於在t處的一階微分）在步長大小h上做線性外插來得到預測值。



倫基－庫達法

- **倫基－庫達(RK)**法(Runge-Kutta methods)可以達到泰勒級數方式的正確度，而不需計算高階的微分。

$$\phi = a_1k_1 + a_2k_2 + \cdots + a_nk_n$$

ϕ稱為增量函數，代表整個區間的斜率。

- n 表示項數(階數)，a是常數，p和q也都是常數。k之間是遞迴的關係。

$$\begin{aligned}
k_1 &= f(t_i, y_i) \\
k_2 &= f(t_i + p_1h, y_i + q_{11}k_1h) \\
k_3 &= f(t_i + p_2h, y_i + q_{21}k_1h + q_{22}k_2h) \\
&\vdots \\
k_n &= f(t_i + p_{n-1}h, y_i + q_{n-1,1}k_1h + q_{n-1,2}k_2h + \cdots + q_{n-1,n-1}k_{n-1}h)
\end{aligned}$$

古典四階倫基－庫達法

- 最普遍使用的倫基－庫達法是古典四階**RK**法：

$$y_{i+1} = y_i + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4)h$$

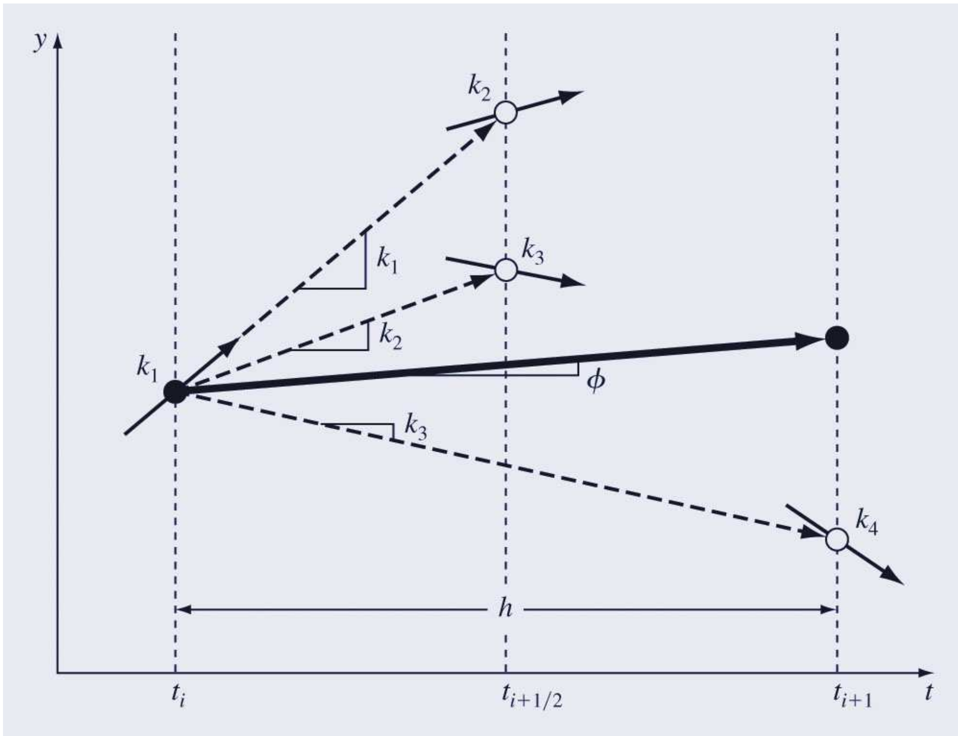
(4個斜率之加權平均)

$$k_1 = f\left(t_i, y_i\right)$$

$$k_2 = f\left(t_i + \frac{h}{2}, y_i + k_1\frac{h}{2}\right)$$

$$k_3 = f\left(t_i + \frac{h}{2}, y_i + k_2\frac{h}{2}\right)$$

$$k_4 = f\left(t_i + h, y_i + k_3 h\right)$$



方程式系統

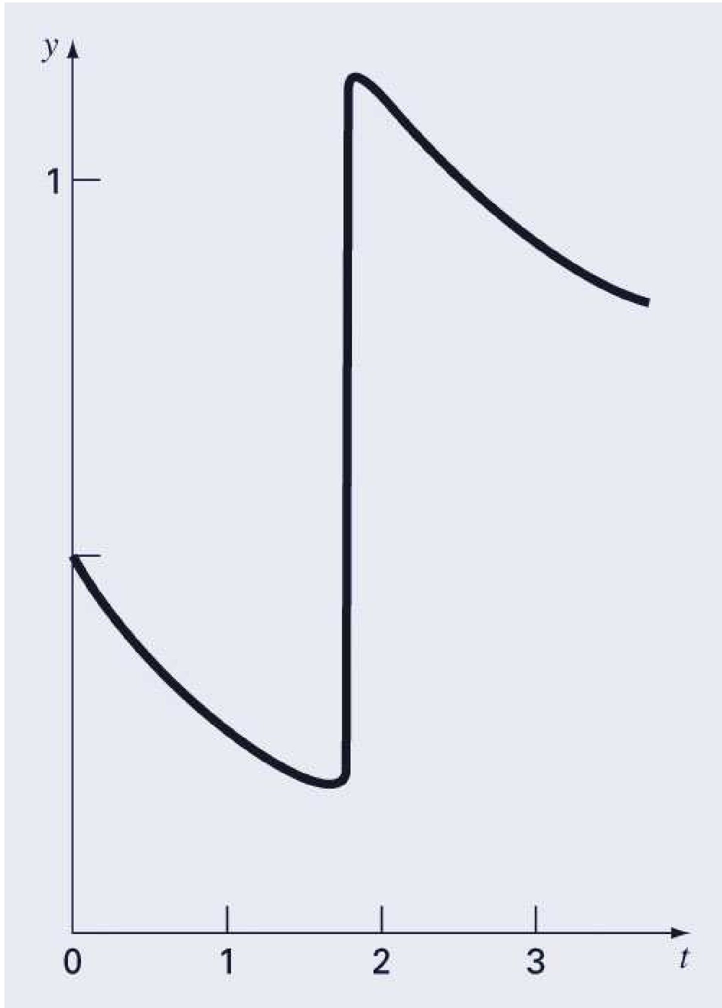
- 許多實際的工程及科學問題需要求解的是聯立常微分方程式系統，而不只是單一方程式。

$$\frac{dy_1}{dt} = f_1(t, y_1, y_2, \cdots, y_n)$$
$$\frac{dy_2}{dt} = f_2(t, y_1, y_2, \cdots, y_n)$$
$$\vdots$$
$$\frac{dy_n}{dt} = f_n(t, y_1, y_2, \cdots, y_n)$$

- 求此系統的解需要n個在開始值t 時已知的起始條件。
- 對於單一方程式之數值方法皆可推展：對於每一個方程式都使用單步方法，然後再進行下一步。

適應性倫基－庫達法

- 自動調整步長大小可以避免過度使用計算量，因此有很大的好處。
- 因為此方法對於解的軌跡具有「適應性」，所以又稱為適應性步長大小控制 (adaptive step-size control)。
- 要利用這個方法，每一步皆需要估計局部截斷誤差。此誤差估計可以當作縮短或增長步長大小的依據。



適應性方法

- 有兩種主要方法可以合併使用適應性步長大小控制與單步方法。
  - 步長減半 (step halving) 的方式是每一步做兩次，一次是一整步，接下來是兩個半步。這兩者之間的差異代表局部截斷誤差的估計值，我們根據這個誤差估計值調整步長大小。
  - 第二個方式稱為嵌入式**RK**法 (embedded RK method)，局部截斷誤差的估計值是根據兩個不同階數RK法預測之間的差異。這是目前主流的使用方式，因為此方法比步長減半法更有效率。

Matlab求解常微分方程式

- 求解微分方程式 $\dfrac{dy(t)}{dt} = f(t, y)$：**ode45, ode23**

[t, y] = ode45(@fun, tspan, y0)

t：時間 (行向量)
y：解所形成之矩陣(每一行為一個應變數，每一列都對應到行向量 t 中的時間)
fun：描述 ode function m-file 檔檔名
tspan：積分時間 ([起始時間 結束時間])
y0：初始值

- **ode定義檔格式：** 輸入為 *t* 與 *y* ，而輸出為表示 *dy/dt* 的行向量

```
function dydt = fun(t, y)

dydt(1) = < Insert a function of t and/or y here. >
dydt(2) = …………..
dydy = dydt'; % 轉換成行向量
```

Matlab求解常微分方程式

Ex:

$$\begin{cases} \dfrac{dy_1}{dt} = y_1 + y_2 e^{-t} \\ \dfrac{dy_2}{dt} = -y_1 y_2 + \cos(t) \end{cases} \qquad y_1(0) = 0, \quad y_2(0) = 1$$

1. edit fun.m

```
function  dydt=fun(t,y)
dydt(1) = y(1)+y(2)*exp(-t);
dydt(2) = -y(1)*y(2)+cos(t);
dydt = dydt';
```

2.求解ode45 ode23 % 回到matlab command window

```
[t,y]=ode45(@fun, [0 10],[0 1]) %[0 10]: t上下限, [0 1]: x的起始值
y1=y(:,1);
y2=y(:,2);
plot(t,y1,'r',t,y2,`g')
```

Matlab求解常微分方程式

- 欲傳其他參數給 ode function m-file

[t, y] = ode45(@fun, tspan, y0, options, p1, p2)

t：時間
y：解所形成之矩陣
fun：描述 ode function m-file 檔檔名
tspan：積分時間 ([起始時間 結束時間])
y0：初始值
options：ode45之設定選項，若不指定則以 [ ] 代替
p1, p2：傳給 fun.m 之參數

- **ode定義檔格式：**輸入為 t , y, p1, p2 ，而輸出為表示 *dy*/*dt* 的行向量

```
function dydt = fun(t, y, p1, p2)

dydt(1) = < Insert a function of t and/or y here. >
dydt(2) = …………..
dydt = dydt'; % 轉換成行向量
```

Matlab求解常微分方程式

求解高階微分方程式

Ex: $y'' + e^{-t} y' + \cos(t) y = \sin(t) e^{-t}, \quad y(0) = 1, y'(0) = 0$

Step 1. 將原式轉成聯立的一階微分方程式，令

$$y_1 = y \Rightarrow y_1' = y'$$

$$y_2 = y' = y_1' \Rightarrow y_2' = y'' = \sin(t) e^{-t} - e^{-t} y_2 - \cos(t) y_1$$

$$\Rightarrow \begin{cases} y_1' = y_2 \\ y_2' = \sin(t) e^{-t} - e^{-t} y_2 - \cos(t) y_1 \end{cases}, \quad y_1(0) = 1, \; y_2(0) = 0$$

Step 2. 用前述方法求解

Matlab ODE 指令

- Matlab用於求解起始值常微分方程式問題的指令

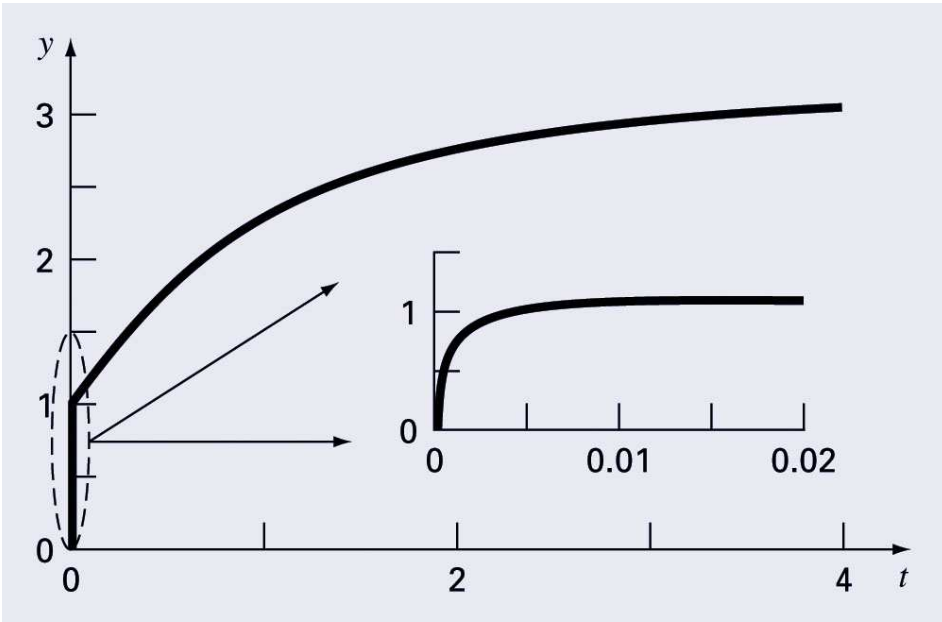
問題形式	指令
Non-stiff ODE	ode45
	ode23
	ode113
Stiff ODE	ode15s
	ode23s
	ode23t
	ode23tb

- Stiff ODE 指的是其內部某些狀態響應快速，而某些則相對具較緩慢動態

勁度

- 勁度系統 (stiff system)表示其具有快速變化以及緩慢變化的部分。
- 勁度系統例子： $\dfrac{dy}{dt} = -1000y + 3000 - 2000e^{-t}$

假使$y(0)=0$，其解 $y = 3 - 0.998e^{-1000t} - 2.002e^{-t}$



Exercise

- 使用ode45來求解下列微分方程式，區間為 t=0 到 20，並畫出 y_1 與 y_2 之圖形。

$$\begin{cases} \dfrac{dy_1}{dt} = a y_1 - 0.6 y_1 y_2 \\ \dfrac{dy_2}{dt} = b y_2 + 0.3 y_1 y_2 \end{cases}, \quad y_1(0) = 2, \ y_2(0) = 1$$

  - Case 1: a = 1.2, b= -0.8
  - Case 2: a = 1.0, b= -0.6

- 求解

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -1000 & -1001 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \qquad \begin{matrix} x_1(0) = 1 \\ x_2(0) = 0 \end{matrix}$$

微分代數系統(DAE)

- 微分代數系統(Differential-Algebraic Equation, DAE)指的是同時伴有微分方程式及代數式的系統。

$$
\begin{aligned}
\dot{y}_1 &= f_1\left(x, y_1, y_2, \cdots, y_N\right) \\
\dot{y}_2 &= f_2\left(x, y_1, y_2, \cdots, y_N\right) \\
&\vdots \\
\dot{y}_n &= f_n\left(x, y_1, y_2, \cdots, y_N\right) \\
\hline
0 &= f_{n+1}\left(x, y_1, y_2, \cdots, y_N\right) \\
0 &= f_{n+2}\left(x, y_1, y_2, \cdots, y_N\right) \\
&\vdots \\
0 &= f_{n+m}\left(x, y_1, y_2, \cdots, y_N\right)
\end{aligned}
$$

I.C. $y_i(0) = y_{i0},\ \ i = 1, 2, \cdots, N$
$(N = n + m)$

$$\boxed{\mathbf{M}\dot{\mathbf{y}} = \mathbf{F}\left(x, \mathbf{y}\right)}$$

x：獨立變數

$$\mathbf{y} = \begin{bmatrix} y_1 & y_2 & \cdots & y_N \end{bmatrix}^T$$

$$\mathbf{F}(\bullet) = \begin{bmatrix} f_1(\bullet) & f_2(\bullet) & \cdots & f_N(\bullet) \end{bmatrix}^T$$

$$\mathbf{M} = \begin{bmatrix} \mathbf{I}_{n\times n} & 0 \\ 0 & 0 \end{bmatrix}$$ (Mass matrix)

Use solver for stiff ODE: ode15s, ode23t

options = odeset('Mass',M,'MassSingular','yes')