Matlab 入門篇

2017/7/23

林崇聖

Matlab-介紹

Matlab 為一種高階的程式語言，可用於資料分析、資料視覺化、數值模擬與計算等，對於科學的理論分析有很大幫助。

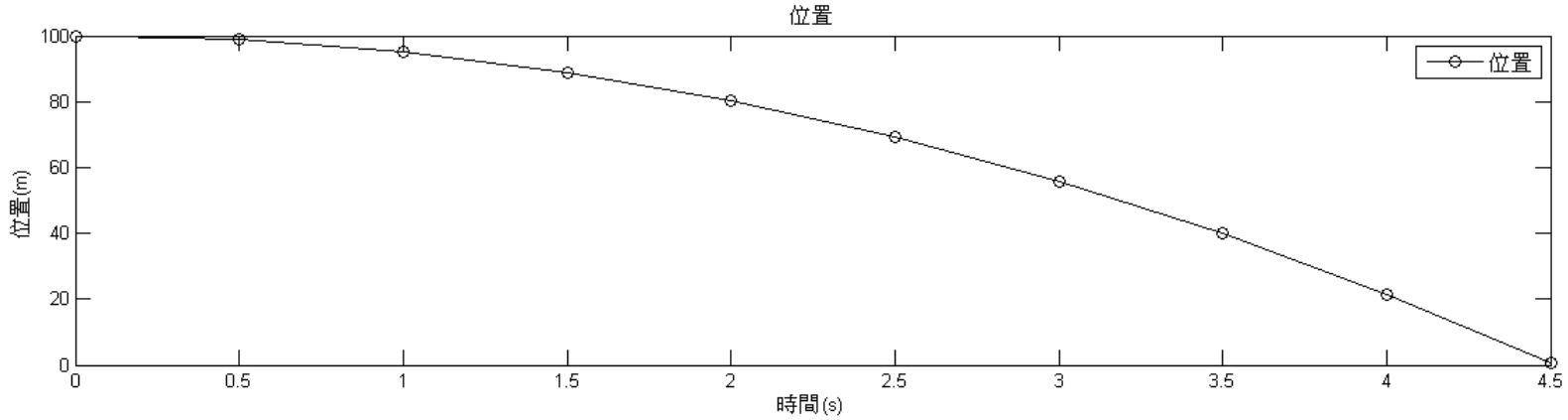
例如：

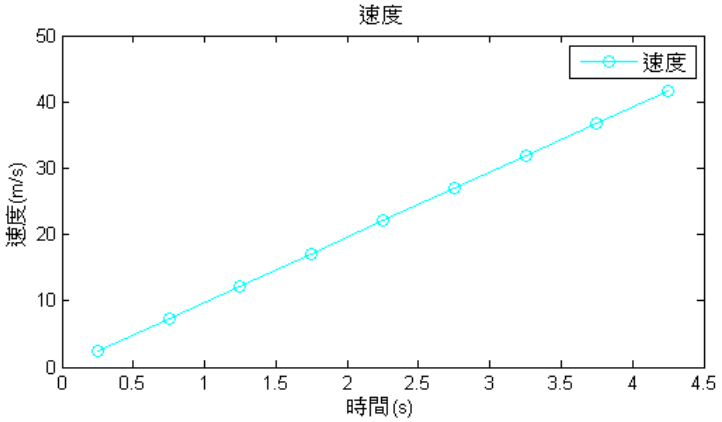
現在有量測物體隨時間的位置資料，此時需要得到該物體的速度、加速度，即可以利用此軟體將位置的資料讀取後，進行速度、加速的計算，接著將時間對位置、速度、加速度作圖，可以快速的得到分析的結果並且以圖形化的方式呈現結果。

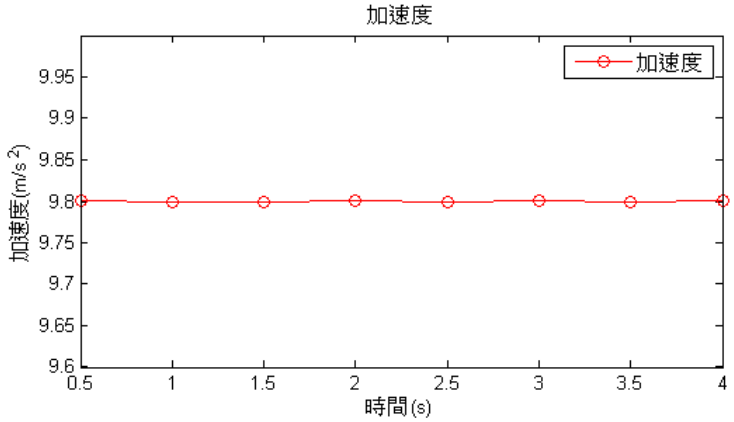
100.0000 98.7750 95.1000 88.9750 80.4000 69.3750 55.9000 39.9750 21.6000 0.7750

量測的位置座標

計算速度、加速度，並畫圖顯示





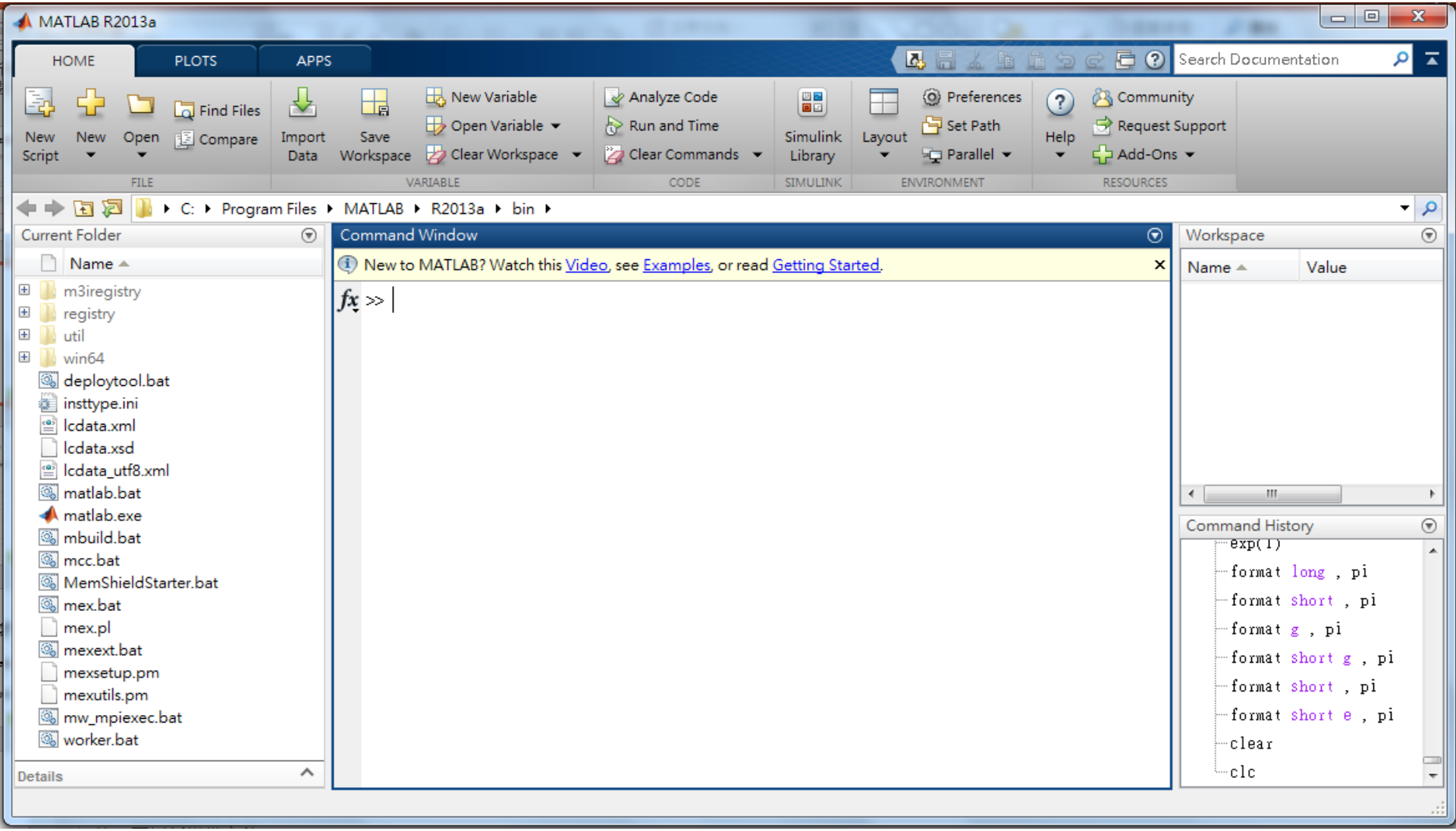


開啟程式

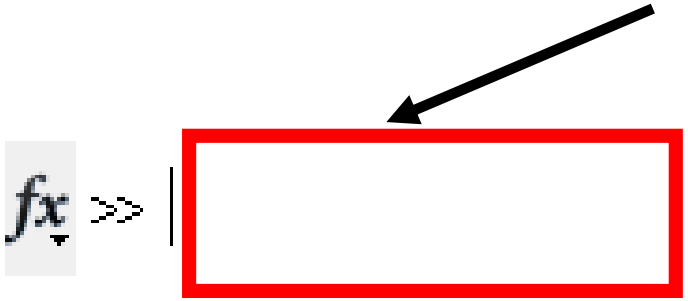
找尋此圖案的程式開啟

OR

搜尋程式：
matlab


MATLAB R2013a
HOME
PLOTS
APPS
Search Documentation
New Script
New
Open
Find Files
Compare
Import Data
Save Workspace
New Variable
Open Variable
Clear Workspace
Analyze Code
Run and Time
Clear Commands
Simulink Library
Layout
Preferences
Set Path
Parallel
Help
Community
Request Support
Add-Ons
FILE
VARIABLE
CODE
SIMULINK
ENVIRONMENT
RESOURCES
C: ▸ Program Files ▸ MATLAB ▸ R2013a ▸ bin ▸
Current Folder
Name
m3iregistry
registry
util
win64
deploytool.bat
insttype.ini
lcdata.xml
lcdata.xsd
lcdata_utf8.xml
matlab.bat
matlab.exe
mbuild.bat
mcc.bat
MemShieldStarter.bat
mex.bat
mex.pl
mexext.bat
mexsetup.pm
mexutils.pm
mw_mpiexec.bat
worker.bat
Details
Command Window
New to MATLAB? Watch this Video, see Examples, or read Getting Started.
fx >>
Workspace
Name
Value
Command History
exp(1)
format long , pi
format short , pi
format g , pi
format short g , pi
format short , pi
format short e , pi
clear
clc

Matlab-運算符號



可輸入運算方程式
或是函數等功能
（後面將此處輸入的內容稱為程式碼）

數學運算：

遵守四則運算

4/2+3

4/(2+3)

運算結果不同

*刮號皆使用小刮號

```
>> 1+1
```

輸入方程式後
按下Enter

```
ans =

     2
```

將會顯示結果

Matlab-數學運算元

Symbol	Role	More Information
+	Addition	plus
+	Unary plus	uplus
-	Subtraction	minus
-	Unary minus	uminus
.*	Element-wise multiplication	times
*	Matrix multiplication	mtimes
./	Element-wise right division	rdivide
/	Matrix right division	mrdivide
.\	Element-wise left division	ldivide
\	Matrix left divison (also known as *backslash*)	mldivide
.^	Element-wise power	power
^	Matrix power	mpower
.'	Transpose	transpose
'	Complex conjugate transpose	ctranspose

https://www.mathworks.com/help/matlab/matlab_prog/matlab-operators-and-special-characters.html

Matlab-運算符號

邏輯比較時：
正確（True)　　傳回1
不正確（False）傳回0

關係運算：

```
>> 1==1

ans =

     1
```

```
>> 1==2

ans =

     0
```

Matlab-運算符號

邏輯運算元：

&AND

A AND B	A=0	A=1
B=0	0	0
B=1	0	1

|OR

A OR B	A=0	A=1
B=0	0	1
B=1	1	1

~NOT

A	A=0	A=1
NOT A	1	0

邏輯運算：

```
>> 1&0
ans =
     0

>> 1|0
ans =
     1

>> ~1
ans =
     0
```

http://1.bp.blogspot.com/-L7OndVqwnUM/VaFCpQfQ5dI/AAAAAAAACC0/DTbpx9a5G9k/s1600/bitwiseops%2Bcopy.gif

Matlab-關係與邏輯運算元

Relational Operators

Symbol	Role	More Information
==	Equal to	eq
~=	Not equal to	ne
>	Greater than	gt
>=	Greater than or equal to	ge
<	Less than	lt
<=	Less than or equal to	le

Logical Operators

Symbol	Role	More Information
&	Logical AND	and
|	Logical OR	or
&&	Logical AND (with short-circuiting)	Logical Operators: Short-Circuit && ||
||	Logical OR (with short-circuiting)	
~	Logical NOT	not

https://www.mathworks.com/help/matlab/matlab_prog/matlab-operators-and-special-characters.html

常用運算

科學符號

10^3 = 1E3 = 1e3 = 1000

三角函數

π： pi = 3.1415926535……

cosd(90) = cos(pi/2)

顯示格式

>> format style

```
>> format long , pi

ans =

   3.14159265358979

>> format short , pi

ans =

    3.1416
```

```
>> format short e , pi

ans =

   3.1416e+00
```

Style	Result	Example
`short (default)`	Short, fixed-decimal format with 4 digits after the decimal point.	`3.1416`
`long`	Long, fixed-decimal format with 15 digits after the decimal point for `double` values, and 7 digits after the decimal point for `single` values.	`3.141592653589793`
`shortE`	Short scientific notation with 4 digits after the decimal point.	`3.1416e+00`
`longE`	Long scientific notation with 15 digits after the decimal point for `double` values, and 7 digits after the decimal point for `single` values.	`3.141592653589793e+00`
`shortG`	Short, fixed-decimal format or scientific notation, whichever is more compact, with a total of 5 digits.	`3.1416`
`longG`	Long, fixed-decimal format or scientific notation, whichever is more compact, with a total of 15 digits for `double` values, and 7 digits for `single` values.	`3.14159265358979`
`shortEng`	Short engineering notation (exponent is a multiple of 3) with 4 digits after the decimal point.	`3.1416e+000`
`longEng`	Long engineering notation (exponent is a multiple of 3) with 15 significant digits.	`3.14159265358979e+000`
`+`	Positive/Negative format with +, -, and blank characters displayed for positive, negative, and zero elements.	`+`
`bank`	Currency format with 2 digits after the decimal point.	`3.14`
`hex`	Hexadecimal representation of a binary double-precision number.	`400921fb54442d18`
`rat`	Ratio of small integers.	`355/113`

https://www.mathworks.com/help/matlab/ref/format.html

符號使用

, 若在一行中想要輸入兩段程式碼，則可以使用逗號隔開，如 1+1 , 2+2

; 若執行後不想顯示結果則可以在程式碼最後加上分號，如 1+1;

… 若一行內無法結束程式碼時，接在符號後面可以達成分行的動作，如

1+…

1

符號使用

```
>> 1+1,2+2

ans =

     2

ans =

     4
```

```
>> 1+1;
>>
```

```
>> 1+...
1

ans =

     2
```

函數使用

函數為特定名稱的指令，例如輸入mod(3,2)
將會得到3÷2的餘數1

```
>> mod(3,2)

ans =

     1
```

函數查詢

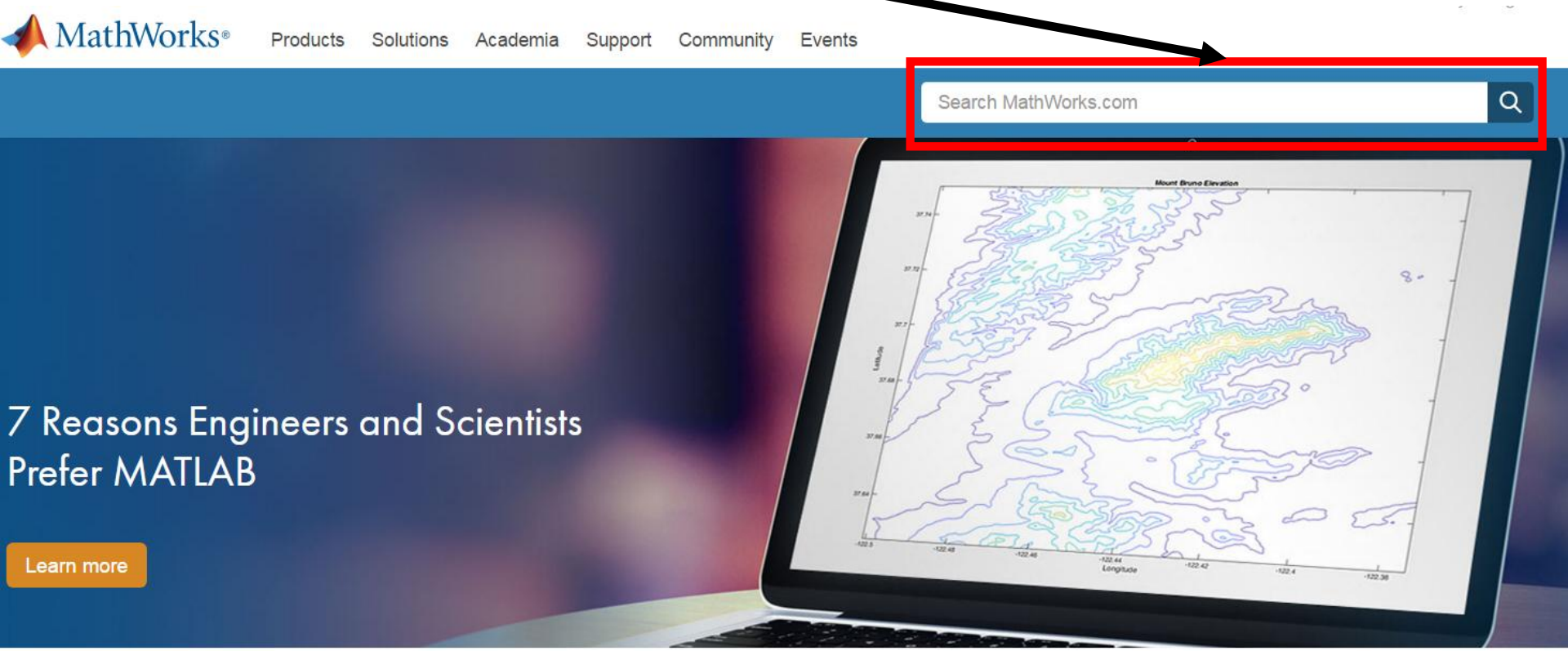
help 函數名

可以執行對此函數的說明，但是說明本身很簡短，因此有時不易瞭解該函數的使用，這時候建議到網路上或是官網查詢函數，將會有詳細的說明和範例

```
>> help mod
 mod    Modulus after division.
    mod(x,y) is x - n.*y where n = floor(x./y) if y ~= 0.  If y is not an
    integer and the quotient x./y is within roundoff error of an integer,
    then n is that integer.  The inputs x and y must be real arrays of the
    same size, or real scalars.

    The statement "x and y are congruent mod m" means mod(x,m) == mod(y,m).

    By convention:
       mod(x,0) is x.
       mod(x,x) is 0.
       mod(x,y), for x~=y and y~=0, has the same sign as y.
```

函數查詢

官網查詢：輸入函數名即可搜尋



https://www.mathworks.com/?s_tid=gn_logo

函數查詢

mod

R2017a

Remainder after division (modulo operation)

collapse all in page

Syntax

```
b = mod(a,m)
```

Description

example

`b = mod(a,m)` returns the remainder after division of `a` by `m`, where `a` is the dividend and `m` is the divisor. This function is often called the modulo operation, which can be expressed as `b = a - m.*floor(a./m)`. The `mod` function follows the convention that `mod(a,0)` returns `a`.

Examples

collapse all

Remainder After Division of Scalar

Compute 23 modulo 5.

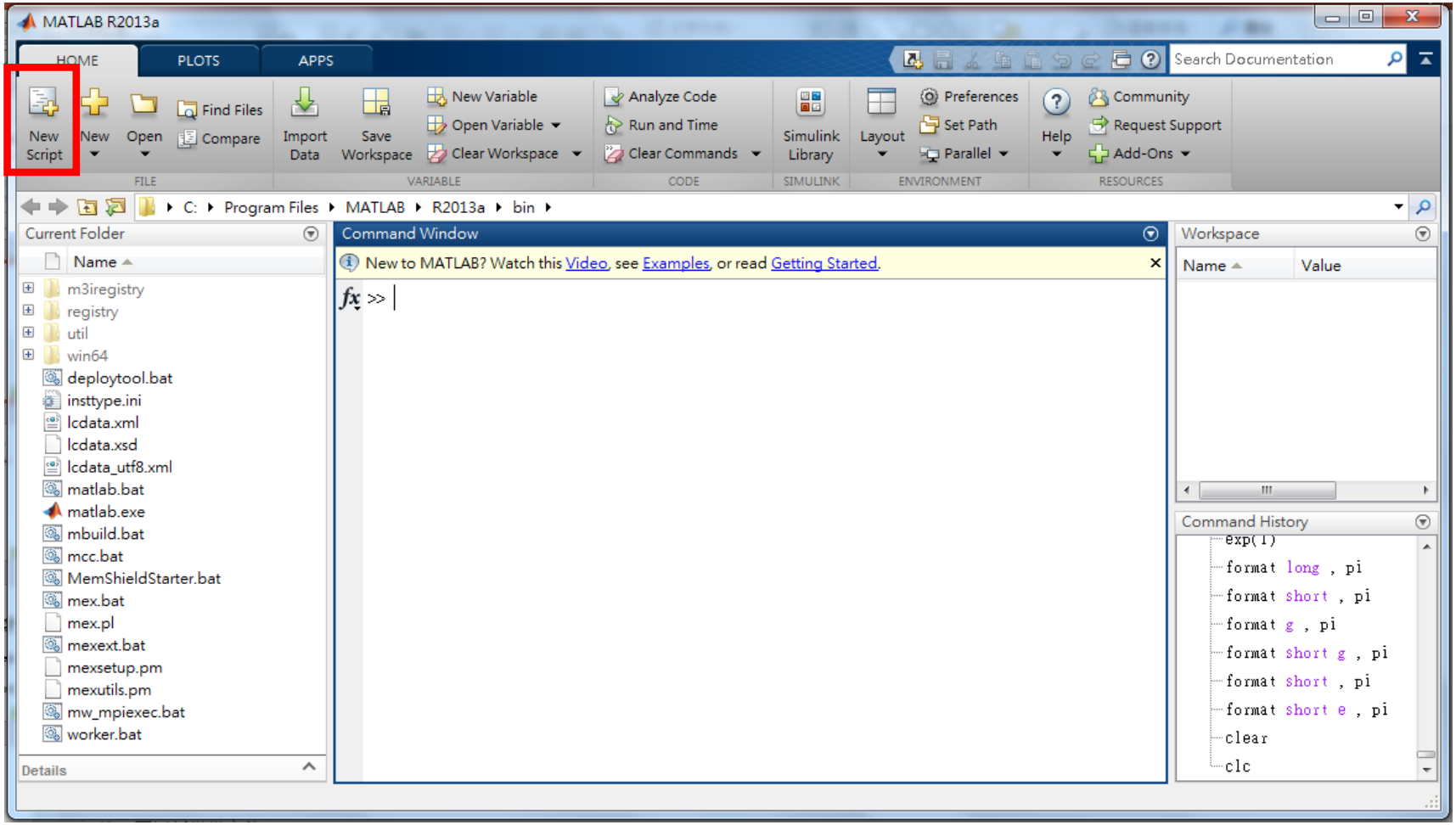
Try this Example

```
b = mod(23,5)
```

```
b = 3
```

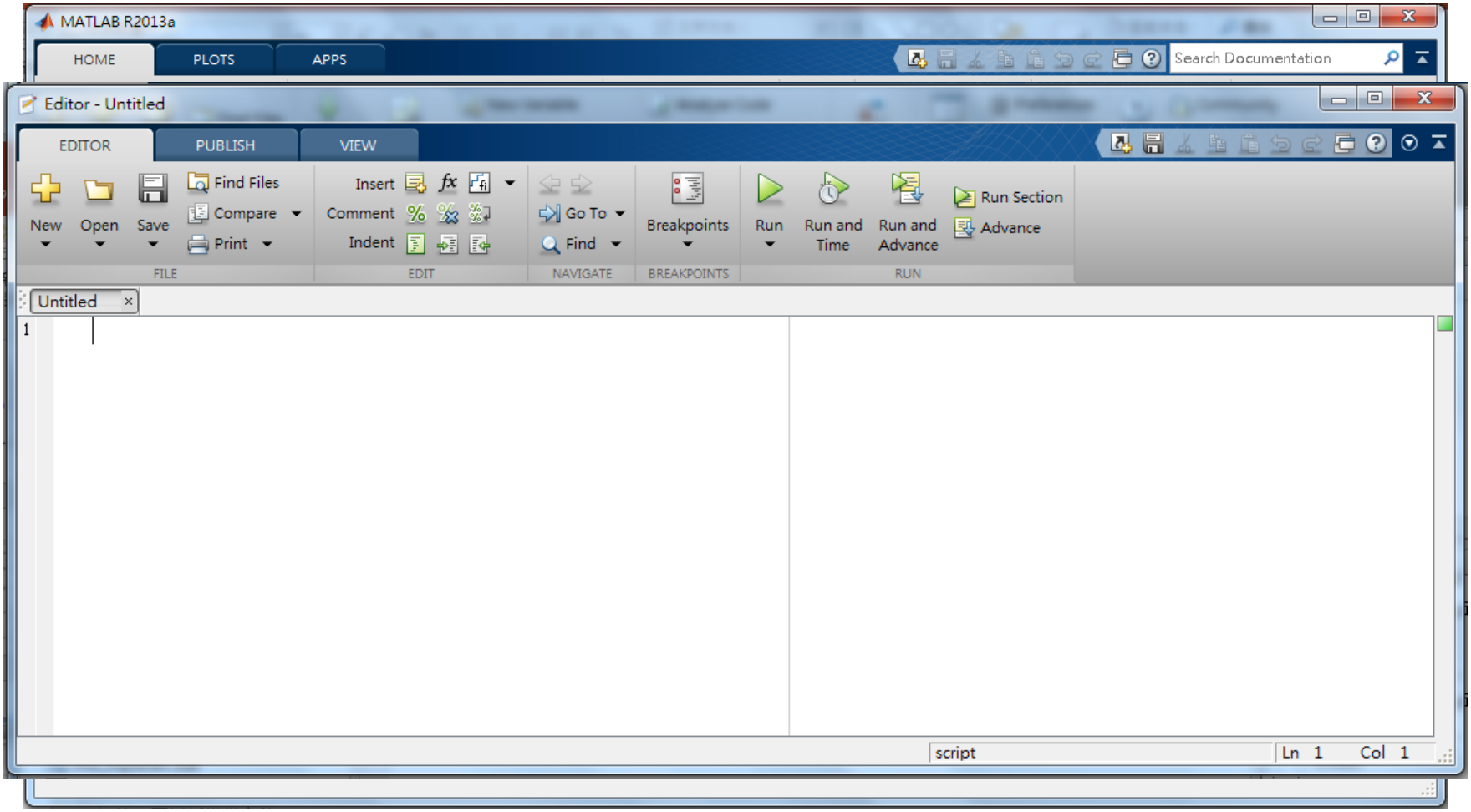
https://www.mathworks.com/help/matlab/ref/mod.html?s_tid=srchtitle

開啟 New Script

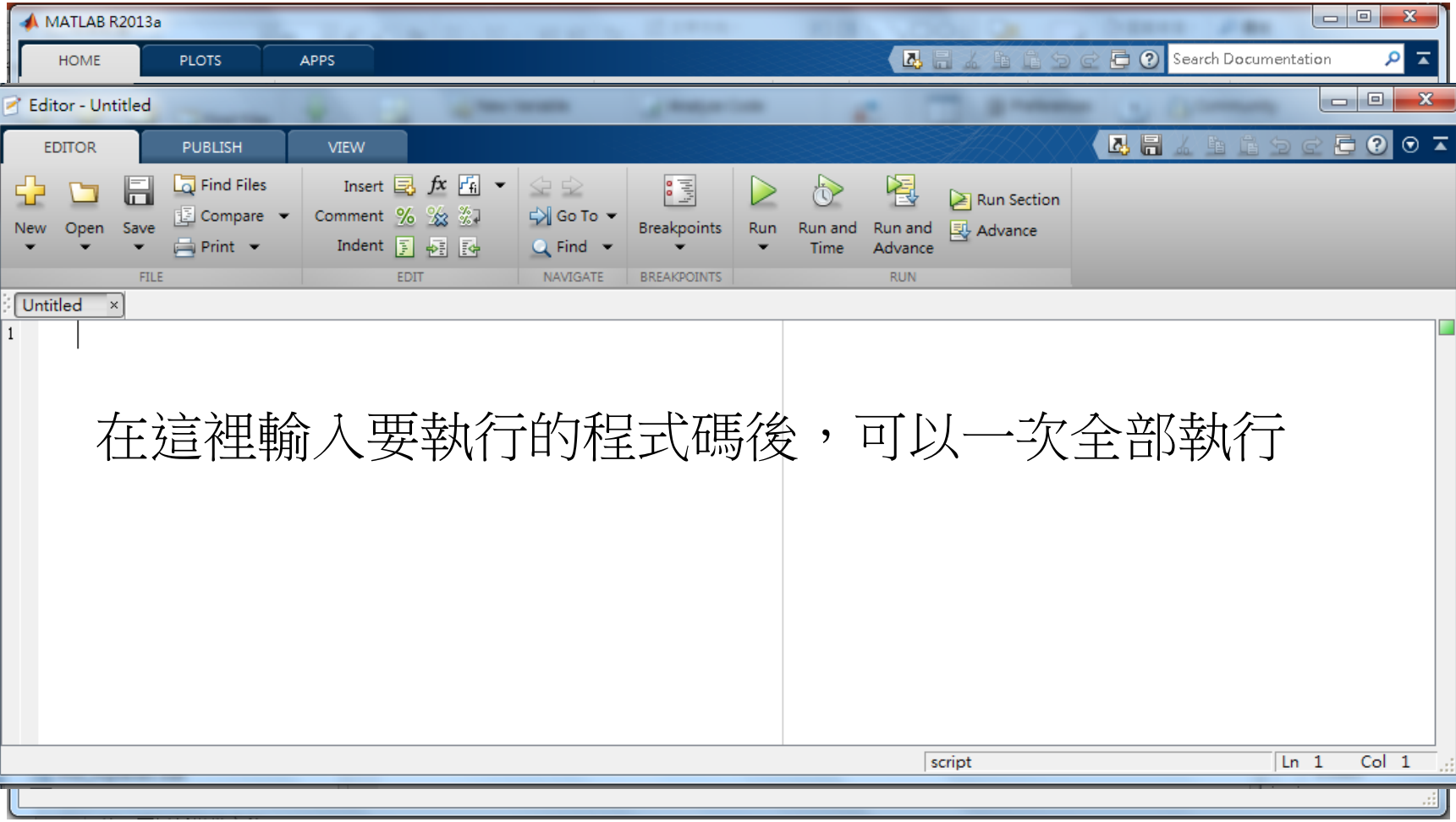


NATIONAL CENTRAL UNIVERSITY

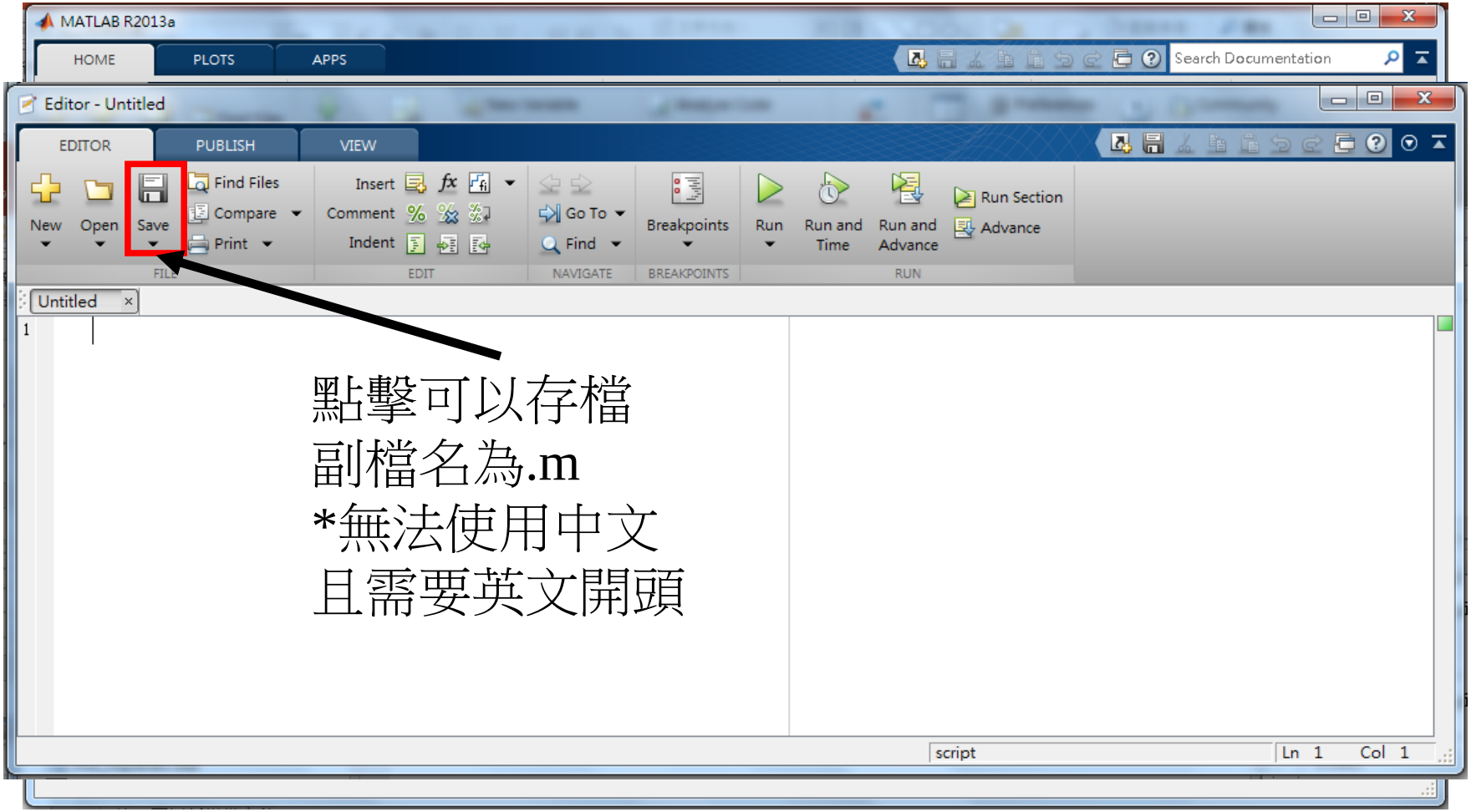
開啟 New Script



檔案編輯

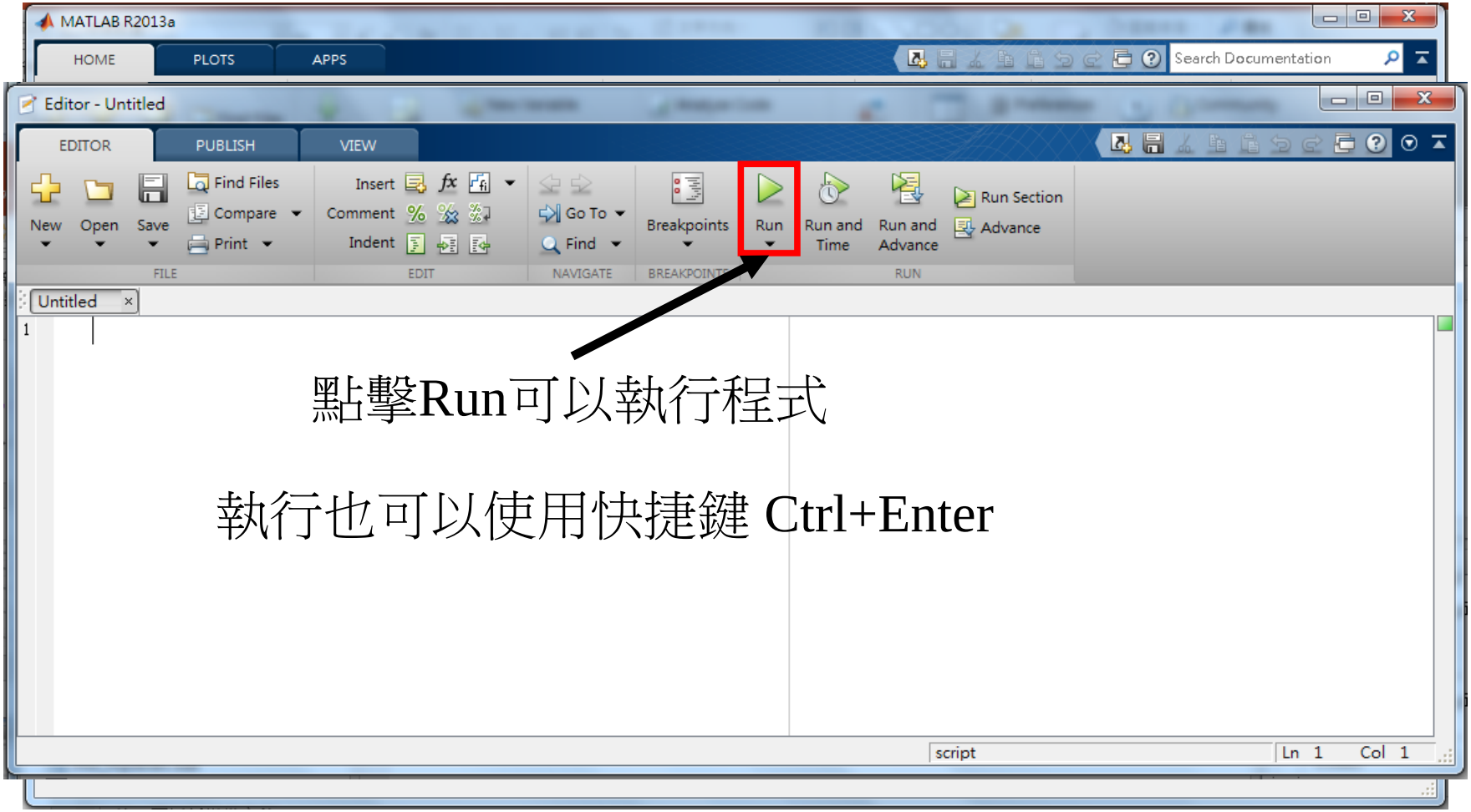

MATLAB R2013a
HOME
PLOTS
APPS
Search Documentation
Editor - Untitled
EDITOR
PUBLISH
VIEW
New
Open
Save
Find Files
Compare
Print
FILE
Insert
Comment
Indent
EDIT
Go To
Find
NAVIGATE
Breakpoints
BREAKPOINTS
Run
Run and Time
Run and Advance
Run Section
Advance
RUN
Untitled
1
在這裡輸入要執行的程式碼後，可以一次全部執行
script
Ln 1 Col 1

檔案編輯



點擊可以存檔
副檔名為.m
*無法使用中文
且需要英文開頭

檔案編輯



點擊Run可以執行程式

執行也可以使用快捷鍵 Ctrl+Enter

符號使用

% 程式的註解可以用%來分隔程式碼，%後面的部分將不會被視為程式碼的一部份，所以不會被執行。

```
>> 1+1 %2+2      ans =    2
```

%{%} 若想使用不只一行的註解，則可以用%{想要輸入的註解%}來達成。

```
%2+2
1+1
```

```
%{
123
456
789
%}
```

符號使用

%% 在編寫一整段程式碼的時候，可以使用%%來分隔不同部分，在執行程式碼的時候，如果使用快捷鍵Ctrl+Enter就可以只執行被%%隔出來的區域，但是使用Run就不會區分。

```
1+1
%%
2+2
```

```
ans =

     2
```

變數的使用

可以將計算的結果儲存到自己定義名稱的符號中，但是宣告名稱的時候需要注意：

1. 第一個字母必須是英文
2. 後續英文、數字和底線_可混和使用
3. 字母的大寫和小寫不同
4. 避免與其他函數的名稱相同
5. 最長只能使用31個字母(後面忽略)

```
mod =                    >> x=1

     1

                         x =

>> mod(1)

                                  1
ans =

     1

>> mod(1,3)
Index exceeds matrix dimensions.
```

變數的型態

使用的變數型態，主要有兩種：

1. 數字型態(int/double)
2. 文字型態(char/string)

因此在使用上1和2不可以視為相同的變數，在Matlab上直接輸入的數字，將會成為數字的型態，而字母則會被視為變數的存在。所以想要宣告文字型態的變數，則需要將想要使用的字母用 ’想要儲存的字母’ 隔開來，如此就可以建立文字的變數。

```
>> x=1
x =
     1
```

≠

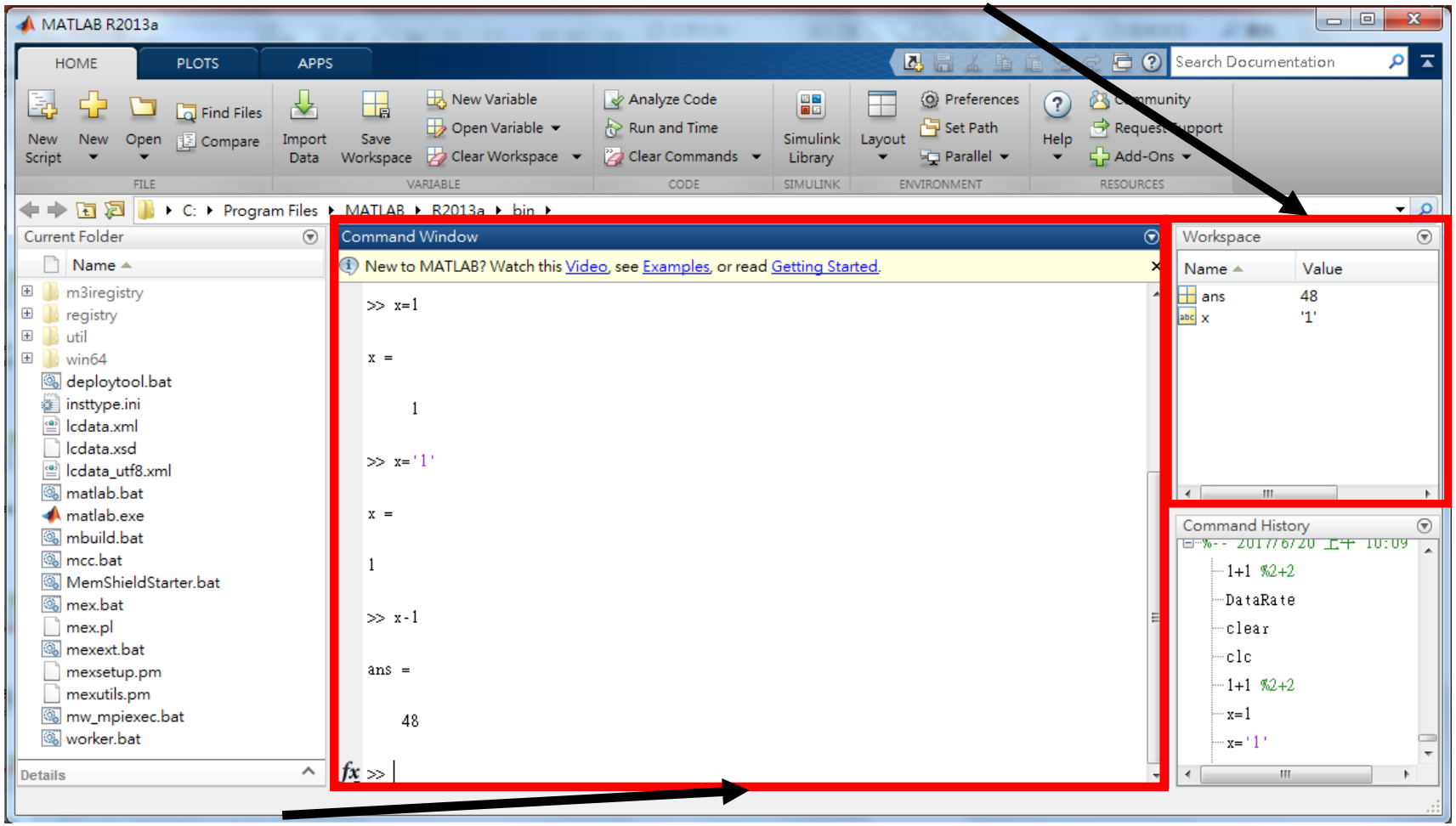
```
>> x='1'
x =
1
```

➡

```
>> x-1
ans =
    48
```

清除顯示和變數

使用了變數後，其結果會顯示於此



而執行的過程則會顯示在此

清除顯示和變數

因此想清除儲存的變數時，可以使用clear。清除顯示的結果則可以使用clc。有時候設計程式時，變數沒有考慮到重複執行的問題，因此會導致執行結果出錯，此時可以在最開頭的地方加入clear 和clc 確保重複執行程式時不會出錯。

```
clear
clc
%{
程式碼
...
%}
```

矩陣的使用

想要將同一組的資訊放在同一個變數中，這時候矩陣就是一個儲存方式。

儲存的方式為使用[A,B,C;D,E,F]，其中英文字母表示不同的變數（當然也可以直接輸入數字或'文字'），而,表示將資訊存在不同行（也可以使用空白隔開[A B]），而;則表示將資訊存在不同列，因此上面的存放方式為

A	B	C
D	E	F

矩陣的使用

但是再儲存矩陣時需要特別注意每個文字型態變數的長度相同，以及每個矩陣之間如果要存放時兩者之間的長度也需要相同。
例如說['str','s']是不能儲存成矩陣。

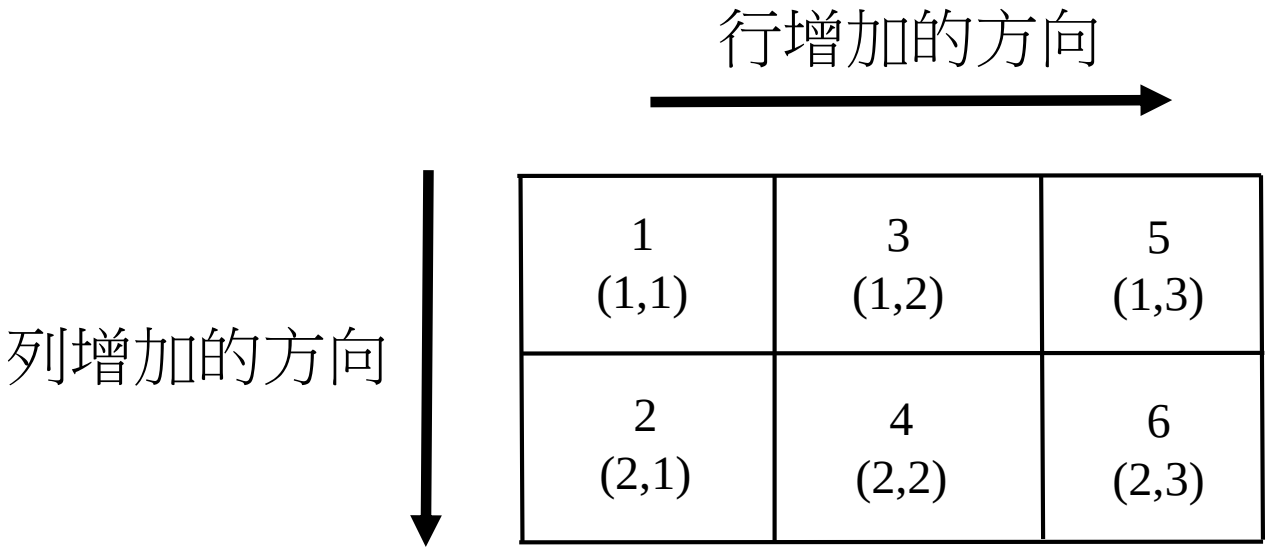
```
>> x=['str';'s']
Error using vertcat
Dimensions of matrices being concatenated are not consistent.
```

```
Subscripted assignment dimension mismatch.
```

只要出現這些紅字就表示矩陣間的長度有不同導致錯誤

矩陣的使用

變數想要呼叫矩陣中特定位置的資料時，則是使用X(1,1)或是 X(1) 來呼叫第一個儲存的資料，因此輸入兩個以上的數字時(A,B)A是指第幾列，B是指第幾行。只輸入一個數字時，則是會由列的方向開始排序，排到第幾個就是對應第幾個數字。



矩陣的使用

若想要呼叫大於一個資訊時，則可以使用:來呼叫。
如X(1:2,1)表示第一列第二列的第一行元素同時被呼叫。

```
>> x=[1 2 3;4 5 6;7 8 9]

x =

     1     2     3
     4     5     6
     7     8     9
```

```
>> x(1:2,1)

ans =

     1
     4
```

矩陣的使用

此外也可以利用:來建立矩陣，例如說想要創立以2為間隔，數字由5到11的矩陣，可以輸入5:2:11。

```
>> 5:2:11

ans =

     5     7     9    11
```

```
>> 5:2:12

ans =

     5     7     9    11
```

矩陣的使用

若不知道此矩陣的大小時，但是想要呼叫最後一個元素，則可以利用end來達成，如X(end,1)表示第一行最後一列的元素。單獨使用：則可以表示全部的元素。

```
>> x=[1 2 3;4 5 6;7 8 9]
x =
     1     2     3
     4     5     6
     7     8     9

>> x(end,1)
ans =
     7

>> x(2:end,:)
ans =
     4     5     6
     7     8     9
```

矩陣的使用

也可以使用size(X)這個函數來得到X矩陣的大小，顯示的結果會是[列 行]的矩陣。也可以直接size(X,1)來得到X有幾列， size(X,2)則是有幾行。

```
>> x=[1 2 3;4 5 6]

x =

     1     2     3
     4     5     6
```

```
>> size(x)

ans =

     2     3
```

```
>> size(x,1)

ans =

     2
```

```
>> size(x,2)

ans =

     3
```

矩陣的使用

也可以直接利用X(3,4)=2來建立矩陣的元素，但是沒有定義元素的部分，Matlab在數字形式會自動補0，而文字形式則會補null，表示這個地方沒有文字存在。

```
>> x=1

x =

     1
```

```
>> x(2,2)=1

x =

     1     0
     0     1
```

矩陣的使用

加減乘除的部分則是符合矩陣運算的方式，因此需要注意矩陣的大小是否可以符合矩陣運算。
另一方面，如果只是想要矩陣間的元素做運算，則可以使用.*的方式達成。例如想要X和Y的每一個元素互相相乘，則X.*Y即可（但是X和Y的大小要相同，否則會無法相乘）。

```
>> x=[1 2;3 4]

x =

     1     2
     3     4
```

```
>> y=[5 6;7 8]

y =

     5     6
     7     8
```

```
>> x*y

ans =

    19    22
    43    50
```

```
>> x.*y

ans =

     5    12
    21    32
```

顯示特定文字

若想顯示計算完，特定格式的文字和數字在指令區，則需要使用fprintf

fprintf(‘需要顯示的格式’,變數1,變數2,…)

當我們不明白計算後的結果是多少時，沒辦法直接將數字變成文字顯示，此時可以使用這個函數，將計算存放的變數指定至函數中，依照需要的文字格式將變數顯示。

例如現在想要輸入學生A的成績：

```
grades=50;
grades=grades^0.5*10;
fprintf('學生A的成績是%.2f\n',grades)
```

```
學生A的成績是70.71
fx >>
```

顯示特定文字

fprintf 特殊符號：

‘　　如同前面的文字格式變數，內容表示的即是文字。

%　　後面接序格式，對應變數的形式，詳細的對照請參照下一頁。

\　　後面接續特定的字母，\n 表示換行 \t 表示Tab \r\n 則是另外一種換行。

所以如果需要使用 ’ \ %這幾個特殊符號到文字中，則同時輸入兩次即可。

```
clear
clc
grades=50;
grades=grades^0.5*10;
fprintf('%% \\ '' \n',grades);
```

```
% \ '
>>
```

未使用也不影響結果

顯示特定文字

fprintf 特殊符號：

%s	將變數顯示文字的時候
%d	將變數顯示整數的時候
%f	將變數顯示浮點數的時候（小數點）
%g	比%f或%e顯示簡潔（可能沒有差異）
%e	科學記號表示

格式參考

Value Type	Conversion	Details
Integer, signed	`%d` or `%i`	Base 10
Integer, unsigned	`%u`	Base 10
	`%o`	Base 8 (octal)
	`%x`	Base 16 (hexadecimal), lowercase letters `a–f`
	`%X`	Same as `%x`, uppercase letters `A–F`
Floating-point number	`%f`	Fixed-point notation (Use a precision operator to specify the number of digits after the decimal point.)
	`%e`	Exponential notation, such as `3.141593e+00` (Use a precision operator to specify the number of digits after the decimal point.)
	`%E`	Same as `%e`, but uppercase, such as `3.141593E+00` (Use a precision operator to specify the number of digits after the decimal point.)
	`%g`	The more compact of `%e` or `%f`, with no trailing zeros (Use a precision operator to specify the number of significant digits.)
	`%G`	The more compact of `%E` or `%f`, with no trailing zeros (Use a precision operator to specify the number of significant digits.)
Characters or strings	`%c`	Single character
	`%s`	Character vector or string array. The type of the output text is the same as the type of `formatSpec`.

https://www.mathworks.com/help/matlab/ref/sprintf.html?s_tid=srchtitle

建立文字的相似函數

sprintf ：
與fprintf相似，但是sprintf是建立文字格式的字串輸出，因此可以將sprintf建立的結果儲存至變數中。

```
grades=50;
grades=grades^0.5*10;
result=sprintf('學生A的成績是%.3f',grades)

result =

學生A的成績是70.711
```

函數使用

Matlab 彩蛋，可以執行結果：

vibes
teapotdemo
logo
fifteen
xpbombs
why

Matlab 程式流程控制篇

2017/7/7

林崇聖

條件選擇

一般在執行的過程中，會需要利用條件去選擇要執行的動作，此時就需要使用條件選擇的功能。

第一種選擇的方法：

if(第一個判斷的條件)%True or False
若第一個判斷成立則執行這裡的程式碼
elseif(第二個判斷的條件)
若第二個判斷成立則執行這裡的程式碼
elseif(…) %elseif 可以使用很多次
…
else %不需要輸入判斷條件
上述的判斷皆未成立時則執行這裡的程式碼
end %判斷結束的地方需要加入end

*可以只使用if-end或if-elseif-end或if-else-end

條件選擇

判斷的條件一旦成功，則後面的條件不會判斷就會結束這個結構。

```
clear
clc
Value=1
if(Value==1|Value==2)
    fprintf('%d\n',Value*2)
elseif(Value==2|Value==3)
    fprintf('%d\n',Value^3)
elseif(Value==4|Value==5)
    fprintf('%f\n',Value^0.5)
else
    fprintf('%f\n',Value^2)
end
```

```
Value =

     1


2
```

```
clear
clc
Value=2
if(Value==1|Value==2)
    fprintf('%d\n',Value*2)
elseif(Value==2|Value==3)
    fprintf('%d\n',Value^3)
elseif(Value==4|Value==5)
    fprintf('%f\n',Value^0.5)
else
    fprintf('%f\n',Value^2)
end
```

```
Value =

     2


4
```

條件選擇

另一種選擇的方法: switch-case-otherwise-end

switch 變數

case 數值/文字 *大於一種選擇時用{選擇1,選擇2,…}

otherwise

end

比對變數存放的數值或文字，尋找是否有符合的case存在，若都不存在時則執行otherwise，且與if相同由上而下執行，若已找到符合的case則下面的case、otherwise皆不會執行。

*與if相似可以只放一個case，otherwise並非必要。

條件選擇

```
value=1                          value =
switch value
    case 1                             1
        a=1
    case 2
        b=2                  a =
    otherwise
        c=3                            1
end
```

```
value=3                          value =
switch value
    case 1                             3
        a=1
    case {2,3}
        b=2                  b =
    otherwise
        c=3                            2
end
```

條件選擇

條件選擇需注意存放的變數是否皆存在，由於執行至某個符合的項目就會跳出，後面選項中的變數可能不會被執行，後續若還是需要使用到此變數，就會出現變數未使用的錯誤，因此注意每個選項的變數使用，是否符合後續的程式執行。

```
value=1
if value==1
    a=1
else
    b=2
end
b>1
```

```
value =

     1

a =

     1

Undefined function or variable 'b'
```

迴圈

有時候我們會需要執行重複的動作，或是每次動作與上次動作類似，此時若需要不斷重複寫入近乎相同的程式碼，則會使整體的程式碼變長、變複雜且耗費時間，迴圈即可處理這類行的動作。

例如：
計算1+2+…+50
若是要自行輸入1+～+50，則會使程式碼變的很長，而且很花時間。
若是要加到更多的數字，或是減少數字，也是非常耗費時間去修改。

迴圈

for-end 迴圈：

for 變數名稱=起使數值:(間隔):結束數值

執行程式

end

如同前面建立矩陣 1:2:8的用法，由1開始計算，以2為間隔，加到不大於8為止。所以對應迴圈會執行的數值為：

1 3 5 7

這四個數值會對應到定義的變數中，每次迴圈依序改變變數的數值。

```
for i=1:2:8
    fprintf('迴圈的變數i=%d\n',i)
end
```

```
迴圈的變數i=1
迴圈的變數i=3
迴圈的變數i=5
迴圈的變數i=7
```

迴圈

while-end 迴圈：

```
while 變數
        執行程式
end
```

當變數是數值 1（True）時，while 將會執行，因此可用來判別當某數值發生之前仍然繼續執行。

```
i=0
while i<=8
    i=i+2;
    fprintf('迴圈的變數i=%d\n',i)
end
```

```
i =

     0


迴圈的變數i=2
迴圈的變數i=4
迴圈的變數i=6
迴圈的變數i=8
迴圈的變數i=10
```

迴圈

*需注意給予的迴圈條件，有可能導致無法離開while迴圈，造成程式卡住的情形，此時只要在Matlab的指令區按下Ctrl+C即可強制停止程式執行。

*使用break的函數可以強制跳出迴圈，continue則可以跳過此次迴圈。

```
i=0
while i<=8
    i=i+2;
    if i==6
        continue
    end
    fprintf('迴圈的變數i=%d\n',i)
end
```

```
i =

     0

迴圈的變數i=2
迴圈的變數i=4
迴圈的變數i=8
迴圈的變數i=10
```

```
i=0
while i<=8
    i=i+2;
    if i==6
        break
    end
    fprintf('迴圈的變數i=%d\n',i)
end
```

```
i =

     0

迴圈的變數i=2
迴圈的變數i=4
```

迴圈-練習

不同數值之加總，可以快速更改。

```
sumi=0
for i=1:50
    sumi=sumi+i;
end
sumi
```

```
sumi =

     0


sumi =

        1275
```

```
sumi=0
for i=1:100
    sumi=sumi+i;
end
sumi
```

```
sumi =

     0


sumi =

        5050
```

迴圈-練習

不同數值之加總，可以快速更改。

```
sumall=0;
i=1;
while sumall<=1274
    sumall=sumall+i;
    i=i+1;
end
sumall
```

```
sumall =

        1275
```

```
sumall=0;
i=1;
while sumall<=5049
    sumall=sumall+i;
    i=i+1;
end
sumall
```

```
sumall =

        5050
```

Matlab 繪圖篇

2017/7/7

林崇聖

繪圖指令

Matlab有許多的繪圖指令，這裡將只會挑選一小部分的繪圖方式來介紹。
使用到的指令：
plot
plotyy
semilogy、semilogx
loglog
plot3

繪圖指令

首先是最主要的plot，大部分的繪圖需求，使用這個指令即可完成。

plot(x座標,y座標,繪圖形式,選擇格式設定,格式設定輸入值,…)

想要畫在x軸、y軸的哪一點就分別輸入數字形式的矩陣至座標中，若只想畫一點，也可只輸入兩個數值。接著後面的繪圖形式、格式設定若無給予，則會使用預設值，若要設定則可以針對線的形式、顏色、大小等做調整。

繪圖指令

繪圖形式可分為：

‘-*b’

-　第一項表示線條形式（line style）

*　第二項表示點的形式（marker）

b　第三項表示顏色的形式（color）

所以-是實線，點用*的形式，b表示使用藍色

繪圖指令

其他線的形式可參考：

Line Style	Description
-	Solid line (default)
--	Dashed line
:	Dotted line
-.	Dash-dot line

繪圖指令

其他點的形式可參考：

Marker	Description
o	Circle
+	Plus sign
*	Asterisk
.	Point
x	Cross
s	Square
d	Diamond
^	Upward-pointing triangle
v	Downward-pointing triangle
>	Right-pointing triangle
<	Left-pointing triangle
p	Pentagram
h	Hexagram

繪圖指令

其他顏色的形式可參考：

Color	Description
y	yellow
m	magenta
c	cyan
r	red
g	green
b	blue
w	white
k	black

繪圖指令

選擇的格式：
若要調整線的寬度，0.5是預設值
‘LineWidth’,大小　　如plot(x, y, ‘linewidth’,0.5)
若要調整點的的大小，6是預設值：
‘MarkerSize’,6

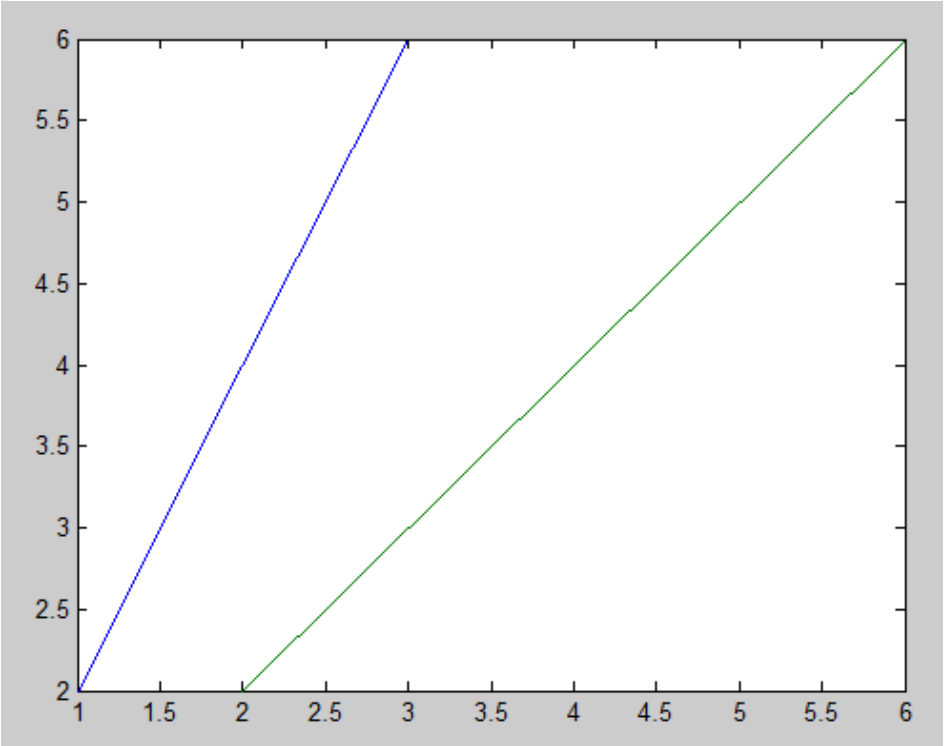
其他可參考：
https://www.mathworks.com/help/matlab/ref/plot.html?searchHighlight=plot&s_tid=doc_srchtitle

繪圖指令

```
x =                    y =

     1     2     3                2     4     6
```

另外可以輸入多組x,y座標來繪圖：

```
>> plot(x,y,y,y)
```

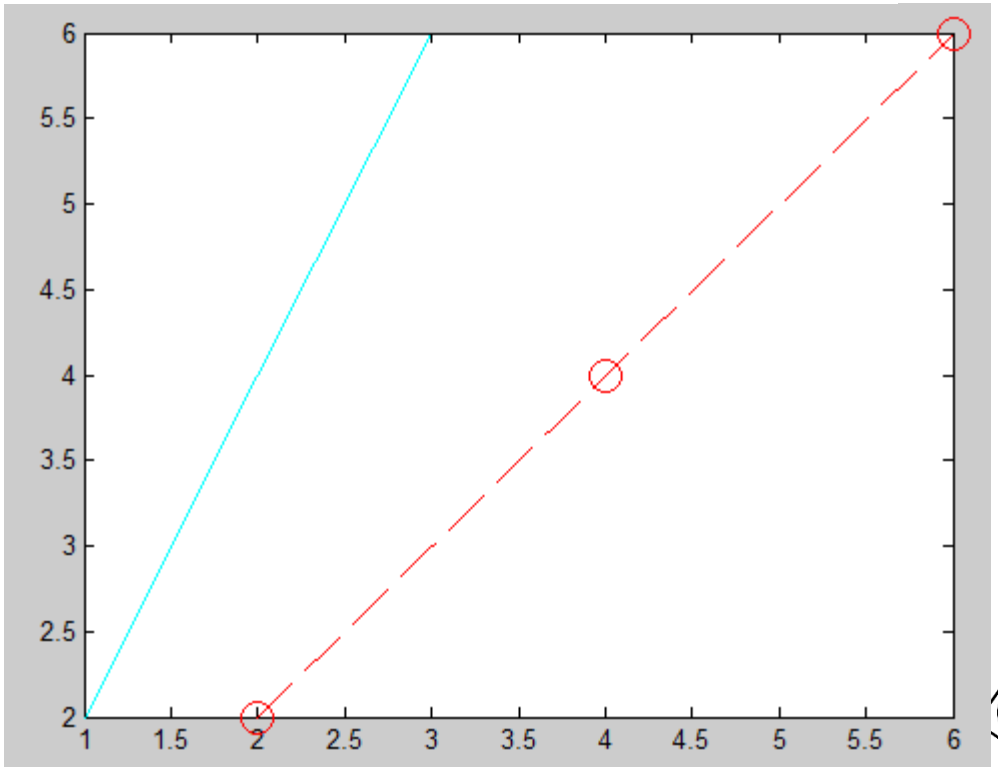


繪圖指令

```
x =                                   y =

     1     2     3                         2     4     6
```

並且可以同時調整多組的繪圖形式：

```
>> plot(x,y,'-c',y,y,'--Or','MarkerSize',12)
```



繪圖指令

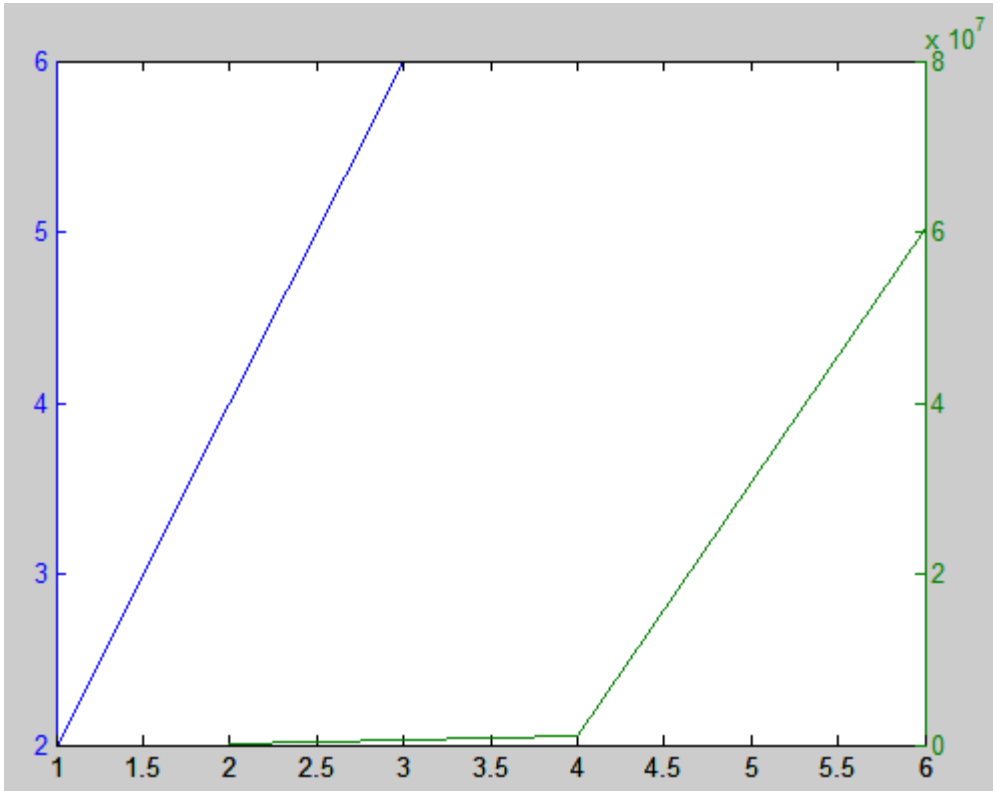
plotyy則是用來建立兩個y軸的圖，因此輸入的形式必須是兩組x,y。
plotyy(x1,y1,x2,y2,funciton1,function2)
但是這裡要控制圖的線寬或顏色等，則不能直接使用plotyy達成，後面所輸入的是不同y軸的圖案類別，所以funcion是指可以達成function(x,y)的函數，如plot(x,y)，這樣一來第一組x1,y1就是使用這個function來繪圖。
plotyy(x1,y1,x2,y2,'plot','loglog')

繪圖指令

plotyy的繪圖

```
x =                          y =

     1     2     3               2     4     6
```

```
>> plotyy(x,y,y,y.^10)
```



繪圖指令

semilogy、semilogx、loglog屬於對數繪圖，也就是座標軸的刻度是使用過log()處理，刻度分布不是等比例的變化，因此適用於數值變化極大的情況，例如數值一開始在10^-3附近的變化，可是卻突然攀升到10^5次方時，要將在小範圍的變化表現出來，plot會有困難，這時候使用對數圖就有幫助。

semilogy/x是指對數的座標軸只有x軸或y軸

loglog則是兩個座標軸皆為對數做圖

使用的形式幾乎等同於plot的用法

loglog(x,y,繪圖形式,…)

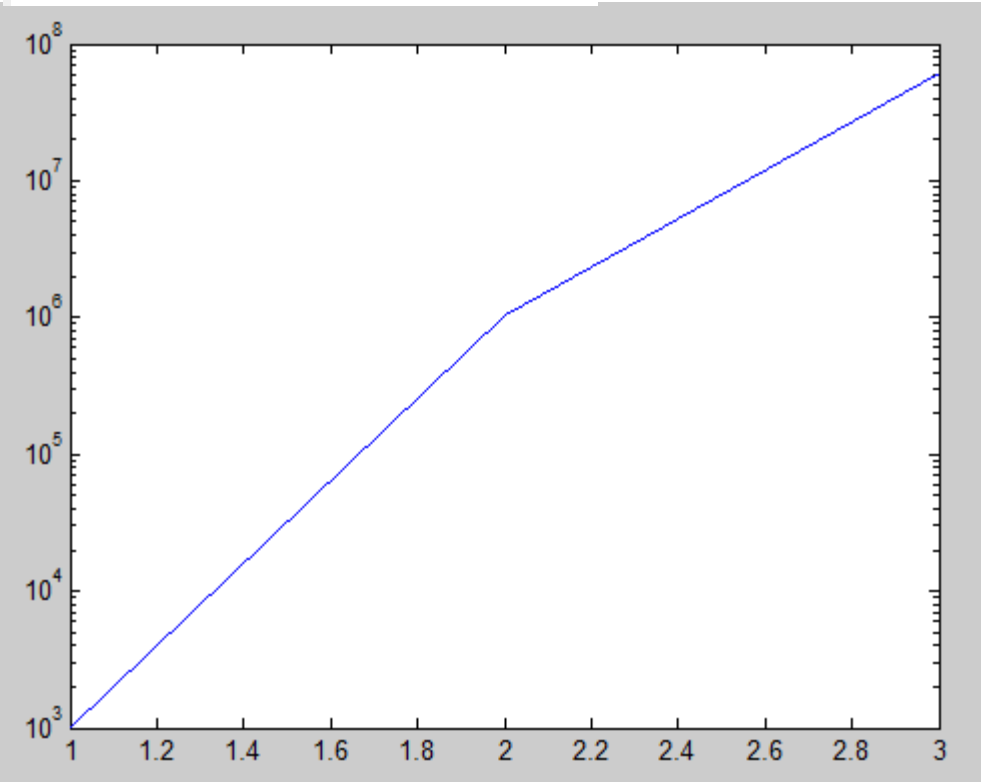
semilogy(x,y,繪圖形式,…)

繪圖指令

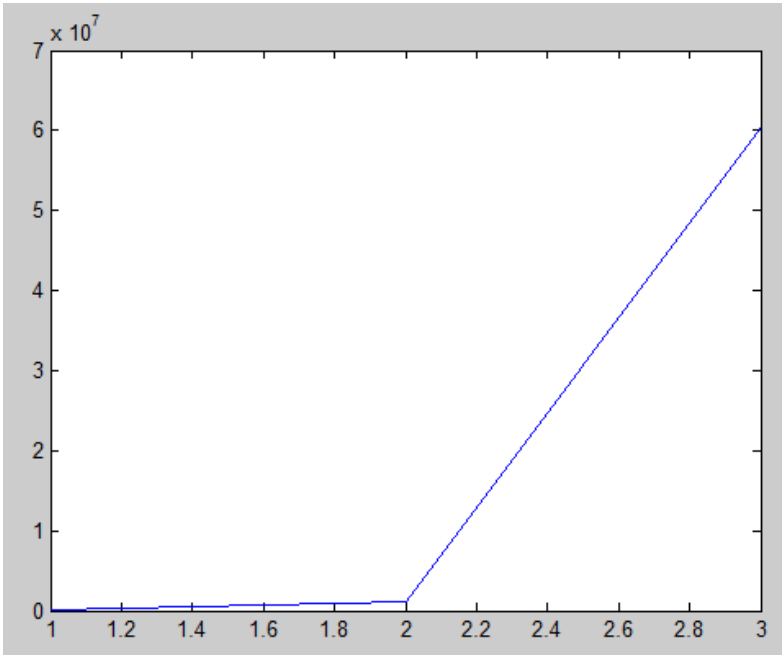
對數的繪圖：

```
x =
     1     2     3

y =
     2     4     6
```

```
>> semilogy(x,y.^10)
```



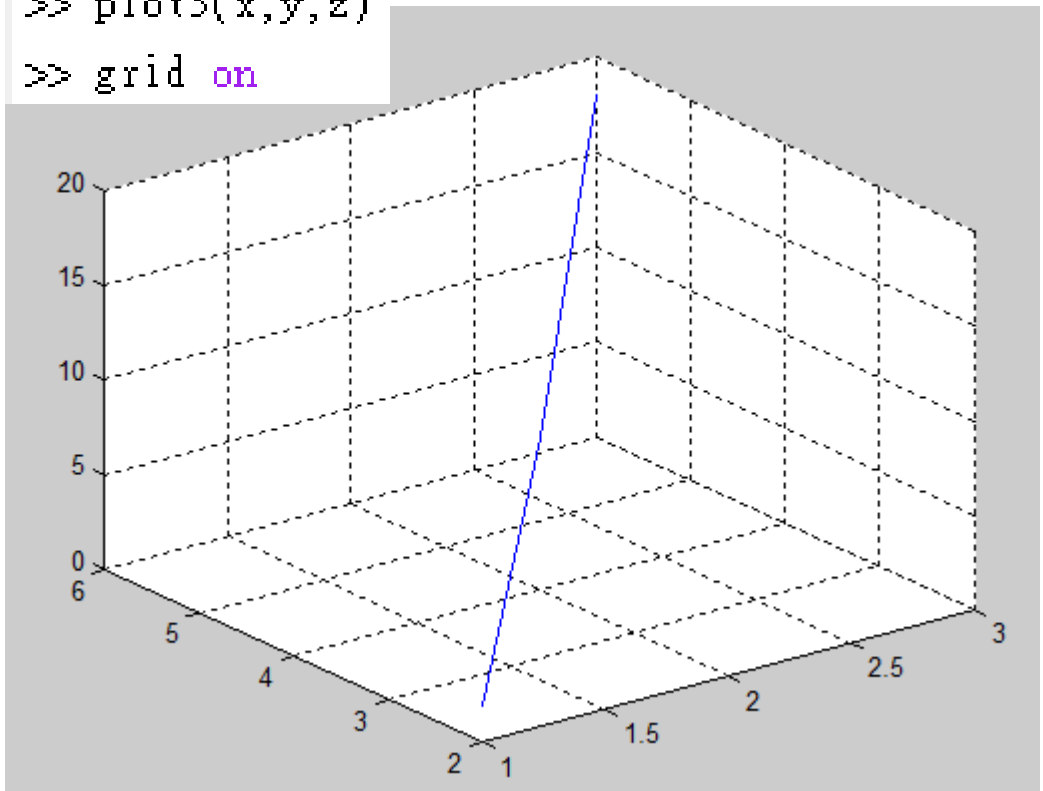

```
>> plot(x,y.^10)
```



繪圖指令

plot3則是用來繪製3D線圖，如果想要使用x,y,z的座標軸，就需要用這個函數，其餘的用法等同於plot。
*grid on 則是開啟背景網格（虛線）

```
>> plot3(x,y,z)
>> grid on
```



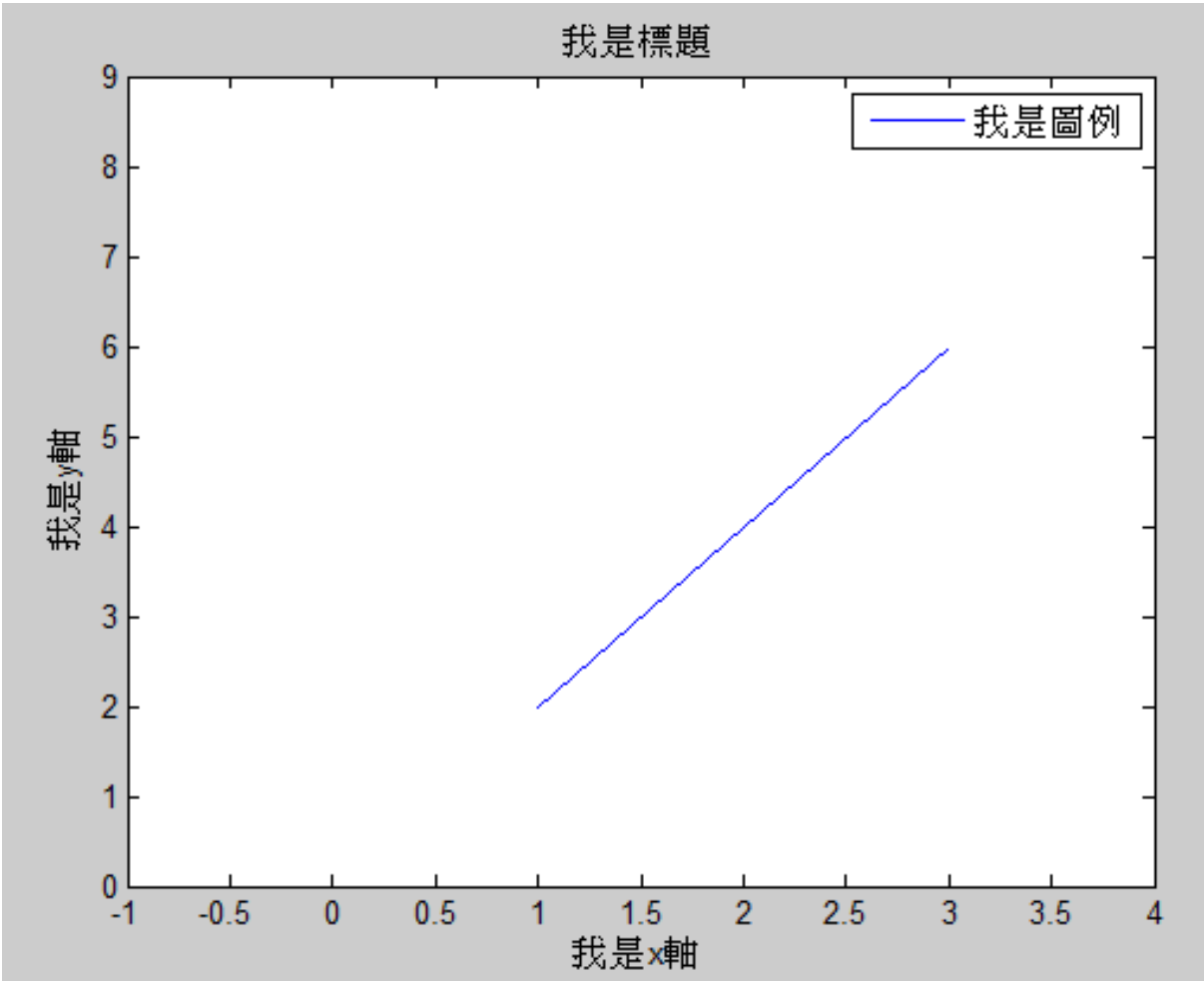

```
x =

     1     2     3

y =

     2     4     6

z =

     2     8    18
```

其他繪圖指令

如果要調整的不是繪圖的線，而是標題或是座標軸，這時候需要用到不同的指令。
xlabel：可以給予x座標軸名稱
ylabel：可以給予y座標軸名稱
title：可以給予圖的名稱
legend：可以給予圖例
axis：可以調整座標軸的範圍

其他繪圖指令

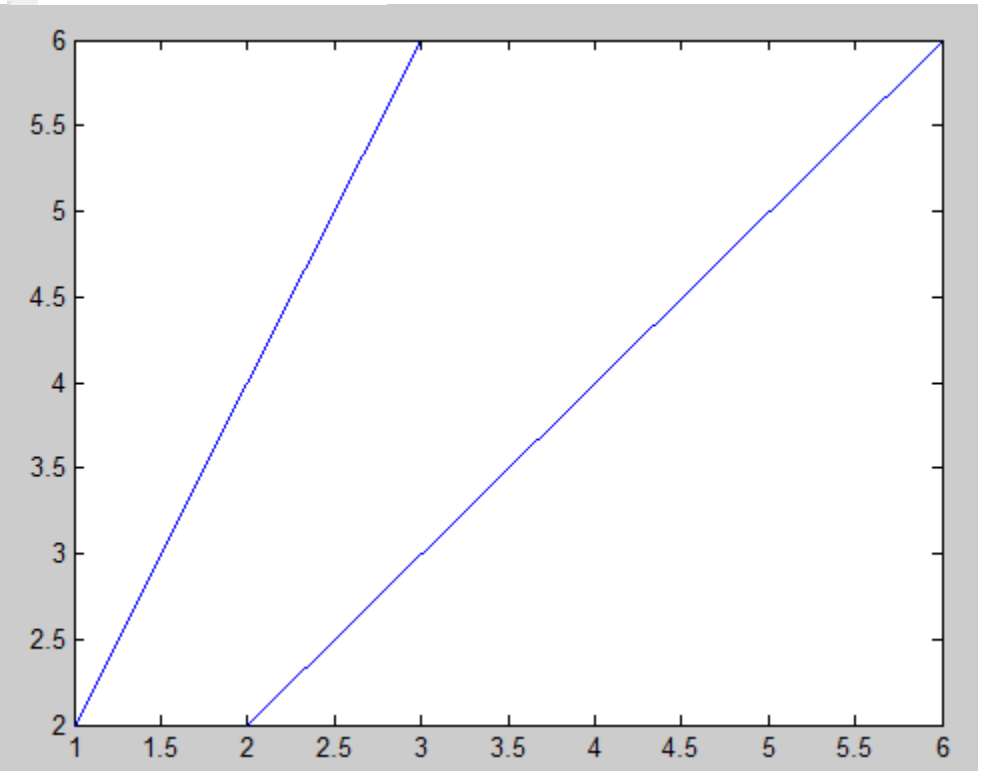
```
>>  plot(x,y)
xlabel('我是x軸')
ylabel('我是y軸')
title('我是標題')
legend('我是圖例')
axis([-1 4 0 9])
```



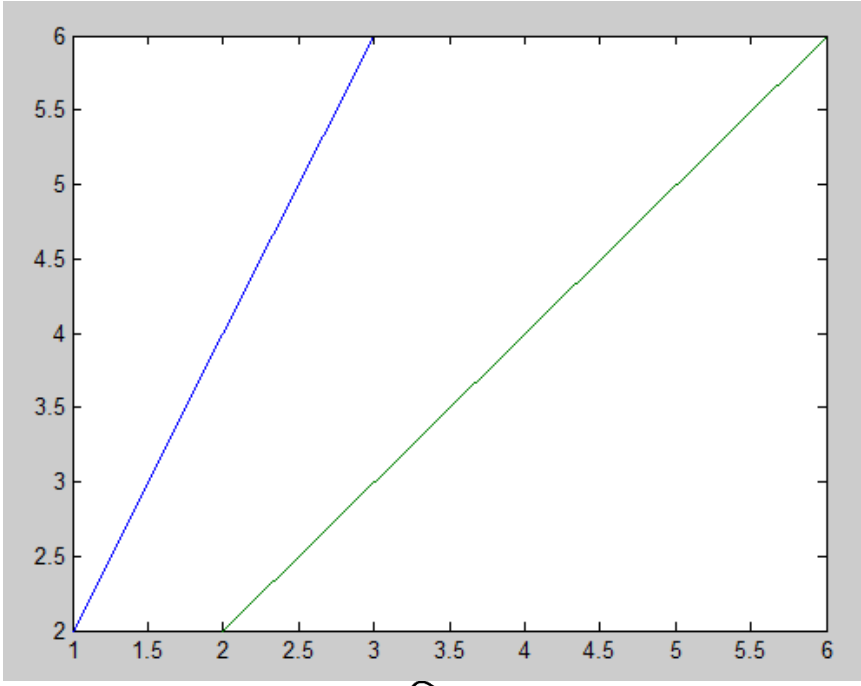
其他繪圖指令

若想要將每次使用的plot都繪製在同一張圖上時可以使用hold on，輸入hold off就可以關閉這個功能

```
>> plot(x,y)
>> hold on
>> plot(y,y)
```




```
x =

     1     2     3

y =

     2     4     6

>> plot(x,y,y,y)
```



其他繪圖指令

若需要繪製兩張以上的圖在同一張影像上時，則使用subplot

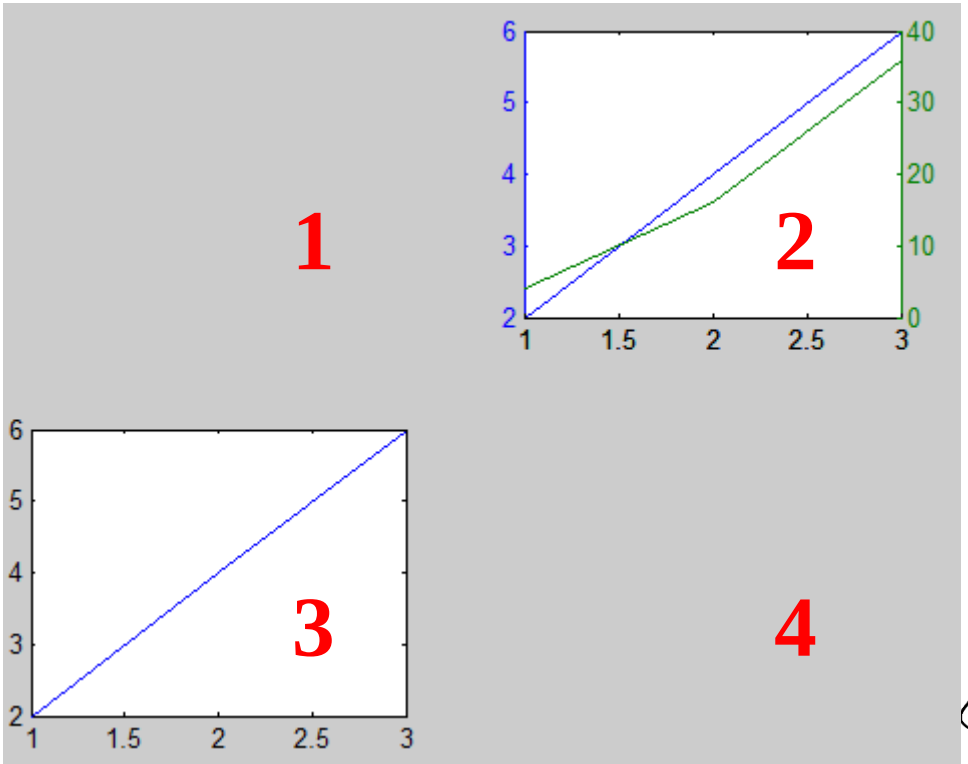
subplot(數字1,數字2,擺放位置)，將會由數字1、2依照矩陣的切割方式，切割出可以擺放的區域，接著對應擺放位置的編號，放入對應的區域中。

例如使用subplot(2,3,5)，則繪圖後會被放置在(2,3)切割的區域中編號第五格的位置。

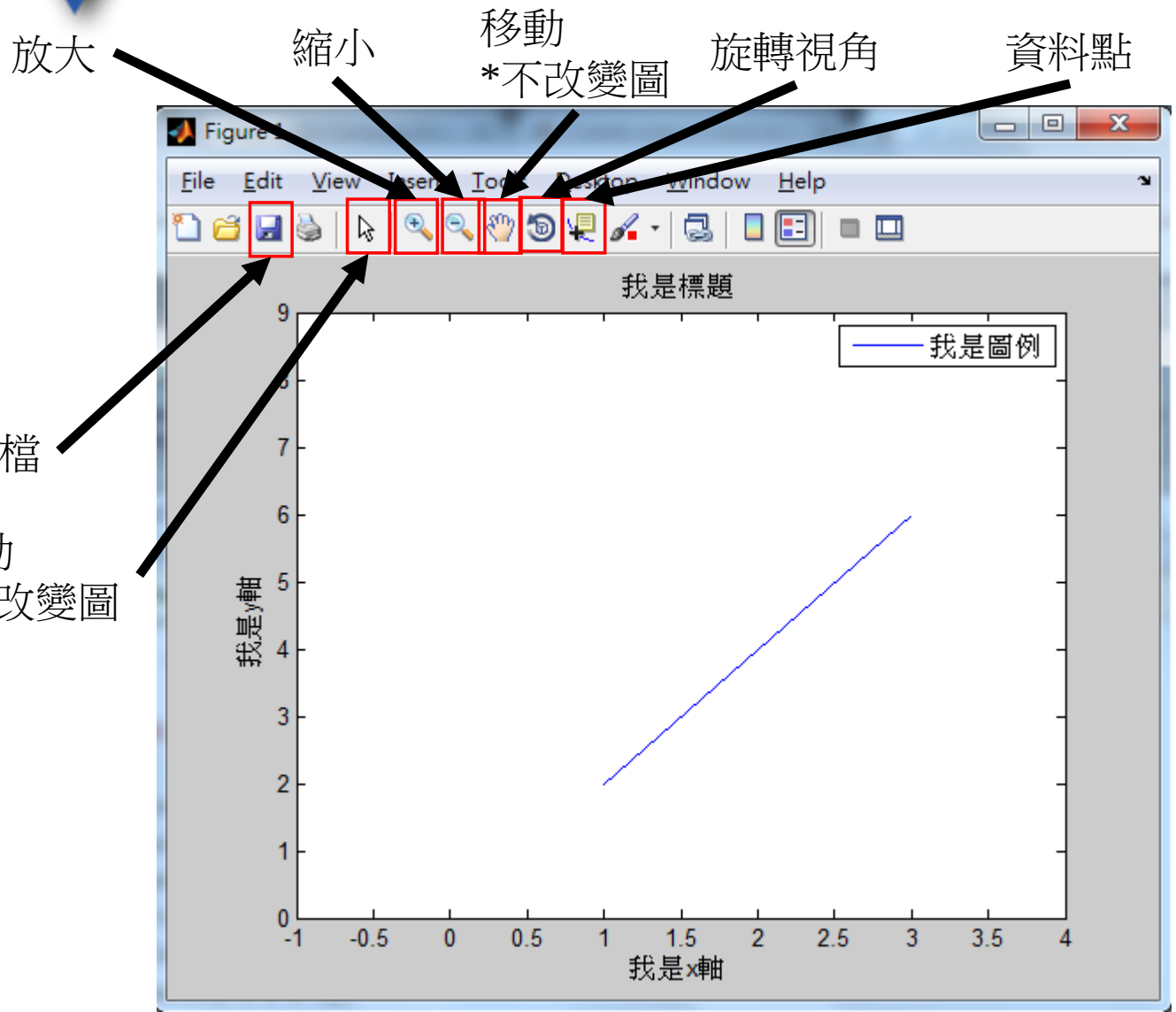
1	2	3
4	5	6

其他繪圖指令

```
>> subplot(2,2,3)
>> plot(x,y)
>> subplot(2,2,2)
>> plotyy(x,y,x,y.*y)
```

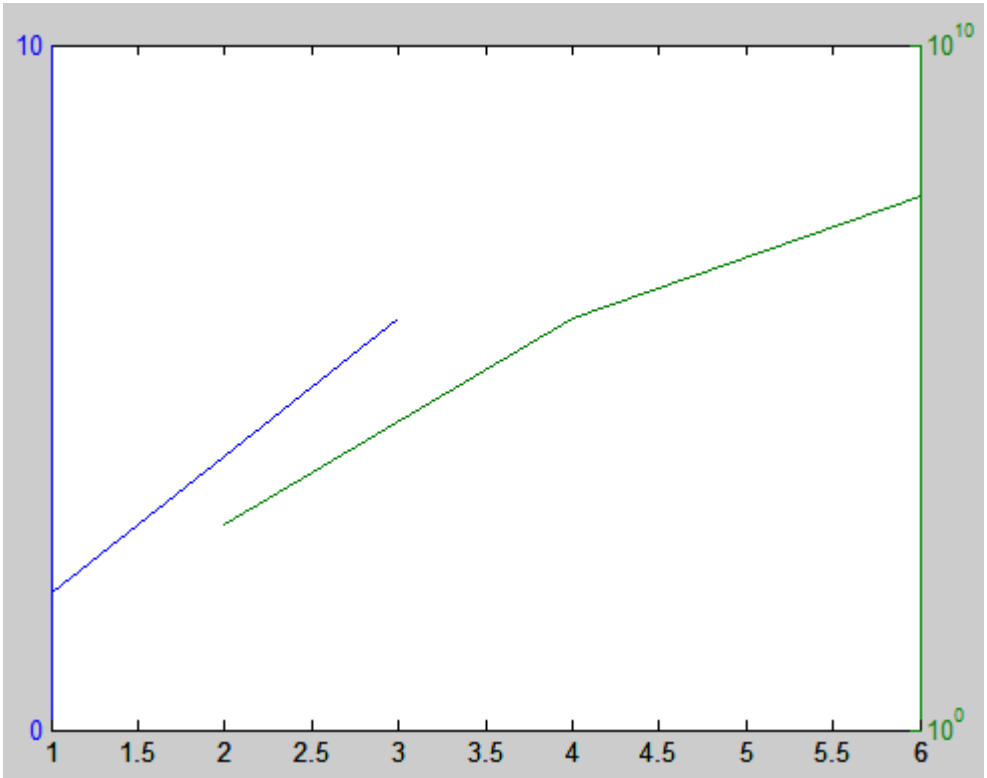


繪圖視窗



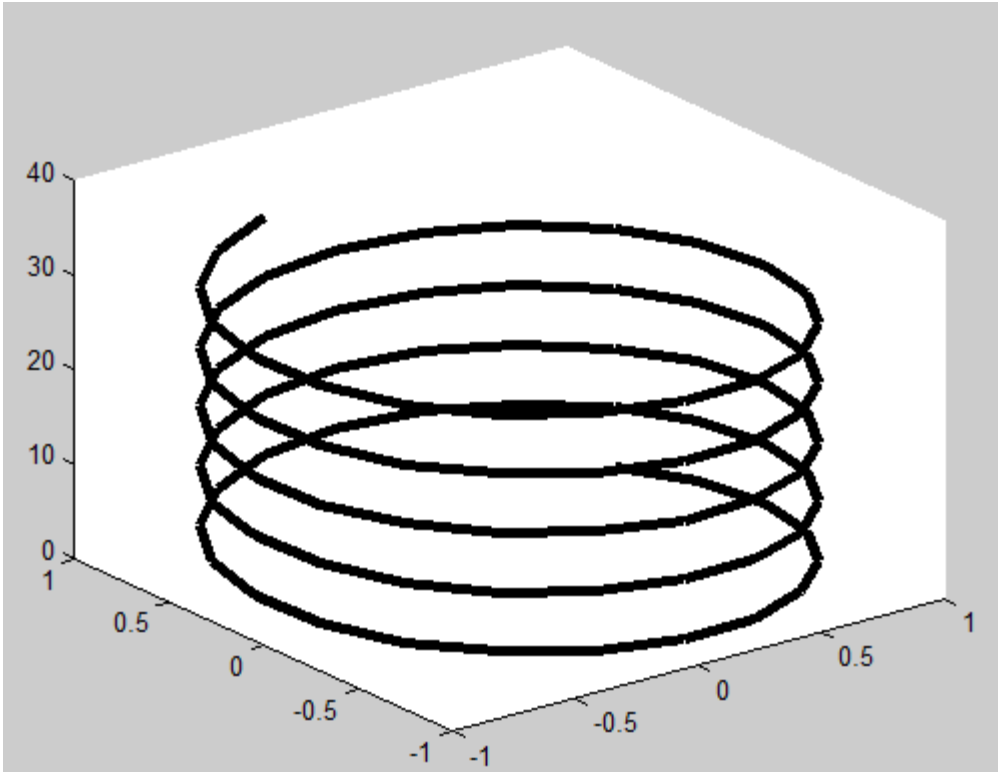
繪圖-練習

```
x =                                    y =

     1     2     3                          2     4     6

>> plotyy(x,y,y,y.^10,'plot','semilogy')
```



繪圖-練習

```
>> t=1:0.1*pi:2*pi*5;
x=sin(t);
y=cos(t);
plot3(x,y,t,'k','LineWidth',4)
```



Matlab 寫檔與讀檔篇

2017/7/7

林崇聖

SPL
NATIONAL CENTRAL UNIVERSITY

寫、讀檔指令

Matlab同樣有許多的讀寫檔指令，因此不會全部做介紹。一般在處理資料時，不可能自己手動不斷輸入資料，因此資料會以檔案的方式儲存，接著才用程式讀取檔案內的資料，讀取以後再進行處理，處理完畢後又需要將這些資料再次儲存，因此就需要使用寫檔的方式建立處理完資料的檔案。

主要的指令有

fopen

fclose

fprintf

fscanf、fseek、ftell

dlmwrite

dlmread

寫、讀檔指令

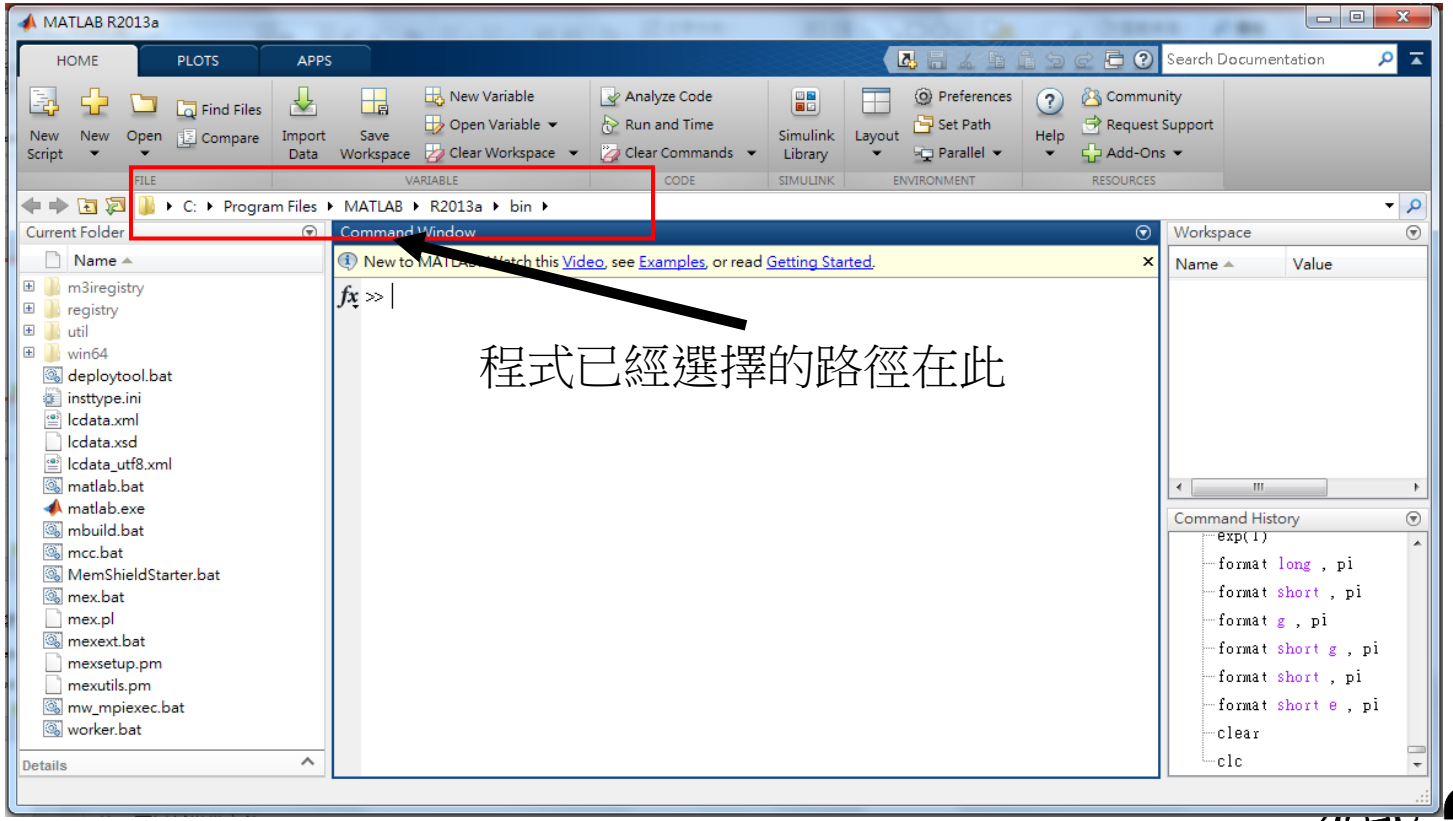
不論要讀取檔案或是寫入檔案，一開始必須先指定路徑開啟/建立目標檔案，才能使用後續部分的指令。
開啟/建立的指令為fopen(‘儲存的路徑’,’開啟檔案的權限設定’)
‘儲存路徑’ 如絕對路徑 ’C:\matlab\data\Data.txt’
‘開啟檔案的權限’ 如 ‘w+’

```
>> save_path='C:\matlab\data\Data.txt'
```

```
>> fopen(save_path,'w+')


ans =


     4
```

寫、讀檔指令

*可注意若非絕對路徑，單純輸入檔案名稱時
如 Data.txt
則此時Matlab會搜尋程式本身已經選擇的路徑做搜尋



寫、讀檔指令

開啟檔案的權限設定：如果未設定則預設為’r’

'r'	Open file for reading.
'w'	Open or create new file for writing. Discard existing contents, if any.
'a'	Open or create new file for writing. Append data to the end of the file.
'r+'	Open file for reading and writing.
'w+'	Open or create new file for reading and writing. Discard existing contents, if any.
'a+'	Open or create new file for reading and writing. Append data to the end of the file.
'A'	Open file for appending without automatic flushing of the current output buffer.
'W'	Open file for writing without automatic flushing of the current output buffer.

常使用的有：’r’僅讀取檔案，’w’僅寫入檔案（若無存在檔案會自動建立檔案，若有存在檔案則會覆蓋），’a’與’w’只差別在不會覆蓋檔案，而是會將新資料接著寫入檔案最後面，’r+’、’w+’、’a+’與原本的功能多加讀取/寫入的功能，

寫、讀檔指令

[開啟檔案的編號,錯誤訊息]=fopen(檔案路徑,權限設定)
fopen還有同時輸出兩個資訊的形式，每個開啟的檔案皆會建立一個指定的數值，後面再使用時即是這個編號。若開啟/建立失敗時則檔案編號固定會回傳-1，且會說明為何失敗。

```
>> [fileID,errmsg]=fopen(save_path,'r')

fileID =

    -1


errmsg =

No such file or directory
```

寫、讀檔指令

且使用完檔案後，必須要使用fclose(‘all’)的指令，釋放讀取檔案的指令，若一直開啟檔案，則會佔用系統資源，若使用大量的讀寫檔案時，容易造成應用程式終止等問題。
因此完整的讀檔方式為：
fid=fopen(檔案路徑,存取權限);
利用fid使用其他指令達成檔案處理的過程
fclose(‘all’);

*也可針對特定開啟的檔案執行關閉
fclaose(‘fid’)

```
>> fopen(save_path,'w+')

ans =

     4

>> fclose('all')

ans =

     0
```

寫、讀檔指令

寫入檔案的指令：
fprintf
這個指令除了能將特定文字顯示於指令區以外，同樣也能寫入特定文字至檔案中，並且使用方式與顯示於指令區時相同，僅需要多加一個開啟檔案編號的變數輸入即可。

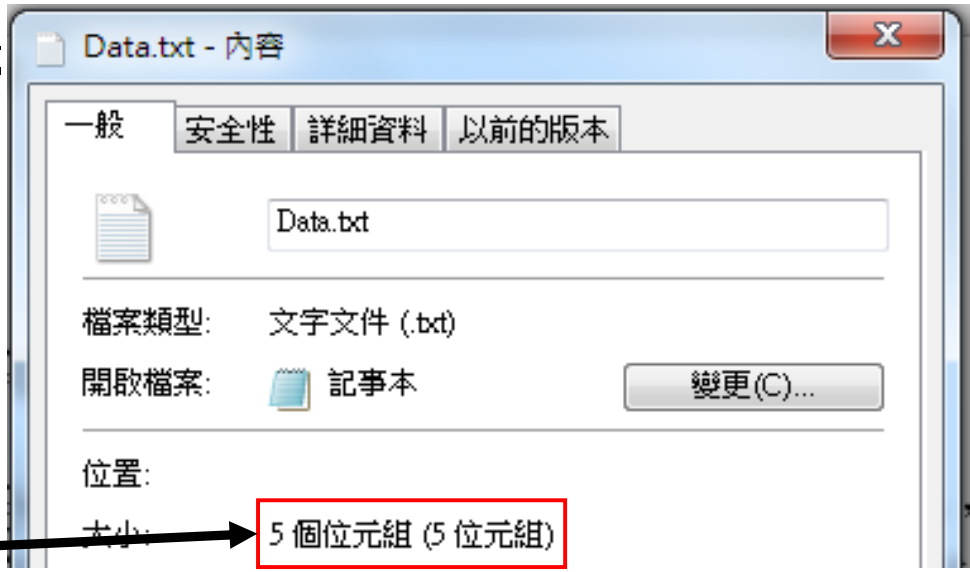
fprintf(開啟檔案之編號,‘需要顯示的格式’,變數1,變數2,…)
開啟檔案之編號，為前面使用fopen後輸出的變數
開啟檔案之編號=fopen(檔案路徑,存取權限)

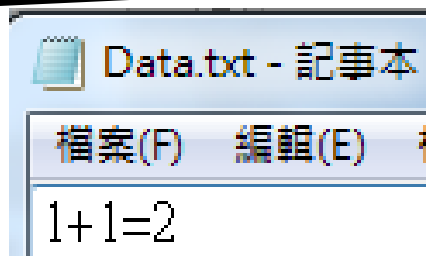
寫、讀檔指令

fprintf：

*若讓fprintf輸出時，則會說明特定文字的大小為多少，這裡的5表示 '1+1=2' 這串文字大小為 **5位元組**（bytes）。

```
>> fid=fopen(save_path,'w+');
fprintf(fid,'1+1=%d',1+1)
fclose('all');

ans =

     5
```



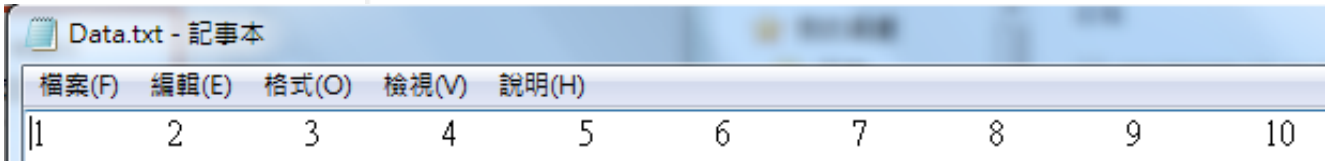


寫、讀檔指令

讀取檔案的指令：
fscanf(開啟檔案之編號,讀取資料之特定格式)
將檔案內容依照特定的格式讀出，特定格式與fprintf使用上相同。

```
>> fid=fopen(save_path,'w+');
for i=1:10
    if i==10
        fprintf(fid,'%d',i);
    else
        fprintf(fid,'%d\t',i);
    end
end
fseek(fid,0,-1);
fscanf(fid,'%d\t')
fclose('all');
```

```
ans =

     1
     2
     3
     4
     5
     6
     7
     8
     9
    10
```



寫、讀檔指令

fseek、ftell指令的使用，與寫入後的游標位置有關，當fprintf寫入完畢後，檔案若未使用fopen重新開啟，也未使用fclose('all')，則此時游標位置將停留在檔案最後。
若想要繼續使用其他指定，如fscanf就會發生讀不到東西的狀況。

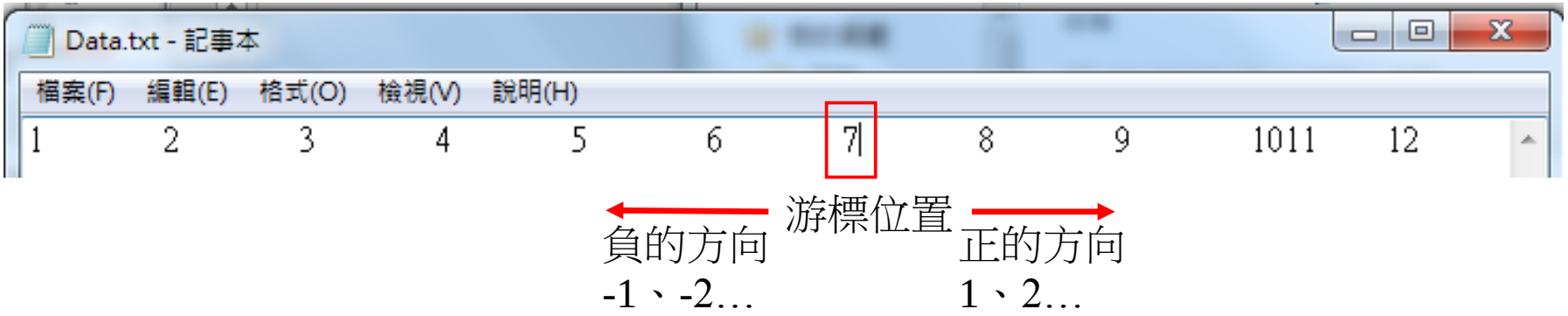
```
>> fid=fopen(save_path,'w+');
for i=1:10
    if i==10
        fprintf(fid,'%d',i);
    else
        fprintf(fid,'%d\t',i);
    end
end
%fseek(fid,-20,1);
fscanf(fid,'%d\t')
fclose('all');

ans =

     []
```

```
>> ftell(fid)
fseek(fid,0,-1);
ftell(fid)

ans =

    50


ans =

     0
```

ftell則會回傳現在的游標位置，因此使用fseek移動位置後ftell就改變了。

寫、讀檔指令

結果=fseek(開啟檔案之編號,游標移動的距離,游標起使位置)
若指令執行成功，則**結果**會得到0，反之失敗時，會得到-1。
開啟檔案之編號為fopen的結果
游標移動的距離則是根據游標起使位置改變。



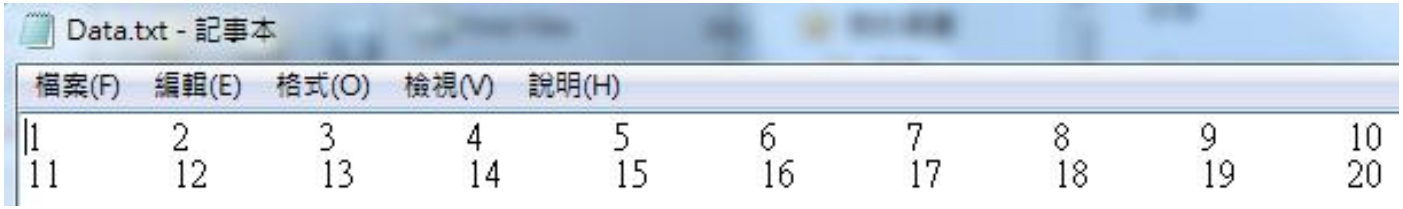
游標起使位置則以-1為檔案開頭，0為當前位置，-1為檔案最後。
前面使用的fseek(fid,0,-1)表示將游標移動至：
fid開啟的檔案的開頭位置且額外移動0的距離
所以才會使游標回到檔案開頭的位置。

寫、讀檔指令

讀寫檔案時需要注意的地方是特殊符號也會包含在一個字中，所以移動幾個位置是以幾個字計算，若存在空白、Tab('\t')、換行('\n')，都會計算為一個字。
*要特別注意換行的使用，在windows的作業系統時，是以'\r\n'兩個字來完成的。

這是使用'\n'換行的結果，利用文字文件開啟後並沒有換行。

使用'\r\n'後，才有真正的使10、11之間換行。



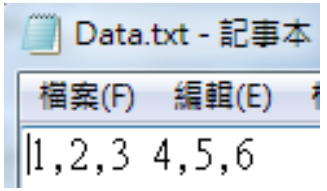
寫、讀檔指令

dlmwrite(檔案路徑,資料,間隔方式,R,C)
為利用特定間隔方式將資料儲存的指令，可以不使用fopen的指令來達成寫檔案的動作。*也因此不需要fclose
檔案路徑為需要存檔的檔案位置。
資料則是將要寫入的數值或文字。
間隔方式則是決定以矩陣儲存的每個元素之間的間隔方式，若未填入時則預設為’,’（逗號為間隔方式）。
R、C則是決定開始寫檔的位置要平移多少距離，並且R是以列的方式移動，C則是以行的方式移動。

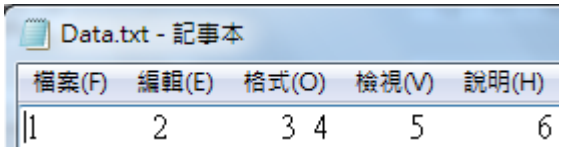
寫、讀檔指令

*若是以行區隔的數值會以間隔方式隔開，但是列區隔的數值會用'\n'的換行方式間隔。

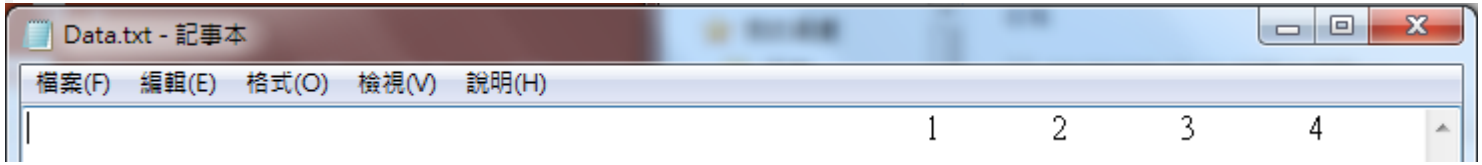
```
>> dlmwrite(save_path,[1 2 3;4 5 6]
```




```
>> dlmwrite(save_path,[1 2 3;4 5 6],'\t')
```




```
>> dlmwrite(save_path,[1 2 3;4 5 6],'\t',2,1)
```



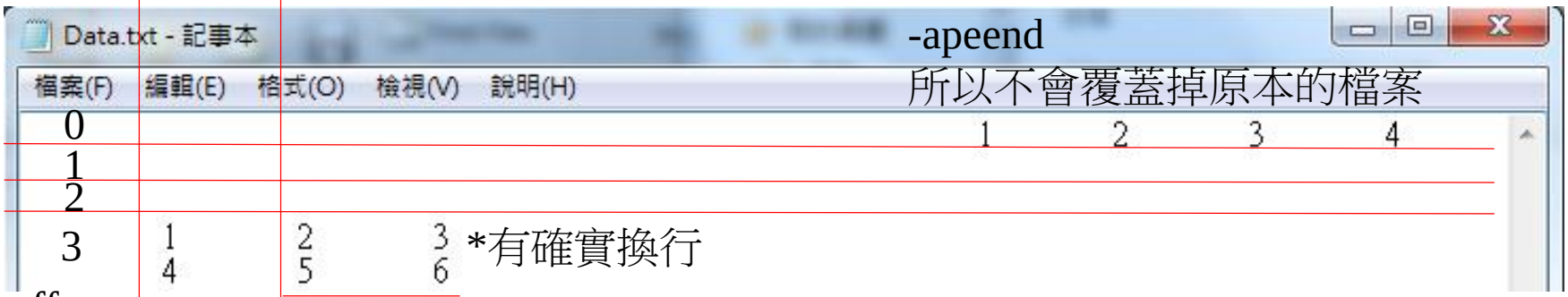
寫、讀檔指令

若是需要使用其他的設定時，則形式與前面將不同
每種設定前需要用文字來指定
dlmwrite(檔案路徑,資料,指定格式敘述,指定格式)

```
dlmwrite(save_path,[1 2 3;4 5 6],'delimiter','\t','newline','pc','-append','roffset',3,'coffset',1)
```

將換行的方式改為windows的格式

coffset
0 1



-apeend
所以不會覆蓋掉原本的檔案

roffset

間隔為Tab('\t')

寫、讀檔指令

dlmread(檔案路徑,間隔方式,R,C)
為利用特定間隔方式將資料讀入，一樣不需fopen的指令。

```
>> dlmwrite(save_path,[1 2 3;4 5 6;7 8 9],'\t')
>> dlmread(save_path,'\t')

ans =

     1     2     3
     4     5     6
     7     8     9
```

```
>> dlmread(save_path,'\t',1,0)

ans =

     4     5     6
     7     8     9
```

```
>> dlmread(save_path,'\t',0,2)

ans =

     3
     6
     9
```

寫、讀檔指令

dlmread(檔案路徑,間隔方式,[R1,C1,R2,C2])
如果讀取的檔案也要限制結束的位置，則要改由矩陣的形式輸入。

```
>> dlmread(save_path,'\t',[0 0 1 1])


ans =


     1     2
     4     5
```

迴圈、條件選擇-作業

費式數列:

1+1+2+3+5+8+13+……+1346269

1.請問 1346269 為第幾項？

2.請問加到此項目時總和為多少？

繪圖-作業

1.將下列位置座標輸入至矩陣中，並繪製成圖（每一個位置點間隔時間為0.5秒）

100.0000 98.7750 95.1000 88.9750 80.4000 69.3750 55.9000 39.9750 21.6000 0.7750

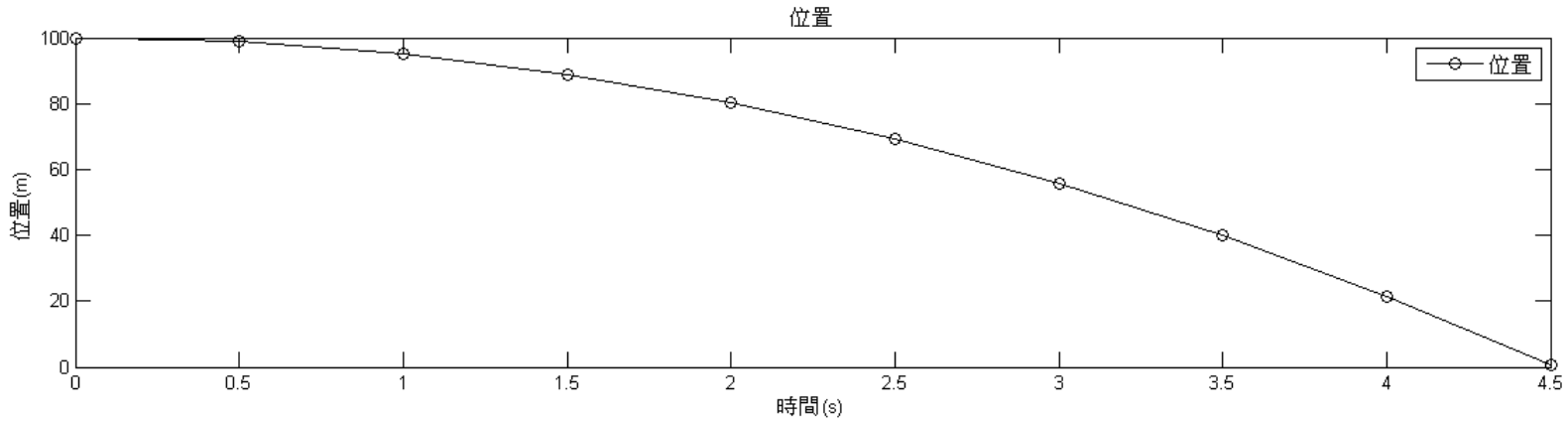
2.計算$\frac{\text{後來的位置}-\text{原來的位置}}{\text{後來的時間}-\text{原來的時間}}=\frac{\text{後來的時間}-\text{原來的時間}}{2}$的速度並繪製成圖

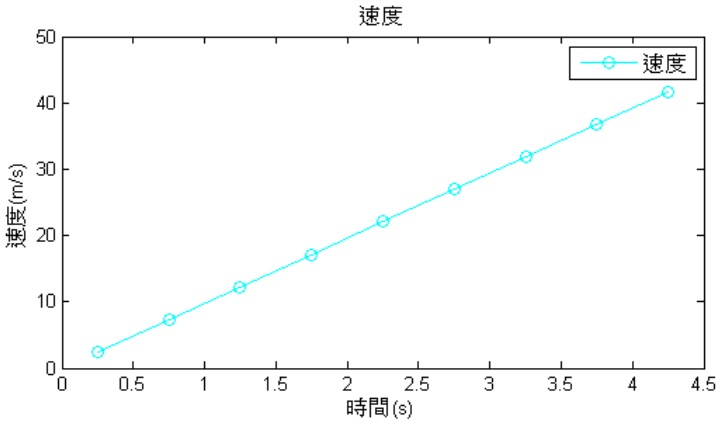
3.利用相同的方式計算加速度並繪製成圖

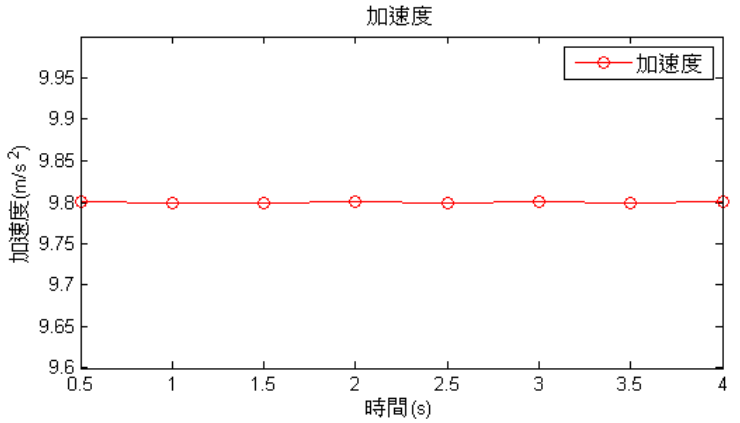
4.將三張圖合併至一起如下一頁的圖

***以上皆須加入圖名、圖例、座標軸之說明**
***速度、加速度繪圖時需取絕對值**

繪圖-作業


位置
位置
位置(m)
時間(s)
0
20
40
60
80
100
0
0.5
1
1.5
2
2.5
3
3.5
4
4.5


速度
速度
速度(m/s)
時間(s)
0
10
20
30
40
50
0
0.5
1
1.5
2
2.5
3
3.5
4
4.5


加速度
加速度
加速度(m/s 2)
時間(s)
9.6
9.65
9.7
9.75
9.8
9.85
9.9
9.95
0.5
1
1.5
2
2.5
3
3.5
4

讀檔、寫檔-作業

1. 將繪圖題之位置寫入檔案中

2. 將繪圖題之速度寫入檔案中

3將繪圖題之加速度寫入檔案中

進階挑戰題-寫檔

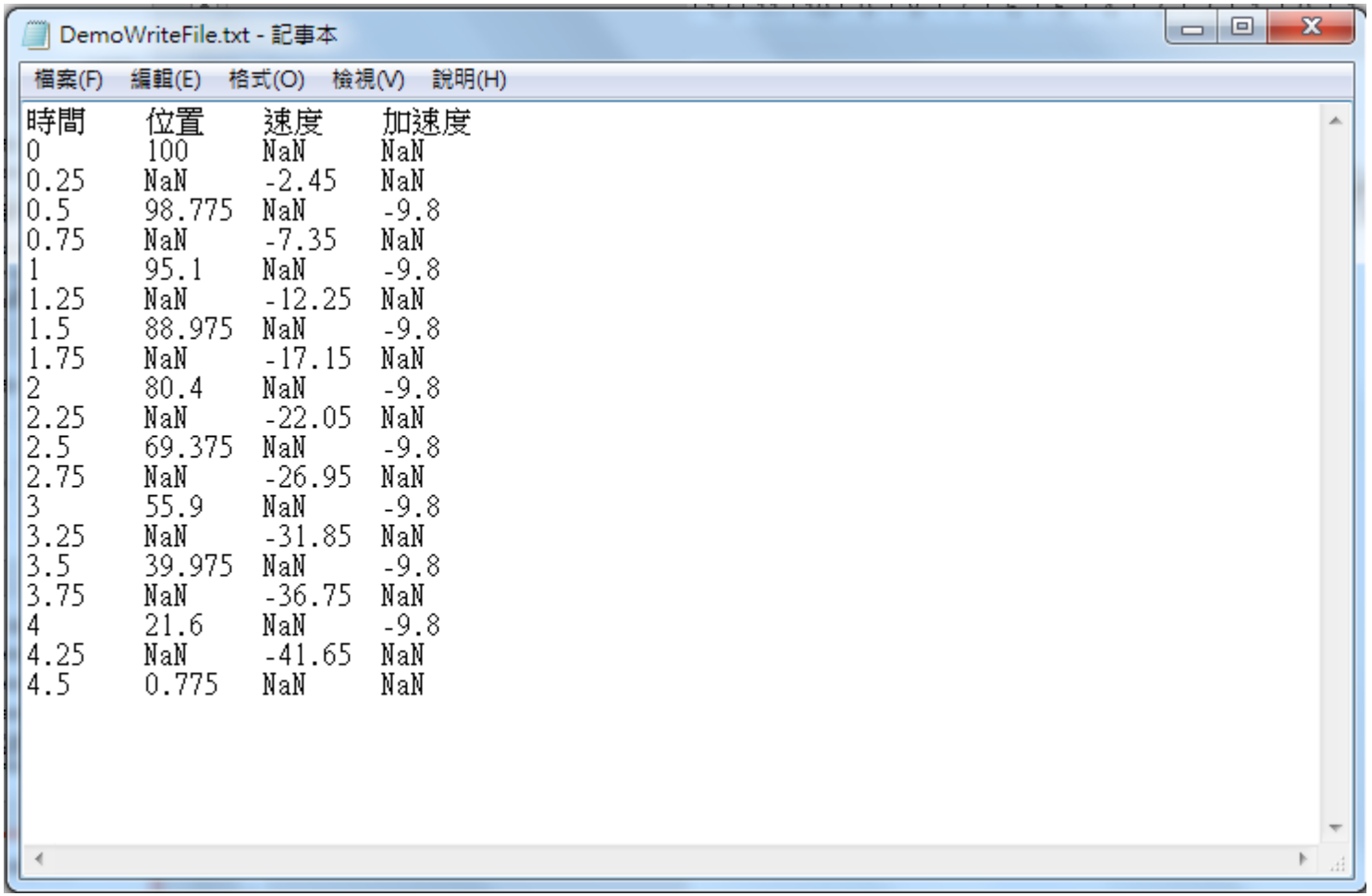
將讀寫檔案的位置、速度、加速度對時間的關係寫入檔案中，若無對應時間的資料時，寫入NaN，如下一頁之結果

*建立矩陣時，可將要儲存的值輸入NaN

```
>> a=[1 2 NaN 3]
```

```
a =

     1     2   NaN     3
```

進階挑戰題-寫檔

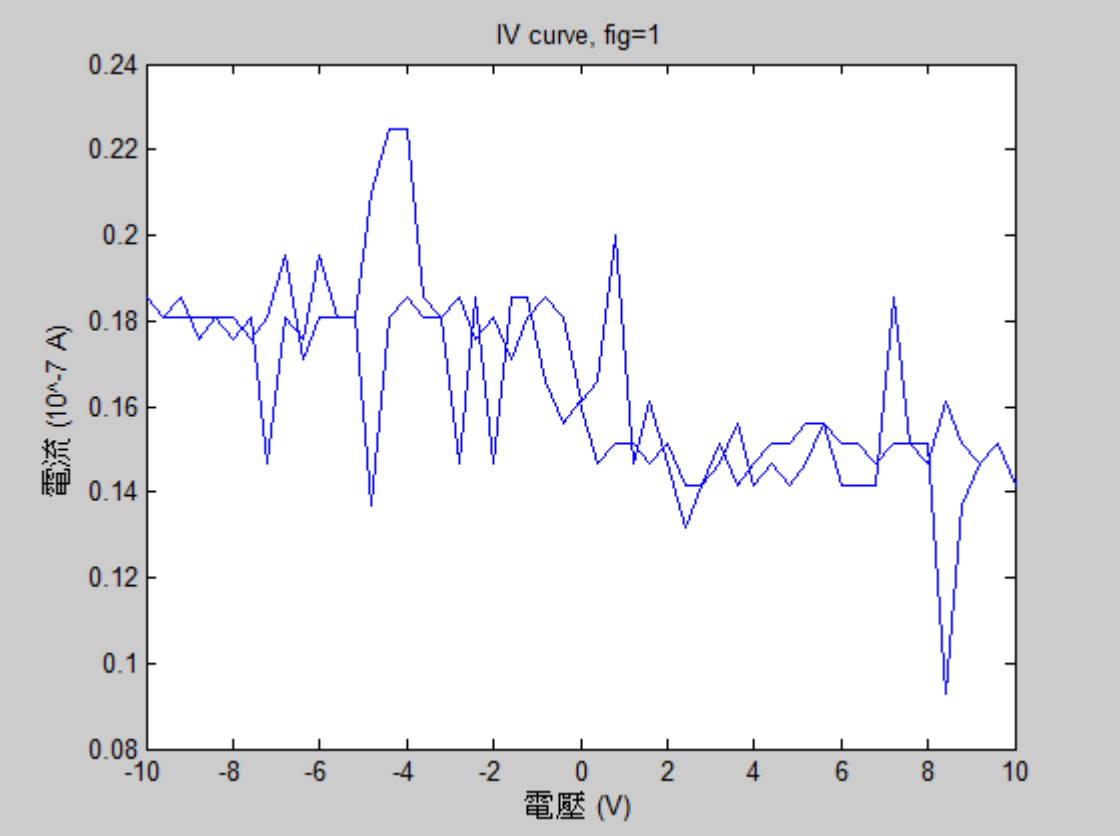

DemoWriteFile.txt - 記事本
檔案(F) 編輯(E) 格式(O) 檢視(V) 說明(H)
時間 位置 速度 加速度
0 100 NaN NaN
0.25 NaN -2.45 NaN
0.5 98.775 NaN -9.8
0.75 NaN -7.35 NaN
1 95.1 NaN -9.8
1.25 NaN -12.25 NaN
1.5 88.975 NaN -9.8
1.75 NaN -17.15 NaN
2 80.4 NaN -9.8
2.25 NaN -22.05 NaN
2.5 69.375 NaN -9.8
2.75 NaN -26.95 NaN
3 55.9 NaN -9.8
3.25 NaN -31.85 NaN
3.5 39.975 NaN -9.8
3.75 NaN -36.75 NaN
4 21.6 NaN -9.8
4.25 NaN -41.65 NaN
4.5 0.775 NaN NaN

進階挑戰-IV Curve

將各位已經儲存好的檔案(使用LabVIEW儲存的檔案)

```
0.000000        -5.200000       0.180664 0.011853       -4.800000       0.209961 0.020889       -4.400000       0.2
9.600000        0.151367 0.382153       10.000000       0.141602 0.388694       9.600000        0.151367 0.396740
0.180664 0.724121       -4.800000       0.136719 0.731086       -5.200000       0.180664 0.739120       -5.600000
```

將一次完整掃描電壓的資料繪製成圖
並將多次掃描電壓的圖綜合成GIF檔案儲存



SPL
NATIONAL CENTRAL UNIVERSITY

進階挑戰-IV Curve

參考程式碼：

```
clear
clc
read_path='C:\Users\sckaeylin\Desktop\實驗室\高中生參與衛星任務\Matlab\G08.lvm.txt';
Data=dlmread(read_path,'\t');
save_path='C:\Users\sckaeylin\Desktop\實驗室\高中生參與衛星任務\Matlab\IVcurve.gif';
for i=1:size(Data,1)/100;
    pl=plot(Data(1+100*(i-1):100+100*(i-1),2),Data(1+100*(i-1):100+100*(i-1),3));
    xlabel('電壓 (V)')
    ylabel('電流 (10\^-7 A)')
    title(['IV curve, fig=' num2str(i)])
    drawnow
    frame=getframe(1);
    [A,map]=rgb2ind(frame2im(frame),256);
    if i==1
        imwrite(A,map,save_path,'gif','LoopCount',Inf,'DelayTime',0.1);
    else
        imwrite(A,map,save_path,'gif','WriteMode','append','DelayTime',0.1);
    end
end
```

加入圖標說明

即時繪圖且將plot得到的圖案轉換成可以輸出圖案檔案的資料

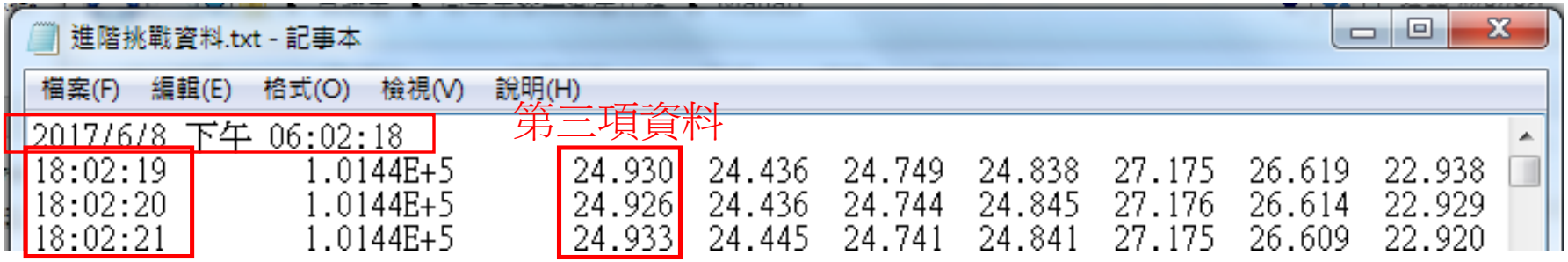
輸出GIF檔

進階挑戰題-資料處理

讀取附件的檔案（請至網站上下載）：進階挑戰資料.txt

檔案標題時間

第三項資料

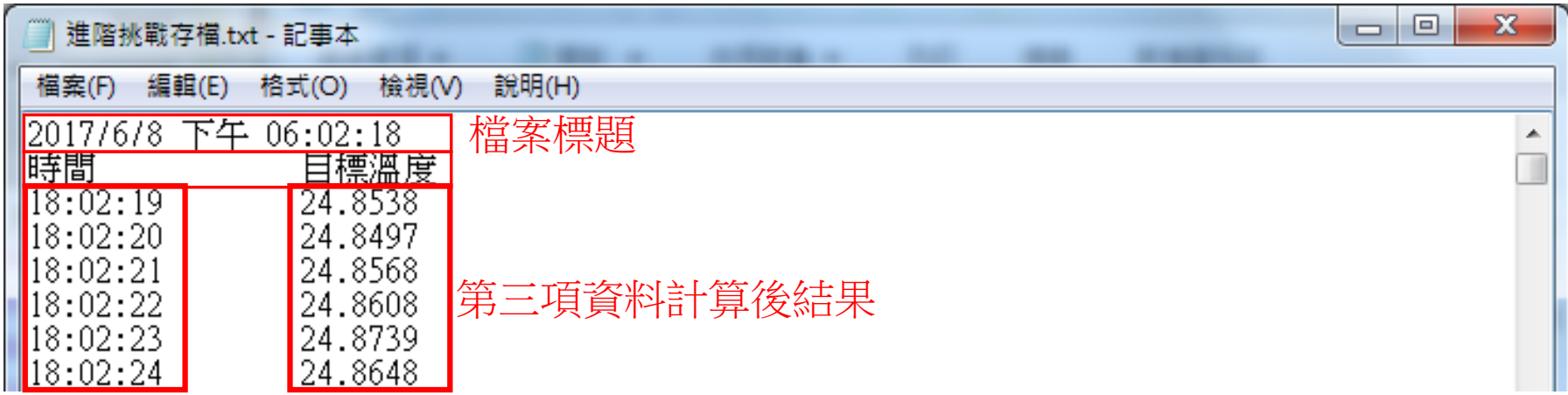


選擇第三項資料處理，$\frac{\text{第三項資料}-0.23}{0.99381294964}$

得到新的資料後，將新的資料寫入，寫入時檔案標題時間保留，並將第二項改為新資料，第三項以後刪除。時間與資料間隔使用Tab。且檔案標題與下方資料要再增加標籤說明，時間、目標溫度，之間間隔需要與下方資料對齊。

進階挑戰題-資料處理

寫入的檔案就如下面所示：



檔案標題

加入標籤

第三項資料計算後結果

時間保留紀錄

進階挑戰題-資料處理

除了寫入檔案以外還需要繪製圖，將x軸為時間變化，y軸為量測的第二項資料，以及經過計算後的第三項資料，利用subplot繪製成上下兩張圖。

第二項資料需要使用semilogy繪製，皆須加入圖名、圖例、座標說明並調整繪製範圍。且繪製虛線標示特定數值。

第二項資料

圖名：Pressure, Start time=star_time

*star_time為進階挑戰資料內的“檔案標題”

x座標：time (s)

y座標：Pressure (pa)

圖例：pressure

繪製範圍：x軸範圍 $0\sim5\times10^{5}$

y軸範圍 $10^{-5}\sim2\times10^{5}$

進階挑戰題-資料處理

第三項資料

圖名：Thermocouple Temparature, Start time=star_time

*star_time為進階挑戰資料的檔案標題

x座標：time (s)

y座標：temparature (℃)

圖例：目標溫度

繪製範圍：x軸範圍 $0\sim5\times10^{5}$

y軸範圍 -60～90

虛線繪製

第二項資料：y軸 5×10^{-3} 繪製虛線

第三項資料：y軸-40.5、-39.5、-30.5、-29.5、59.5、60.5、69.5、70.5 繪製虛線

進階挑戰題-資料處理

最後必須在程式前後加上tic、toc指令

這兩個指令可以達成計時的功能，因此最後的挑戰題會需要加上這兩個指令來得到執行完整體程式的時間。

如果需要建立相同值特定大小的矩陣，可以使用

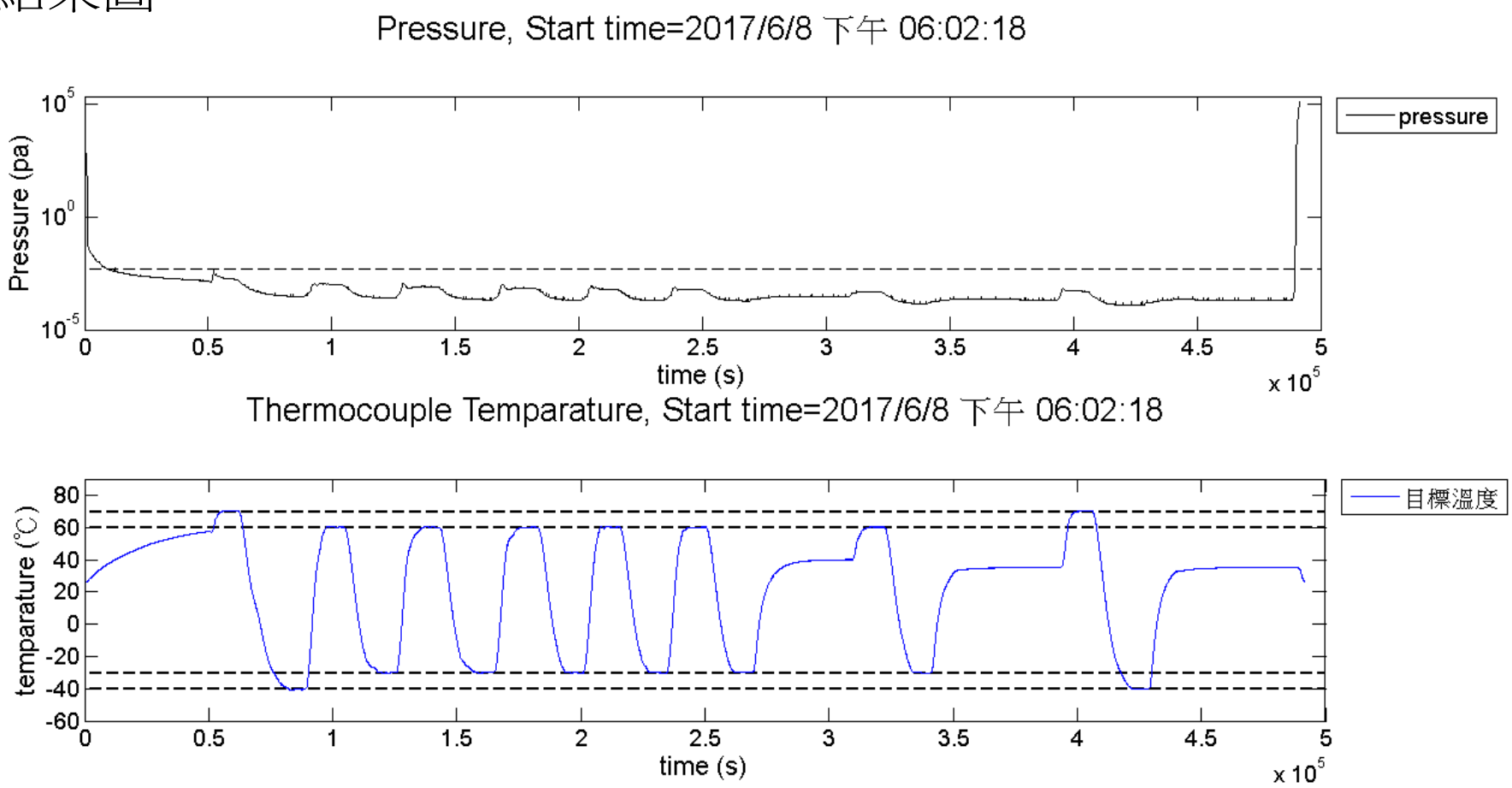
ones(列數量,行數量) 可建立數值皆1的矩陣

zeros(列數量,行數量) 可建立數值皆0的矩陣

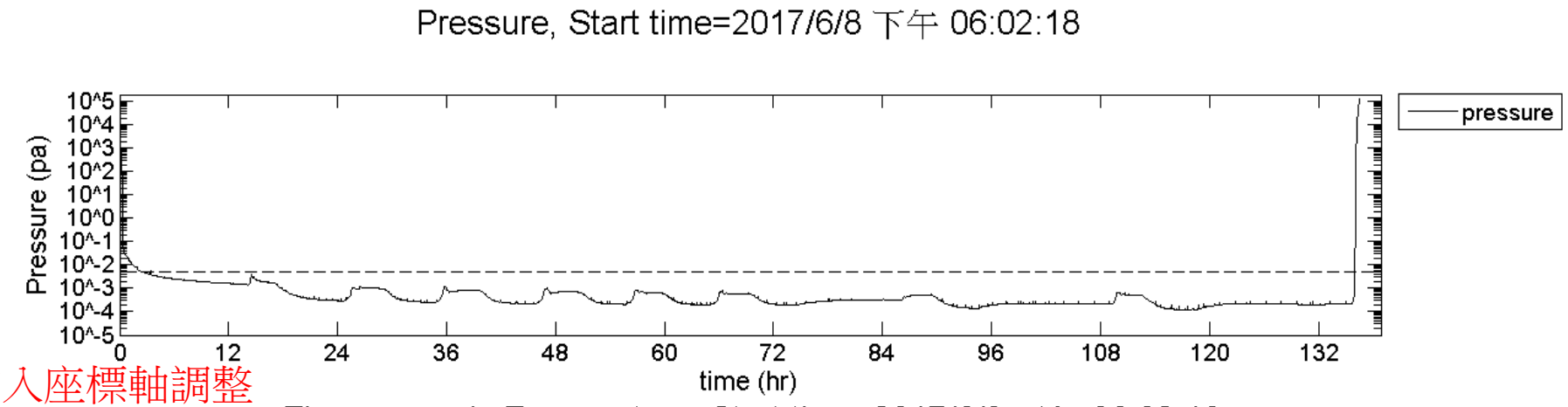
```
>> tic
ones(2,3)
toc

ans =

     1     1     1
     1     1     1

Elapsed time is 0.004963 seconds.
```

進階挑戰題-資料處理

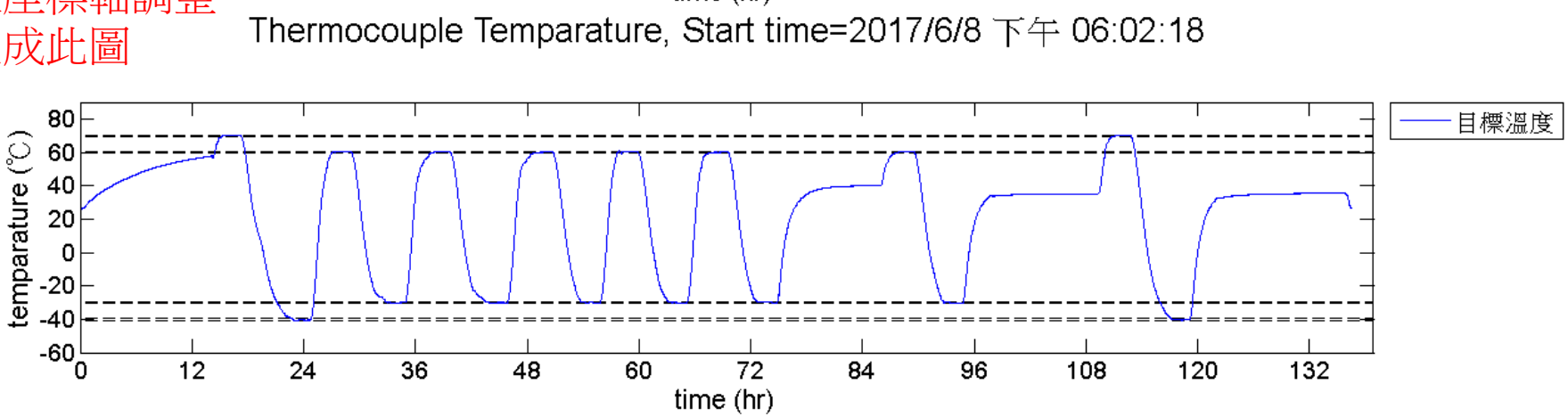
結果圖：



進階挑戰題-資料處理



若加入座標軸調整
則可完成此圖



NATIONAL CENTRAL UNIVERSITY