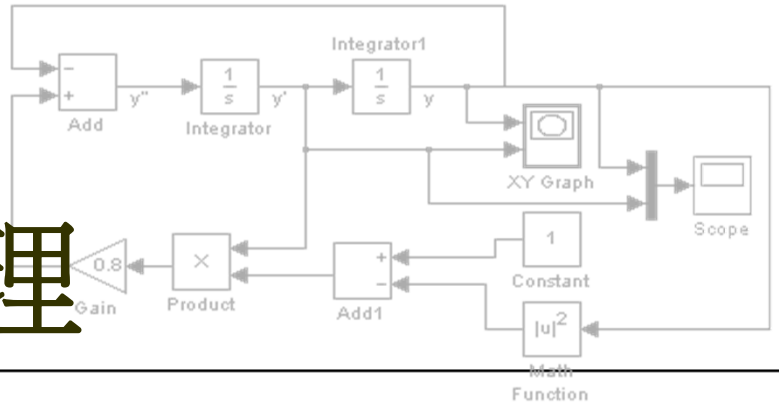


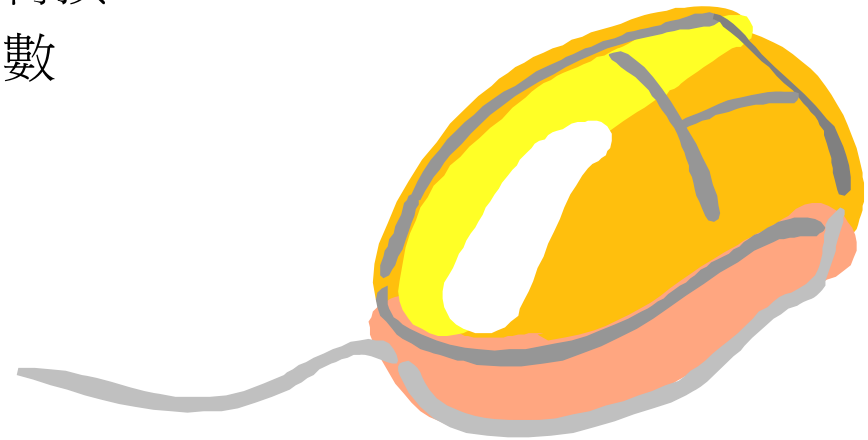
字串的處理

認識字串

認識字串的儲存方式

學習不同數字系統之間的轉換

學習各種與字串相關的函數



認識字串

- 字串可看成是由字元所組成的陣列：

```
>> str1=['M' 'a' 't' 'l' 'a' 'b']
str1 =
Matlab

>> str2='I love Java'
str2 =
I love Java

>> size(str1)
ans =
     1     6
```

- 如要顯示字串裡每一個字元的ASCII碼，可用double函數：

```
>> ascii=double(str1)
ascii =
    77    97   116   108    97    98

>> char(ascii)
ans =
Matlab
```

字串陣列

- 用二維陣列儲存字串時，每一個字串的長度必須相等：

```
>> season=['spring';'summer';'autumn']
season =
spring
summer
autumn

>> season(1:5)
ans =
ssapu

>> season(1,:)
ans =
spring
```

- 以二維陣列儲存字串時，如果字串的長度不相等，則會有錯誤訊息：

```
>> month=['April';'May';'June']
??? Error using ==> vertcat
All rows in the bracketed expression must
have the same number of columns.
```

```
>> month=['April';'May  ';'June ']
month =
April
May
June
```

字串與數字系統的轉換
執行指令字串

- Matlab提供了eval與feval函數，可用來對字串求值：

表 9.3.1　字串求值函數

函 數	說 明
`eval(`*str*`)`	執行字串 *str*
`feval(`*func_name*`,`*arg*`)`	以 *arg* 為引數，執行函數 *func_name*
`feval(`*func_name*`,`arg_1`,`arg_2`,...)`	以 arg_1, arg_2, ... 為引數，執行函數 *func_name*

```
>> eval('32+6')
ans =
    38

>> eval('x=cos(pi/4)')
x =
    0.7071

>> feval('zeros',3)
ans =
     0     0     0
     0     0     0
     0     0     0
```

字串與數值的轉換

- 下表是用來進行數值與字串之間轉換的函數：

表 9.3.2　數值與字串的函數

函 數	說 明
`int2str(x)`	先將 *x* 經四捨五入轉換成整數，再將它們轉換成字串
`num2str(x)`	將 *x* 轉換成字串，並以 4 個位數來顯示
`num2str(x,n)`	將 *x* 轉換成字串，但以 *n* 個位數來顯示
`mat2str(x)`	將陣列 *x* 轉換成 Matlab 的表示方式，但以字串來顯示
`str2num(str)`	將字串 *str* 以 eval 函數求值，如果不能轉換，則回應空陣列
`str2double(str)`	將字串 *str* 轉換成數值，如果不能轉換，則回應 NaN

```
>> str1=int2str(1024)
str1 =
1024

>> str2num('123456')
ans =
      123456

>> str2num('123+456')
ans =
   579
```

不同數字系統的轉換

- 下表列出了各種進位系統之間的轉換函數：

表 9.3.3　不同數字系統的轉換函數

函 數	說 明
`dec2bin(`*x*`)`	將 10 進位的整數 *x* 轉換成 2 進位的字串
`dec2hex(`*x*`)`	將 10 進位的整數 *x* 轉換成 16 進位的字串
`bin2dec(`*bin_str*`)`	將 2 進位的字串 *bin_str* 轉換成 10 進位
`hex2dec(`*hex_str*`)`	將 16 進位的字串 *hex_str* 轉換成 10 進位
`dec2base(`*x*`,`*base*`)`	將 10 進位的整數 *x* 轉換成 *base* 進位的字串
`base2dec(`*str*`,`*base*`)`	將 *base* 進位的字串 *str* 轉換成 10 進位的整數

```
>> dec2bin(1122)
ans =
10001100010

>> bin2dec('10001100010')
ans =
      1122

>> hex2dec('AA5F')
ans =
      43615
```

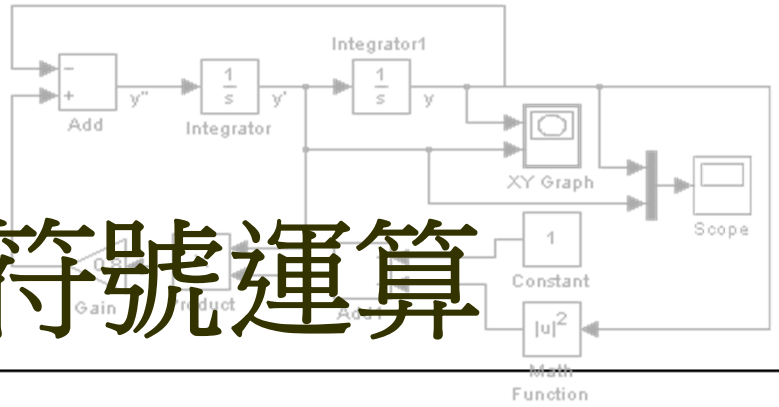
字串處理函數

- 下表列出了常用的字串處理函數：

表 9.4.1　字串處理函數

函 數	說 明
`upper(`*str*`)`	將字串 *str* 轉換成大寫
`lower(`*str*`)`	將字串 *str* 轉換成小寫
`deblank(`*str*`)`	將字串 *str* 後面的空白字元全部刪除
`strcmp(`str_1`,`str_2`)`	比較字串 str_1 與 str_2 是否相等，若是，則回應 1，否則回應 0
`strncmp(`str_1`,`str_2`,`*n*`)`	比較字串 str_1 與 str_2 在第 *n* 個位置的字元是否相等，若是，則回應 1，否則回應 0
`findstr(`*str*`,`*s*`)`	找出字串 *str* 裡，子字串 *s* 所出現的位置
`strrep(`*str*`,`s_1`,`s_2`)`	將字串 *str* 裡，子字串 s_1 代換成字串 s_2
`strtok(`*str*`,`*token*`)`	將字串 *str* 裡，字元 *token* 之後的字串全都刪掉。若省略 *token*，則以空白鍵當 *token*
`strvcat(`str_1`,`str_2`)`	將字串垂直排列

```
>> upper('Merry Christmas')
ans =
MERRY CHRISTMAS

>> str1=deblank('snoopy  ')
str1 =
Snoopy

>> findstr('kitty','t')
ans =
4

>> strvcat('hello','kitty')
ans =
hello
kitty
```



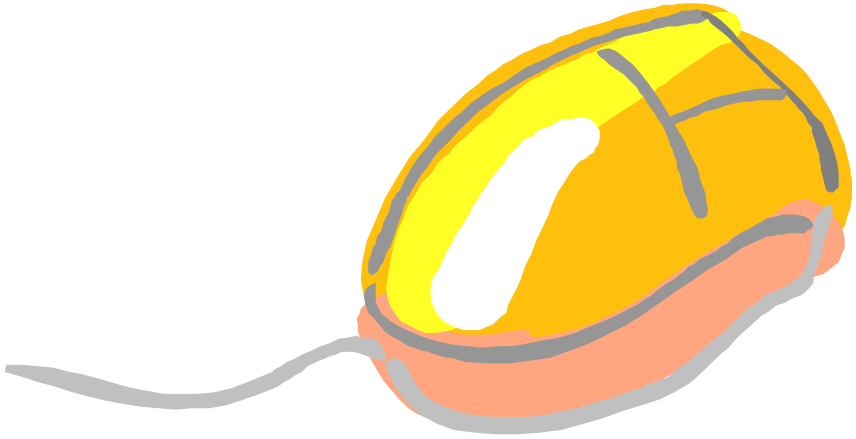
Matlab的符號運算

符號運算 – Symbolic Calculation

基本的代數運算

求解方程式

線性代數的符號運算



基本認識

- 數值運算（numerical calculation）：

  例如利用quad函數來計算積分，或者是利用fzero函數求解方程式

- 符號運算（symbolic calculation）：

  例如計算積分式

$$\int \sin x \, dx = \cos x + C$$

  或者是微分式

$$\frac{d}{dx}\cos^2 x = 2\cos x \cdot \sin x$$

符號物件

- 要進行符號運算，必須先建立符號物件

表 14.1.1　建立符號物件的函數

函 數	說 明
s=sym(*expr*)	建立一個符號物件 *s*
x=sym('*x*')	建立符號變數 *x*
syms *x y z*,…	同時建立符號變數 *x*,*y*,*z*,…
findsym(*expr*)	查詢運算式 *expr* 裡的符號變數

```
>> a=sym(3/5)
 a =
3/5

>> a+1/5
 ans =
4/5
```

- sym函數也可以用來建立符號變數與符號運算式：

```
>> x=sym('x')
x =
X

>> f=2*x^2+3*x+1
f =
2*x^2+3*x+1

>> b=sym('beta')
b =
Beta

>> b^2+sqrt(b)
ans =
beta^2+beta^(1/2)
```

建立符號式陣列

- 陣列裡的元素也可以是符號變數，如下面的範例：

```
>> syms a b c d e
>> m=[a 0 d;b 1 0;c d e]
m =
[ a, 0, d]
[ b, 1, 0]
[ c, d, e]

>> det(m)
ans =
a*e+b*d^2-c*d
```

建立複數變數

- sym函數也可以用來指定變數是否為實數：

```
>> syms x y real
>> imag(x)
ans =
0

>> z=x+i*y
z =
x+i*y

>> abs(z)
ans =
(x^2+y^2)^(1/2)
```

任意精度的計算

- 符號運算工具箱裡提供了三種不同的精度運算：
  - ✓ numeric　Matlab內建的浮點數運算
  - ✓ rational　Maple的精確值分數運算
  - ✓ vpa　Maple的任意精度計算

表 14.1.2　vpa 計算的相關函數

函 數	說 明
`digits(n)`	設定 *vpa* 計算的精度為 *n* 個數字，
`digits`	顯示目前的計算精度
`vpa(s)`	以目前設定的精度計算運算式 *s*
`vpa(s,n)`	以 *n* 個位數的精度計算運算式 *s*

```
>> gld=sym('(1+sqrt(5))/2')
 gld =
 (1+sqrt(5))/2

>> vpa(gld)
 ans =
1.6180339887498948482045868343656

>> digits(100)
>> vpa(gld)
ans =
1.6180339887498948482045868343656381177203091798057628621354486227052604628189024497072072041893911
38

>> vpa(pi)
ans =
3.1415926535897932384626433832795028841971693993751058209749445923078164062862089986280348253421170
68
```

基本代數運算
代數式的基本處理

- 基本的代數處理包含了因式分解、展開、化簡等：

表 14.2.1　代數式的展開與因式分解

函 數	說 明
`expand(`*expr*`)`	將代數式 *expr* 展開
`factor(`*expr*`)`	將代數式 *expr* 做因式分解
`collect(`*expr*`,`*v*`)`	將代數式 *expr* 排列成變數 *v* 的多項式

```
>> syms a b
>> expand((a+b)^3)
ans =
a^3+3*a^2*b+3*a*b^2+b^3

>> factor(ans)
ans =
 (a+b)^3

>> expand(cos(a+b))
ans =
cos(a)*cos(b)-sin(a)*sin(b)

>> expand(sin(2*a))
ans =
2*sin(a)*cos(a)
```

利用simplify或subs指令可化簡代數式：

表 14.2.2　代數式的化簡與代換函數

函 數	說 明
`simplify(`*expr*`)`	化簡代數式 *expr*，若無法再化簡成更精簡的式子，則回應原式
`subs(`*expr*`,`*old*`,`*new*`)`	將代數式 *expr* 的 *old* 以 *new* 來取代

```
>> syms a b x
>> simplify(exp(4*log(sqrt(a+b))))
ans =
(a+b)^2

>> subs(x^2+2*x+1,a+1)
ans =
(a+1)^2+2*a+3
```

多項式與分式的相關運算

- 下面列出了多項式與分式的一些相關運算函數：

表 14.2.3　多項式與分式相關的運算

函 數	說 明
`poly2sym(`*vect*`,`*x*`)`	以向量 *vect* 為多項式係數
`sym2poly(`*symp*`)`	將多項式轉換成由係數所組成的陣列
`coeffs(`*poly*`,`*x*`)`	以 *x* 為多項式的變數，取出多項式的係數
`[`*n*`,`*d*`]=numden(`*expr*`)`	分別取出分式的分子與分母
`[`*q*`,`*r*`]=quorem(`*expr*`)`	計算分式的商與餘數

```
>> syms x y
>> p=poly2sym([1 3 2 3 7],x)
p =
x^4+3*x^3+2*x^2+3*x+7

>> sym2poly(p)
ans =
     1     3     2     3     7

>> coeffs(p)
ans =
[ 7, 3, 2, 3, 1]

>> r=(x^3+2*x^2+3*x+6)/(x^2+3);
>> n=numden(r)
n =
x^3+2*x^2+3*x+6
```

方程式的求解
簡單的solve函數

- fzero只能求得數值解， solve函數則可求得符號解

表 14.3.1　solve 函數的基本用法

函 數	說 明
a = solve(eq)	解方程式 $eq = 0$
a = solve(eq,var)	指定變數 var，解方程式 $eq = 0$

```
>> syms a b c x y
>> eq=x^3-4*x^2+x+6;
>> sol=solve(eq)

sol =
 -1
  2
  3
```

```
>> eq=a*x^2+b*x+c;
>> sol=solve(eq,x)
sol=
 1/2/a*(-b+(b^2-4*a*c)^(1/2))
 1/2/a*(-b-(b^2-4*a*c)^(1/2))

>> solve(sin(x)-1,x)
ans =
1/2*pi

>> solve(2*asin(3*x)-acos(5*x))
ans =
-5/36+1/36*97^(1/2)

>> solve(2*cos(3*x)^2-4*y,x)
ans =
          1/3*acos(2^(1/2)*y^(1/2))
 1/3*pi-1/3*acos(2^(1/2)*y^(1/2))
```

聯立方程式的解

- solve也可以解聯立方程式，如下面的語法：

表 14.3.2　solve 函數的基本用法

函 數	說 明
a = solve($eq_1, eq_2, \ldots, eq_n$)	解聯立方程式 $eq_1, eq_2, \ldots, eq_n$
a = solve($eq_1, eq_2, \ldots, eq_n, var_1, var_2, \ldots, var_n$)	解聯立方程式 $eq_1, \ldots, eq_n$

```
>> sol=solve(a*x+b*y-c,d*x+e*y-f,x,y)
sol =
    x: [1x1 sym]
    y: [1x1 sym]

>> [sol.x;sol.y]
ans =
 (e*c-f*b)/(-d*b+a*e)
 (-d*c+a*f)/(-d*b+a*e)
```

線性代數的運算
基本的矩陣運算

- 符號運算工具箱裡提供了一些函數，可用來進行矩陣的符號運算：

表 14.4.1　陣列元素的提取函數

函 數	說 明
`diag(A)`	取出陣列 A 的主對角線（main diagonal）元素
`triu(A)`	取出陣列 A 之主對角線以上之元素
`tril(A)`	取出陣列 A 之主對角線以下之元素
`det(A)`	求出矩陣 A 的行列式值
`inv(A)`	求出矩陣 A 的反矩陣
`rref(A)`	求出矩陣 A 的簡約列-梯形矩陣

```
>> syms a b c d e f
>> A=[a b d;c e 0;b f 1]
A =
[ a, b, d]
[ c, e, 0]
[ b, f, 1]

>> diag(A)
ans =
 a
 e
 1
```

固有值與固有向量運算

- eig函數可以求得固有值與固有向量的符號式：

表 14.4.3　eig 函數的用法

函 數	說 明
lambda=eig(*A*)	計算矩陣 *A* 的固有值
[*v*,*d*]=eig(*A*)	計算矩陣 *A* 的固有值與固有向量
lambda=eig(vpa(*A*))	使用任意精度的數字來計算固有值
[*v*,*d*]=eig(vpa(*A*))	使用任意精度的數字來計算固有值與固有向量

```
>> syms a
>> A=sym([0 a;1 0])
A =
[ 0, a]
[ 1, 0]

>> eig(A)
ans =
  a^(1/2)
 -a^(1/2)

>> [v,d]=eig(A)
v =
[  a^(1/2),  -a^(1/2)]
[         1,         1]
d =
[  a^(1/2),          0]
[         0, -a^(1/2)]
```

矩陣的秩與空間

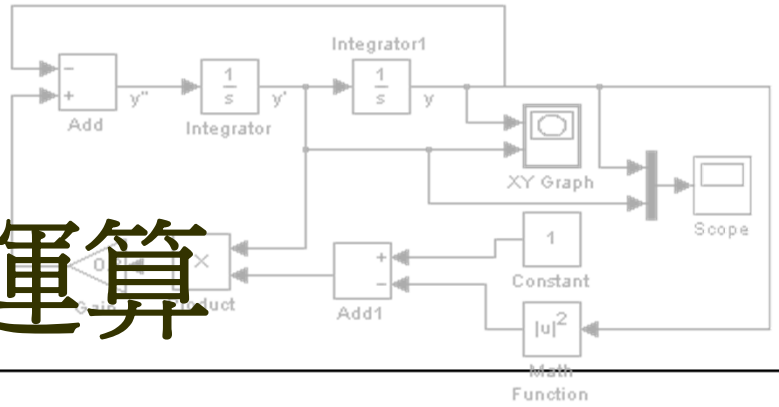
- Matlab提供了一些函數，可計算行空間、零核空間與秩等。

表 14.4.4　計算矩陣的行空間、零核空間與秩

函 數	說 明
colspace(*A*)	計算矩陣 *A* 的行空間
null(*A*)	計算矩陣 *A* 的零核空間
rank(*A*)	計算矩陣 *A* 的秩

```
>> A=sym([1 1 4 2;0 1 3 2;3 2 1 8])
A =
[ 1, 1, 4, 2]
[ 0, 1, 3, 2]
[ 3, 2, 1, 8]
```

```
>> colspace(A)
ans =
[ 0, 0, 1]
[ 1, 0, 0]
[ 0, 1, 0]

>> rank(A)
ans =
3

>> na=null(A)
na =
 -1
 -7
  1
  2
```



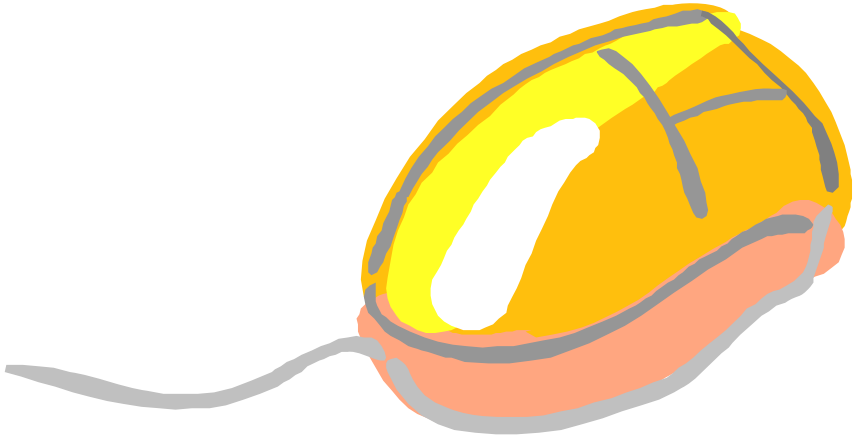
進階符號運算

極限的運算

微分與積分的運算

微分方程式的運算

拉普拉氏與傅立葉轉換



極限的運算

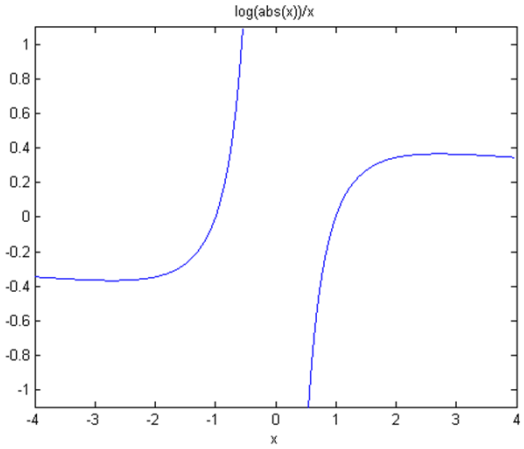
- Matlab以limit函數來計算 $\lim_{x\to a} f(x)$

表 15.1.1　極限計算函數

函 數	說 明
`limit(f,x,a)`	計算極限 $\lim_{x\to a} f(x)$
`limit(f,x,a,'left')`	計算極限 $\lim_{x\to a^-} f(x)$
`limit(f,x,a,'right')`	計算極限 $\lim_{x\to a^+} f(x)$

```
>> syms x
>> limit(sqrt(x^2+2)/(3*x-6),inf)
ans =
1/3
```

```
>> limit(log(abs(x))/x,x,0,'right')
ans =
-Inf

>> ezplot(log(abs(x))/x,[-4,4])
```




```
>> limit(log(abs(x))/x,x,inf)
ans =
0
```

微分運算

- 如要計算微分，可用diff函數：

表 15.2.1　微分函數

函 數	說 明
`diff(`*expr*`,`*v*`)`	計算 *expr* 對符號變數 *v* 的微分
`diff(`*expr*`,`*v*`,`*n*`)`	計算 *expr* 對符號變數 *v* 的 *n* 次微分

```
>> syms a b c x y
>> diff(x^3,x)
ans =
3*x^2

>> diff(diff(x^3,x),x)
ans =
6*x
```

```
>> diff(x^3,x,2)
ans =
6*x

>> diff(sin(x^2),x,3)
ans =
-8*cos(x^2)*x^3-12*sin(x^2)*x

>> diff(sin(a*y),y)
ans =
cos(a*y)*a
```

積分運算

- Matlab是以int函數來進行符號積分：
- int在處理積分運算時，除了積分變數之外，所有的符號均視為常數。

表 15.3.1　計算積分的函數

函 數	說 明
`int(`*expr*`,`*x*`)`	計算 $\int \text{expr}\, dx$
`int(`*expr*`,`*x*`,`*a*`,`*b*`)`	計算 $\int_a^b \text{expr}\, dx$

```
>> syms a b c n x y z
>> int(sin(x)+tan(x),x)
ans =
-cos(x)-log(cos(x))

>> int(x^-1,x)
ans =
log(x)

>> int(cos(x)*sinh(x),x)
ans =
1/4*cos(x)*exp(x)+1/4*exp(x)*sin(x)+
1/4*exp(-x)*cos(x)-1/4*exp(-x)*sin(x)

>> int(int(int(y*z*sin(x),x),y),z)
ans =
-1/4*y^2*z^2*cos(x)
```

數列與級數
級數的計算

- 如果想計算數列相加的符號式，可用symsum函數
- symsum為symbolic summation的縮寫

表 15.4.1　計算加總的符號式函數

函 數	說 明
`symsum(`*expr*`,`*x*`,`*a*`,`*b*`)`	指定變數為 *x*，計算 $\sum_{x=a}^{b}$expr

```
>> syms k n x
>> symsum(x^2,x,1,10)
ans =
385

>> simplify(symsum(x^2,1,n))
ans =
1/3*n^3+1/2*n^2+1/6*n

>> f=symsum(1/2^x,x,1,n)
f =
-2*(1/2)^(n+1)+1

>> limit(f,n,inf)
ans =
1
```

泰勒展開式

- 數學函數的泰勒展開式可以使用taylor函數：

表 15.4.2　泰勒展開式的計算

函 數	說 明
`taylor(`f`,`n`,`v`)`	指定變數為 v，計算 $f(v)$ 對 $v=0$ 的泰勒展開式，展開到 $n-1$ 階
`taylor(`f`,`n`,`v`,`a`)`	同上，但是對 $x=a$ 展開

```
>> syms a x
>> taylor(exp(x),x,0,6)
ans =
1+x+1/2*x^2+1/6*x^3+1/24*x^4+1/120*x^5
```

求解微分方程式

- Matlab是以dsolve函數求解微分方程式：

表 15.5.1　求解微分方程式

函 數	說 明
dsolve('eq_1,eq_2,...','$cond_1$,$cond_2$,...','v')	解微分方程式 eq_1,eq_2,...
dsolve('eq_1','eq_2',...,'$cond_1$','$cond_2$',...,'v')	同上

一階微分方程式

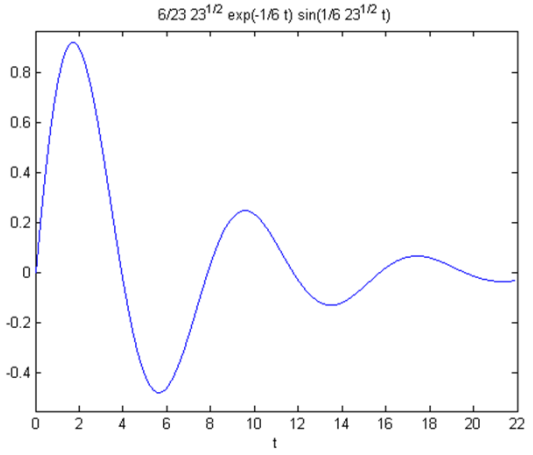
- 下面是以dsolve求解一階微分方程式的範例：

```
>> syms a x y t
>> dsolve('Dy-2*exp(t)+1=0')
ans =
2*exp(t)-t+C1

>> dsolve('Dy-2*exp(t)+1=0','y(0)=3')
ans =
2*exp(t)-t+1
```

二階微分方程式

- 下面是求解二階線性常係數微分方程式的範例：

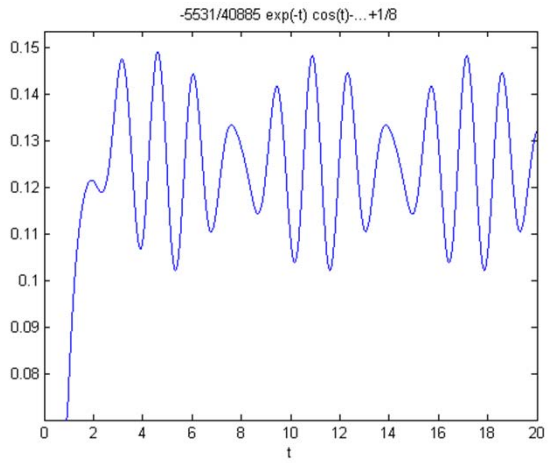
```
>> syms a x y t
>> dsolve('D2y+2*Dy+4*y=0')
ans =
sin(2*t)*C2+cos(2*t)*C1-1/2

>> dsolve('3*D2y+Dy+2*y=0','y(0)=0,Dy(0)=1')
ans =
6/23*23^(1/2)*exp(-1/6*t)*
sin(1/6*23^(1/2)*t)

>> ezplot(ans,[0,22])
```



高階微分方程式

- 下面是求解高階線性常係數微分方程式的範例：

```
>> syms t y
>> eq='D3y+2*D2y+2*Dy=sin(4*t)+cos(5*t)';
>> ini='y(0)=0,Dy(0)=0,D2y(0)=0';
>> sol=dsolve(eq,ini)
>> ezplot(sol,[0,20])
```



聯立微分方程式

- dsolve函數也可以解出聯立微分方程式的符號解：

```
>> syms t x y
>> dsolve('Dx=y-x','Dy=-x','x(0)=1','y(0)=0')
ans =
    x: [1x1 sym]
    y: [1x1 sym]

>> sol=[ans.x; ans.y]
sol =
1/2*exp(-1/2*t)*(-2/3*sin(1/2*3^(1/2)*t)*
3^(1/2)+2*cos(1/2*3^(1/2)*t))-2/3*exp(-
1/2*t)*sin(1/2*3^(1/2)*t)*3^(1/2)
```

拉普拉氏轉換與傅利葉轉換

拉普拉氏轉換

- 拉普拉氏轉換的定義為

$$\int_0^{\infty} e^{-st} f(t)\,dt = F(s)$$

- laplace與ilaplace可計算拉普拉氏轉換與其反運算：

表 15.6.1 計算拉普拉氏與反拉普拉氏轉換

函 數	說 明
laplace(f,t,s)	計算 $f(t)$ 的拉普拉氏轉換
ilaplace(F,s,t)	計算 $F(s)$ 的反拉普拉氏轉換

```
>> syms a s t
>> laplace(cos(a*t),t,s)
ans =
s/(s^2+a^2)

>> ilaplace(ans,s,t)
ans =
cos(a*t)

>> laplace(dirac(t-4)*sin(a*t),t,s)
ans =
sin(4*a)*exp(-4*s)

>> laplace(diff(sym('F(t)')))
ans =
s*laplace(F(t),t,s)-F(0)
```

傅立葉轉換

- 函數 $f(t)$ 的傅立葉轉換之定義為

$$F(\omega) = \int_{-\infty}^{\infty} f(t) e^{-i\omega t} dt$$

$F(\omega)$ 的反傅立葉轉換則定義為

$$f(t) = \frac{1}{2\pi} \int_{-\infty}^{\infty} F(\omega) e^{i\omega t} \, dt$$

- fourier和ifourier兩個函數，可求得傅立葉轉換和它的逆運算：

表 15.6.2 計算傅立葉轉換和它的逆運算

函 數	說 明
`fourier(`f,t,w`)`	計算 $f(t)$ 的傅立葉轉換
`ifourier(`F,w,t`)`	計算 $F(w)$ 的傅立葉轉換

```
>> syms t w
>> fourier(1,t,w)
ans =
2*pi*dirac(w)

>>  fourier(exp(-4*t^2),t,w)
ans =
1/2*pi^(1/2)*exp(-1/16*w^2)

>> ifourier(ans,w,t)
ans =
exp(-4*t^2)
```