Maxima 在線性代數的應用

蔡炎龍
政治大學應用數學系

August 20, 2006

第 1 節 簡介

這篇文章，是介紹 Maxima 這套數學軟體，在學習線性代數的應用。

Maxima 是一個所謂的「電腦代數系統」（Computer Algebra System, CAS），這種系統比較為人熟知的還有 Mathematica 和 Maple 等等。我們選定 Maxima 做為我們使用的程式，主要有三個原因：

免費 Maxima 是免費，又是各平台都有的。所有的人可以在自己的電腦上練習。

功能完整 Maxima 雖然不要錢，並不代表不好。Maxima 不論計算或圖形功能都十分完整。事實上，Maxima 是最早的全功能 CAS 系統 Macsyma 的後代。

具代表性 許多新的 CAS 系統，如 Maple, Mathematica 都多少受到 Macsyma 的啟發。所以學會 Maxima，要學會 Maple 或 Mathematica 等軟體都是很容易的事。

這篇文章主要是介紹線性代數相關功能。我們不假設同學已會基本的 Maxima 使用方式，所以我們會用到的概念，也許不純粹是線性代數的，也會一併介紹。專就線性代數而言，我們要會的其實並不多。想要快速進入狀況，可以跳過前面的部份，直接看線性代數相關指令，在操作上有問題時，再回頭看有問題部份的相關說明即可。

如果同學們比較喜歡使用 Mathematica，Maple，或是 Matlab 等商業軟體也是可以的。我們系上的電腦室有提供這些軟體，可以上機試用看看。

第 2 節 基本概念

我們先介紹一下 Maxima 操作的方式。

2.1 Maxima 當計算機

我們先來看，如果我們要把 Maxima 當計算機用，會是什麼情況？

```
(%i1) 1+1;
(%o1) 2
(%i2) 3*4*7;
(%o2) 84
(%i3) 9/3;
(%o3) 3
```

到目前為止，似乎還沒什麼特別。除了可以做複雜一點點的運算，和平常的計算機或數值計算軟體也沒什麼不同。以下的例子就不一樣了：

```
(%i4) 7/3;
```
(%o4) $\frac{7}{3}$

```
(%i5) 1/2 + 2/3;
```
(%o5) $\frac{7}{6}$

從(D4)我們看到，7/3 這種運算， Maxima 不是告訴我們 2.3333...，而是分數的形式！難道 Maxima 真的懂分數？不要懷疑，這就是所謂電腦代數系統 (CAS) 的特長。我們可以像 (D5) 的例子一樣，輸入個分數的四則運算試試即知。

如果堅持要用浮點數，那只要加個 `float` 指令即可：

```
(%i6) float(7/3);
(%o6) 2.333333333333333
```

為了完整，我們順便再介紹指數，根號，階乘表示法：

```
(%i7) 2^10;
(%o7) 1024
(%i8) sqrt(9);
(%o8) 3
(%i9) 5!;
(%o9) 120
```

我們可以看出，這些運算不是自然的數學符號，就是和我們平常電腦程式語言的寫法。

2.2 指令結尾

在上面的例子中，我們發現，在 Maxima 下指令，結束時一定要打上分號「;」，讓 Maxima 知道我們下的指令已結束。為什麼要多這一個動作，主要是為了有時打比較長的指令可以換行之故。

另一個結束方式是打入「＄」的符號。不同於分號的地方是「運算結果不會顯示出來」：

```
(%i10) 2+3$
(%i11) 2+3;
(%o11) 5
```

有一些 CAS 程式，如 Matehmatica 是用分號表示不顯示運算結果。不過 Maxima 中分號已用上，必需用其他字元。

2.3 離開 Maxima

離開 Maxima 打入 “quit();” 即可。

當然，很多人可能會覺得奇怪，為什麼不是打入 “quit” 就好了呢？原來像這種程序導向的語言，什麼動作其實都是執行一個函數。所以我們事實上是執行一個叫「離開」的函數。這函數沒有引數，所以就成了 quit() 的形式。

2.4 結果的引用

我們時常會需要引用前面的結果，這時就用百分比符號 “%” 。比方說：

(%i12) 7/3;

(%o12) $\frac{7}{3}$

(%i13) float(%);

(%o13) 2.333333333333333

Maxima 也可以指定使用第幾個輸出的結果，不過自己定一個標籤可能是最好的方式。比方說，我們可以這樣用：

```
(%i14) myresult:34+(65*72)/119;
```

(%o14) $\frac{8726}{119}$

```
(%i15) float(myresult);
```

(%o15) 73.32773109243698

2.5 重要常數

Maxima 當然有內建 e 或是 π 常常用到的數，只是表示法奇怪一點。e 是 `%e` 而 π 是 `%pi`。

2.6 定義變數

Maxima 定義變數的想法有點特別，在定義一個變數時，其時是**給某個數字、矩陣，或想要定義的任何式子等等一個標籤**。讓我們來看幾個例子：

```
(%i16) a:  37;
(%o16) 37
(%i17) a
(%o17) 37
(%i18) b:  22+100*(375-128);
(%o18) 34722
(%i19) a+b;
(%o19) 24759
```

2.7 函數

Maxima 函數的定義和使用非常直覺，我們看幾個例子就知道：

```
(%i20) f(x) := 3*x^2 + 5;
```

(%o20) $f(x) := 3x^2 + 5$

```
(%i21) f(2);
```
(%o21) 17
```
(%i22) g(x,y) := sin(x)*cos(y);
```
(%o22) $g(x,y) := \sin(x) * \cos(y)$
```
(%i23) g(2*%pi,4);
```
(%o23) 0

重點就是，在定義函數時要用 “:=” 去定義。 比較一下和變數定義的不同，想想為什麼要有兩種不一樣的定義方式。

第 3 節 進階使用

3.1 列式而不運算

我們先計算一個瑕積分，用到無窮大的部份 Maxima 是以 `inf` 表示：

```
(%i1) integrate(%e^(-x^2),x,0,inf);
```
(%o1) $\dfrac{\sqrt{\pi}}{2}$

還記得這在微積分是怎麼積出來的嗎？Maxima 居然會積！不過，今天這不是我們的重點。今天重點是，有時你不是要秀答案，只是要列出式子。我們要怎麼樣讓 Maxima 不要太自動就算出來呢？答案是加個 “'”號在前面，例如：

```
(%i2) 'integrate(%e^(-x^2),x,0,inf);
```
(%o2) $\displaystyle\int_0^{\infty} \frac{1}{e^{x^2}}\,dx$

3.2 kill 指令

有時我們設定了一堆變數，函數，後來又不想再用下去，可以用 `kill` 指令。而 `kill(all)` 更是把我們定義過的變數，函數全部刪除。看些例子就更加清楚：

(%i3) `f(x) := 3*x^2 + 5;`

(%o3) $f(x) := 3x^2 + 5$

(%i4) `f(x);`

(%o4) $3x^2 + 5$

(%i5) `kill(all)`

(%o5) $done$

(%i6) `f(x);`

(%o6) $f(x)$

3.3 ev的使用

我們可以把 Maxima 的 `ev` 指令想成一個獨立的環境。有點像在寫程式時的函式一樣, 並不會影響到其他的運作。第一種 `ev` 的應用是把我們設成不要執行的指令執行:

(%i7) `f:'integrate(x^2,x)`

(%o7) $\int x^2\, dx$

(%i8) `ev(f,integrate)`

(%o8) $\frac{x^3}{3}$

另一個很有用的使用方式是, 我們有個式子, 比方說:

(%i9) `f:  a*x^2 + b*x + c;`

(%o9) $ax^2 + bx + c$

假設我們想令一個式子是 $a=1, b=-2, c=-8$ 的情況, 我們當然可以先令各個變數是這樣, 們問題是這麼一來, f 也永遠是 x^2-2x-8, a, b, c 這三個變數也不再是「符號」, 而是有值的。為了避免這個問題, 我們可以用 ev 指令, 在下了這個指令後, 我們可以發現, 並沒有變動到原來 a, b, c 或是 f:

```
(%i10) g:  ev(f, a=1, b=-2, c=-8);
```
(%o10) x^2-2x-8
```
(%i11) a;
```
(%o11) a

第 4 節

線性代數相關指令

這節我們正式介紹線性代數相關，也就是矩陣相關的指令。

4.1 矩陣及向量

我們先來看矩陣和向量的定義方式。前面說過，在 Maxima 裡，所謂設定一個變數的值，只不過是給某個數字或矩陣等等一個名稱。我們這裡就舉應用在矩陣和向量時的情況：

```
(%i1) A:matrix([1,2,3],[-2,8,3],[1,4,9]);
```
(%o1) $\begin{pmatrix} 1 & 2 & 3 \\ -2 & 8 & 3 \\ 1 & 4 & 9 \end{pmatrix}$
```
(%i2) v:  [2,3,5];
```
(%o2) $[2,3,5]$

我們可以看出，要定義一個矩陣，就是把矩陣一列列的輸入；定義一個向

量，其實和我們用手寫向量出來也差不多。不過，問題是我們在線性代數常常要把向量寫成「行向量」，而非如上的「列向量」表示方式。我們可以用下面兩種不同的方式達成：

```
(%i3) v:  transpose([2,3,5]);
```

(%o3) $\begin{pmatrix} 2 \\ 3 \\ 5 \end{pmatrix}$

```
(%i4) v:  matrix([2],[3],[5]];
```

(%o4) $\begin{pmatrix} 2 \\ 3 \\ 5 \end{pmatrix}$

其實向量應該是一個一列或一行的矩陣, 但是 Maxima 提供了簡單定義列向量的方法。這裡要強調一點, 一般來說因為矩陣乘法的關係, 我們寫成列向量和行向量差別很大。不過 Maxima 其實不太在意這點: 它可以聰明地發現你要做的事, 並且正確得計算出來！簡單的說, **一般而言, 我們不需要麻煩得定義行向量, 用列向量即可**。

4.2 矩陣的表示和截取

這節我們討論矩陣的抽象表示和取出一個矩陣行，列，甚至 entry 的方法。這在很多理論和計算的嘗試會用到。

Maxima 是一個 CAS 系統，所以我們可以完全用符號去定義一個矩陣，比方說：

```
(%i5) A: matrix([a[1,1],a[1,2]],[a[2,1],a[2,2]]);
```

(%o5) $\begin{pmatrix} a_{1,1} & a_{1,2} \\ a_{2,1} & a_{2,2} \end{pmatrix}$

你也可以做完全抽象的代數計算：

```
(%i6) c*A;
```

(%o6) $\begin{pmatrix} a_{1,1}\,c & a_{1,2}\,c \\ a_{2,1}\,c & a_{2,2}\,c \end{pmatrix}$

如此一來，我們要試著導出一些定理就非常方便！

現在，我們重新把 A 定義成一個實數矩陣，再看看怎麼樣找出 A 的某一列，某一行，或某個 entry。

(%i7) `A: matrix([1,2,3],[-2,8,3],[1,4,9]);`

(%o7) $\begin{pmatrix} 1 & 2 & 3 \\ -2 & 8 & 3 \\ 1 & 4 & 9 \end{pmatrix}$

(%i8) `row(A,1);`

(%o8) $\begin{pmatrix} 1 & 2 & 3 \end{pmatrix}$

(%i9) `col(A,2);`

(%o9) $\begin{pmatrix} 2 \\ 8 \\ 4 \end{pmatrix}$

(%i10) `A[2,3];`

(%o10) 3

4.3 矩陣向量之四則運算

我們要做矩陣加法、減法、乘法非常直覺而容易。乘法用的運算元是 “.”。我們假設有了前面矩陣 A 和向量 v 的定義，來看以下的例子：

(%i11) `A.v;`

(%o11) $\begin{pmatrix} 23 \\ 35 \\ 59 \end{pmatrix}$

你也可以定義非向量的矩陣試試矩陣的乘法。比方說，兩個矩陣 A, B 的乘積是 `A.B`，要注意 `A*B` 並不會得到矩陣相乘的結果！到底 `A*B` 是什麼意思，大

家不妨自己試試，看可不可以找出其中的意義。

向量內積的做法和你想的一樣：

```
(%i12) w:  [2,3,5];
```

(%o12) $[2, 3, 5]$

```
(%i13) w.w;
```

(%o13) 38

你可能發現了一個問題，那就是我們上面內積的例子是用列向量。那行向量可以嗎？可以的！Maxima會聰明的知道你想做什麼，不信可以試試看。

矩陣和向量的純量乘法是用平常的 “*” 號：

```
(%i14) 2*A;
```

(%o14) $$\begin{pmatrix} 2 & 4 & 6 \\ -4 & 16 & 6 \\ 2 & 8 & 18 \end{pmatrix}$$

現在我們來看一下有可能會產生誤會的地方。假設我們現在要算 $A \cdot A$，你可能會想是 `A^2`，結果並不正確！其實`A^2` 是把 A 的每一個 entry 都平方。正確計算 $A \cdot A$ 要用 `A^^2`

4.4 矩陣相關函數

我們要計算矩陣的行列式值，求轉置矩陣, 矩陣的秩等等的基本運算， Maxima當然也都有（A還是我們之前定義的矩陣）：

```
(%i15) transpose(A);
```

(%o15) $$\begin{pmatrix} 1 & -2 & 1 \\ 2 & 8 & 4 \\ 3 & 3 & 9 \end{pmatrix}$$

```
(%i16) determinant(A);
```

(%o16) 54

(%i17) `rank(A);`

(%o17) 3

我們當然也可以手動計算行列式值。但這時需要知道矩陣第 i,j 這個位置的子式 (minor), 也就是 A 矩陣去掉第 i 列, 第 j 行所成的矩陣, 這指令叫 `minor`:

(%i18) `minor(A,1,1);`

(%o18) $\begin{pmatrix} 8 & 3 \\ 4 & 9 \end{pmatrix}$

矩陣的餘因子 (cofactor) 在 Maxima 中並沒有定義, 好在我們自己可以很容易定一個 `cofactor` 函數:

(%i19) `cofactor(M,i,j):=(-1)^(i+j)*determinant(minor(M,i,j));`

(%o19) $cofactor(M,i,j) := (-1)^{i+j} DETERMINANT(MINOR(M,i,j))$

(%i20) `cofactor(A,1,1);`

(%o20) 60

我們在計算反矩陣等會用到的古典伴隨矩陣 (classical adjoint matrix) 也很容易算出來:

(%i21) `adjoint(A);`

(%o21) $\begin{pmatrix} 60 & -6 & -18 \\ 21 & 6 & -9 \\ -16 & -2 & 12 \end{pmatrix}$

說到反矩陣，要用 Maxima 求出來也是易如反掌:

(%i22) `invert(A);`

(%o22) $$\begin{pmatrix} \frac{10}{9} & -\frac{1}{9} & -\frac{1}{3} \\ \frac{7}{18} & \frac{1}{9} & -\frac{1}{6} \\ -\frac{8}{27} & -\frac{1}{27} & \frac{2}{9} \end{pmatrix}$$

或是你也可以用前面的方式求反矩陣:

```
(%i23) A^^(-1);
```

(%o23) $$\begin{pmatrix} \frac{10}{9} & -\frac{1}{9} & -\frac{1}{3} \\ \frac{7}{18} & \frac{1}{9} & -\frac{1}{6} \\ -\frac{8}{27} & -\frac{1}{27} & \frac{2}{9} \end{pmatrix}$$

在解線性方程組常用到的梯形矩陣也是容易得很：

```
(%i24) echelon(A);
```

(%o24) $$\begin{pmatrix} 1 & 2 & 3 \\ 0 & 1 & \frac{3}{4} \\ 0 & 0 & 1 \end{pmatrix}$$

4.5 使用模組

用了 Maxima 一陣子，你可能會預期它該會的都會。比方說求一個矩陣的 trace，這應該夠容易了吧？

事情並不是那麼簡單。Maxima 本身是「不會」算 trace 的！當然我們可以自己寫個小程式，不過先別急。我們可以使用使用適當的模組來做這件事。

所謂模組就是一段小程式，通常是增加一些指令，供你使用。你也許會覺得奇怪，那為什麼 Maxima 不一開始就把這些模組都加進來？那是因為如此一來太佔用記憶體，也許很多對某些人重要的指令你永遠也不用去用！

我們要算一個矩陣的 trace，要使用 `nchrpl` 這個模組，這個模組提供了

mattrace 指令去計算 trace。

使用的方法如下，先以

```
(%i25) load("ncharl");
```

讀入 ncharl 模組，接著就可以使用這個模組提供的指令：

```
(%i26) A: matrix([1,2,3],[2,2,1],[3,3,1]);
```

(%o26) $\begin{pmatrix} 1 & 2 & 3 \\ 2 & 2 & 1 \\ 3 & 3 & 1 \end{pmatrix}$

```
(%i27) mattrace(A);
```

(%o27) 4

第 5 節 線性代數應用實例

5.1 特徵值和特徵向量

我們這裡討論線性代數很重要的特徵值相關的計算。我們定義一個矩陣 A, 計算特徵值和特徵向量時我們都以這個矩陣為主要討論對象:

```
(%i1) A: matrix([4,0,1],[2,3,2],[1,0,4]);
```

(%o1) $\begin{pmatrix} 4 & 0 & 1 \\ 2 & 3 & 2 \\ 1 & 0 & 4 \end{pmatrix}$

我們計算一下特徵值：

```
(%i2) eigenvalues(A);
```
(%o2) $[[5,3],[1,2]]$

怎麼樣，很方便吧...等等，特徵值怎麼會出來兩個向量呢！？原來，真正的特徵值是放在結果的第一個 list 當中，也就是 5 和 3。那第二個 list 代表什麼呢？代表的就是每個特徵值的幾何重數, 也就是每個特徵值對應的特徵向量空間之維度。換言之，這是比較完整的特徵值資訊！

我們也可以用 `eigenvectors` 計算特徵向量。事實上，`eigenvectors` 也會把特徵值列出來，所以是包含前面 `eigenvalues` 功能的指令。不過如果我們一開始就介紹 `eigenvectors`，看到那有點複雜的結果大家可能會昏倒。現在已經會了 `eigenvalues`，大概就沒問題了：

```
(%i3) eigenvectors(A);
```
(%o3) $[[[5,3],[1,2]],[1,2,1],[1,0,-1],[0,1,0]]$

第一部份和 `eigenvectors` 輸出一樣，就是說我們有 5 有和 3 兩個特徵值，其 mutiplicities 分別是 1 和 2。因此，對於 5 應該要有一個對應的特徵向量，即 $[1,2,1]$，對於 3會有兩個，分別是接下來的 $[1,0,-1]$ 和 $[0,1,0]$。這些向量會生成相對應特徵值的向量空間。

5.2 手動特徵值的計算

上一節介紹 Maxima 內建特徵值計算，並不一定每個人都喜歡。比方說顯示的方式比較特別，另外就是不是一步一步算的，心裡有時也有不踏實的感覺。因此，我們這裡介紹一下如何用 Maxima 一步一步的把特徵值求出來。

我們再用一次上一節的例子：

```
(%i4) A: matrix([4,0,1],[2,3,2],[1,0,4]);
```

(%o4) $\begin{pmatrix} 4 & 0 & 1 \\ 2 & 3 & 2 \\ 1 & 0 & 4 \end{pmatrix}$

我們先求特徵多項式，也就是 $A - tI$ 的行列式值：

```
(%i5) f:  charpoly(A,t);
```

(%o5) $f : t + (3 - t)(4 - t)^2 - 3$

如果想要看到比較漂亮的式子, 可以將 f 展開:

```
(%i6) expand(f);
```

(%o6) $-t^3 + 11t^2 - 39t + 45$

我們還可以將 f 做因式分解, 這樣就可以清楚看到 A 有幾個特徵值, 和各特徵值的代數重數:

```
(%i7) factor(f);
```

(%o7) $-(t - 5)(t - 3)^2$

這樣我們就求得 A 的特徵值是 5 和 3。

另一個解法是，我們可以求 $f = 0$ 的零根。做法是使用 `solve` 指令：

```
(%i8) solve(f=0, t);
```

(%o8) $[t = 5, t = 3]$

當然，我們算法正確，應該是得到和前面一樣的結果。

5.3 解線性方程組

線性代數的核心問題，就是解線性方程組。解線性方程組一樣可以用上一節介紹的 `solve` 指令來解。我們來看一個簡單的例子，並且用 Maxima 來解。

我們考慮下面的線性方程組：

$$
\begin{aligned}
x + 2y + 3z &= 6 \\
2x - 3y + 2z &= 14 \\
3x + y - z &= -2
\end{aligned}
$$

我們一樣可以用前面用過的 `solve` 指令來解：

```
(%i9) eq1:  x + 2*y + 3*z = 6;
```
(%o9) $x + 2y + 3z = 6$
```
(%i10) eq2:  2*x - 3*y + 2*z = 14;
```
(%o10) $2x - 3y + 2z = 14$
```
(%i11) eq3:  3*x + y - z = -2;
```
(%o11) $3x + y - z = -2$
```
(%i12) solve([eq1, eq2, eq3],[x,y,z]);
```
(%o12) $[[x = 1, y = -2, z = 3]]$

5.4 手動求特徵向量空間的基底

我們在前面介紹過, 使用 `eigenvectors` 指令就可以求出特徵向量空間的一組基底。我們再用一次前面的矩陣:

```
(%i13) A: matrix([4,0,1],[2,3,2],[1,0,4]);
```
(%o13) $\begin{pmatrix} 4 & 0 & 1 \\ 2 & 3 & 2 \\ 1 & 0 & 4 \end{pmatrix}$

我們已經求出 A 的特徵值是 5 和 3, 我們這裡用特徵值 3 做範例, 看看怎麼樣能求出對應的特徵向量。我們現在要求的就是什麼樣的向量 v, 會滿足 $(A-3I)v=0$。這裡我們可以用 `ident` 指令可以很容易造出 $n\times n$ 的單位矩陣。以下我們就把大略的設定做好:

(%i14) `I: ident(3);`

(%o14) $\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$

(%i15) `v:[x,y,z];`

(%o15) $[x,y,z]$

(%i16) `u:  (A-3*I).v;`

(%o16) $\begin{pmatrix} z+x \\ 2z+2x \\ z+x \end{pmatrix}$

我們現在就是要看什麼樣的 x,y,z 會讓 u 是零向量。這個例子其實用手解也很容易, 但是我們給 Maxima 一個機會。我們要做的就是解一個線性方程組:

(%i17) `eq1:  u[1,1]=0;`

(%o17) $z+x=0$

(%i18) `eq2:  u[2,1]=0;`

(%o18) $2z+2x=0$

(%i19) `eq3:  u[3,1]=0;`

(%o19) $z+x=0$

(%i20) `solve([eq1,eq2,eq3],[x,y,z]);`

(%o20) $[[x=-\%r1, y=\%r2, z=\%r1]]$

這看來有點可怕的 `%r1` 和 `%r2` 是什麼呢? 原來這只是表示兩個參數, 換成我們一般的寫法, 我們可能會寫成 $x=-t, y=s, z=t$。至此, 我們已找到特徵向量的一般表示式, 如果要找到一組基底也很容易, 我們先令 `ans` 代表前面解出的式子, 再把 $(\%r1,\%r2)$ 代入 $(1,0)$, $(0,1)$ 即可:

```
(%i21) ans:  %;
```
(%o21) $[[x = -\%r1, y = \%r2, z = \%r1]]$
```
(%i22) ev(ans, %r1=1, %r2=0);
```
(%o22) $[[x = -1, y = 0, z = 1]]$
```
(%i23) ev(ans, %r1=0, %r2=1);
```
(%o23) $[[x = 0, y = 1, z = 0]]$

我們可能會希望把結果設成兩個向量 v_1, v_2, 方便以後使用。我們可以再用 `ev` 來做到這樣的事:

```
(%i24) v1:  ev([x,y,z], %o22)
```
(%o24) $[-1, 0, 1]$
```
(%i25) v2:  ev([x,y,z], %o23)
```
(%o25) $[0, 1, 0]$

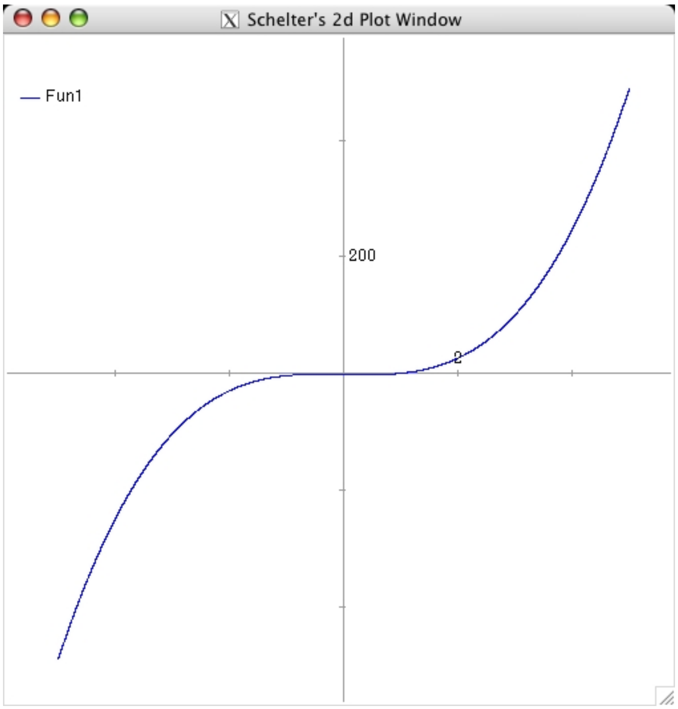
第 6 節

Maxima 的繪圖功能

6.1 二維繪圖

Maxima 二維繪圖的指令是用 `plot2d`。比方說，我們要畫 $4x^3 - 2x - 2$ 這個函數，設定 x 軸的範圍是從 -5 到 5，就下這個指令：

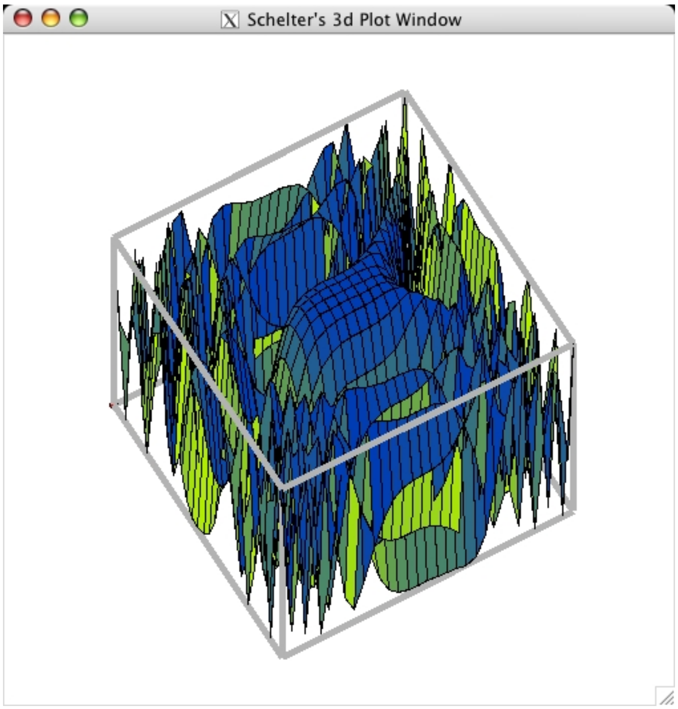
```
(%i1) plot2d([4*x^3-2*x-2],[x,-5,5]);
```



6.2 三維繪圖

三維繪圖也一樣容易，只要改用 plot3d 的指令即可：

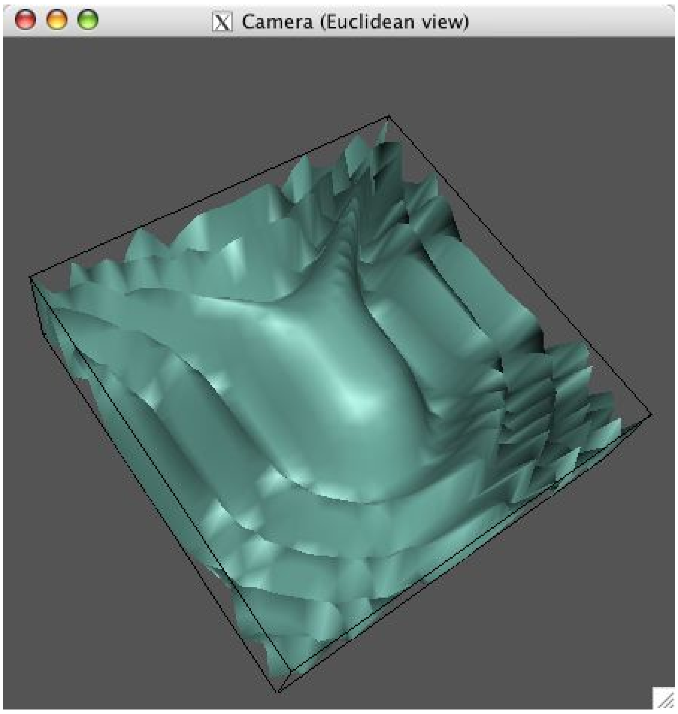
```
(%i2) plot3d(cos(-x^2+y^3/4),[x,-4,4], [y,-4,4]);
```



Geomview 是一個 UNIX 的軟體，Maxima 可以運用 Geomview 做出非常漂亮的 3D 圖形。我們來看上個例子以 Geomview 輸出的結果。

```
(%i3) plot3d(cos(-x^2+y^3/4),[x,-4,4], [y,-4,4], [plot_format,geomview]);
```

Geomview 不但可以畫出漂亮 3D 圖形，更重要的是它可以彌補 Maxima 的一些缺點。比方說，Maxima 本身的 3D 繪圖不可以同時顯示兩個或兩個以上函數圖形（2D可以），但利用 Geomview，這樣的繪圖變成可能。



6.3 點繪圖

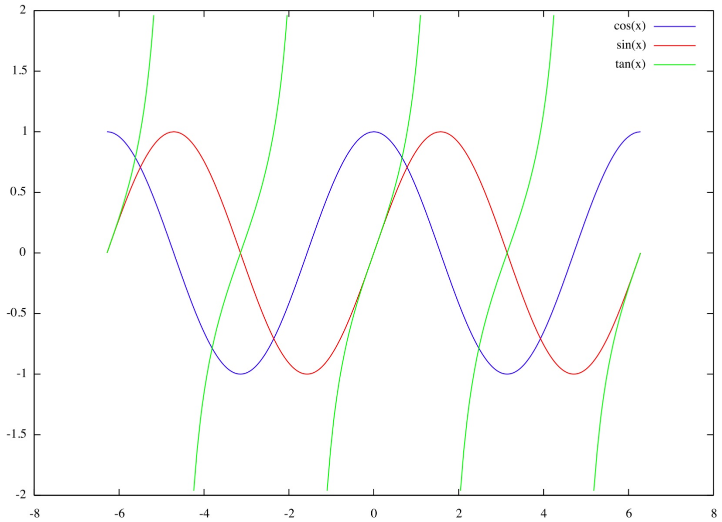
有很多繪圖的應用，就只需要畫出點，或是用一些點來描述一些函數。這事實上比畫函數還簡單，但是 Maxima 直到 5.9.2 版才有這樣的功能。詳情請參考 (5.9.2 之後的) 使用手冊。

6.4 多個函數的繪圖

如果要比較幾個函數，要如何下指令呢？我們來看個例子就明白了：

```
(%i4) plot2d([cos(x), sin(x), tan(x)], [x, -2*%pi, 2*%pi], [y,-2,2])$
```

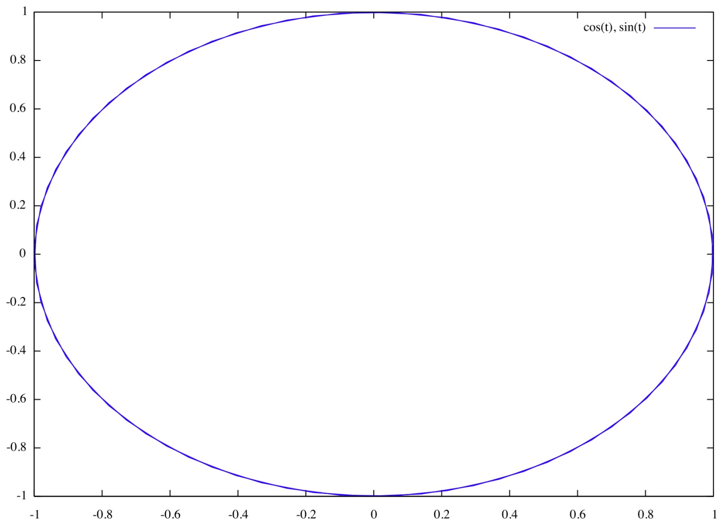
這個例子會同時畫出 $\cos(x), \sin(x)$ 和 $\tan(x)$ 的圖形。



6.5 參數式繪圖

我們僅簡單舉一參數式繪圖之例子, 詳情請參考 Maxima 使用手冊。

```
(%i5) plot2d([parametric, cos(t), sin(t), [t, -2*%pi, 2*%pi], [nicks,
80]]);
```



第 A 節

Maxima 的安裝

在 Maxima 的官方網站有不同版本的 Maxima 供各平台使用：

`http://maxima.sourceforge.net/`

不過，不同平台可能有一些不同的選擇。我概略說明一下我建議的安裝方式。不管用 Windows, Mac, 或是 Linux，我都推薦使用 TeXmacs 這個文書處理軟體當界面，因為這樣可以顯示最漂亮的數學符號。

A.1 Windows

Windows 至少有三種可以執行 Maxima 的方式，不管哪一種，都要先裝 xMaxima。首先就是在官方網站下載 Maxima Windows 版。安裝也很容易，下載後點兩下就可以自動安裝。

xMaxima 的缺點是純文字顯示，不能顯示漂亮的數學符號。使用 PC 的同學，當然可以試著安裝 Linux，採用下面介紹的方式使用 Maxima。如果還沒確定，或不想花那麼多時間安裝 Linux，可以先試用有 Maxima 的 LiveCD。Linux 的 LiveCD 是可以開機的 CD，你只要放進你的電腦，用光碟開機，就可使用，不用灌 Linux。

長庚大學黃朝錦教授提供了有 TeXmacs （見後 Linux 的說明）及 Maxima 的 LiveCD，你可在下面 download。

`ftp://math.cgu.edu.tw/pub/KNOPPIX/`

請選擇 TeXmacs 的 `.iso` 檔，再燒成光碟即可。注意有一般光碟和 DVD 版，看自己的需要下載。

Windows 還可以裝 wxMaxima，這個界面比 xMaxima 漂亮，不過還不及接

下來要介紹，採用 WinTeXmacs 的界面，所以我不詳細介紹。

Windows （或其他平台），讓 Maxima 看來最漂亮的大概就是用 WinTeXmacs。

A.2 Linux

不同的 Linux 都有不同軟體管理程式，像 Maxima 大概所有管理程式都有提供，所以我不詳細說明如何安裝，只列出建議安裝的套件：

Maxima Maxima 主程式。

TeXmacs 一個可打漂亮數學式子的編輯器，提供漂亮的 Maxima 介面。

Geomview 配合 Maxima 可畫出高級 3D 圖形。

使用時，是執行 TeXmacs，在裡面執行 Maxima 的 session 即可。這是我推薦的漂亮版 Maxima。

A.3 Mac OS X

Mac OS X 是一個 UNIX 系統，所以需要的程式和 Linux 一樣。首先你先要安裝 Apple 的 X11 軟體。這是因為 UNIX 上用的 X-Windows 系統當然和 Mac OS X 的 aqua 視窗系統不同，UNIX 軟體大多只能用 X-Windows 顯示。

在安裝 TeXmacs 之前，你必需要有完整的 LaTeX 系統。我強烈推薦用 i-installer 安裝：

`http://www.rna.nl/tex.html`

如果不知怎麼做，可參考我的文章：

`http://homepage.mac.com/yenlung/WebWiki/LaTeXonMac.html`

接著，使用 Fink 去安裝 Maxima, TeXmacs, Geomview:

`http://www.rna.nl/tex.html`

要注意的是 Geomview 只有 unstable 版。