第八章撰寫底稿與函數

本章學習目標

- 認識M檔案
- 學習撰寫底稿與函數
- 學習偵錯的技巧
- 學習如何使用全域變數
- 學習Matlab搜尋M檔案的方式

1

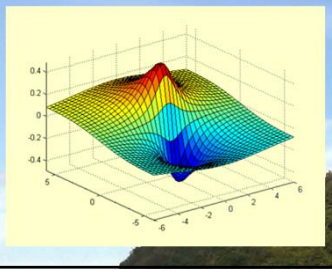
撰寫底稿

- 儲存Matlab程式碼的檔案稱為M檔案(附加檔名是 .m)
- 底稿 (script) 是M檔案的一種，它是由一系列的敘述組成

在指令視窗裡一行一行的鍵入，如此較不方便

在下一頁中，我們將把它撰寫成M檔案

```
>> x=linspace(-6,6,36);
>> y=linspace(-6,6,36);
>> [xx yy]=meshgrid(x,y);
>> zz=yy./(xx.^2+yy.^2+1);
>> surf(xx,yy,zz); axis tight
```



2

使用M檔案編輯器

- 在指令視窗裡鍵入 edit 開啟M檔案編輯器

```
% script8_1.m, 底稿練習-繪出三維函數圖
clear                                   % 清除工作區內的所有變數
x=linspace(-6,6,36);                    % 建立向量x
y=linspace(-6,6,36);                    % 建立向量y
[xx,yy]=meshgrid(x,y);
zz=yy./(xx.^2+yy.^2+1);
surf(xx,yy,zz); axis tight              % 繪出z=y/(x^2+y^2+1)的三維圖形
```

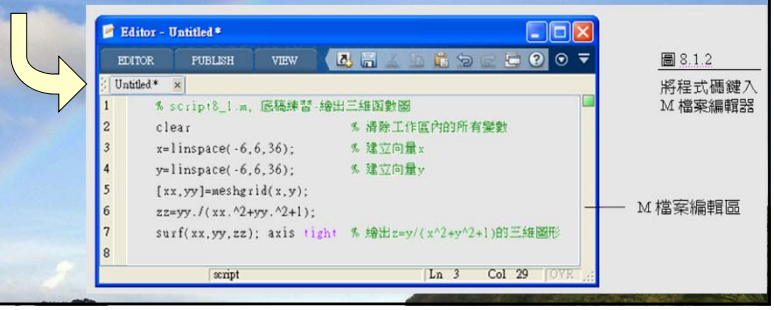


圖 8.1.2 將程式碼鍵入 M 檔案編輯器

儲存M檔案

- M檔案預設存放在「目前工作目錄」(current directory)中
- 本書把「目前工作目錄」設在C:\work裡

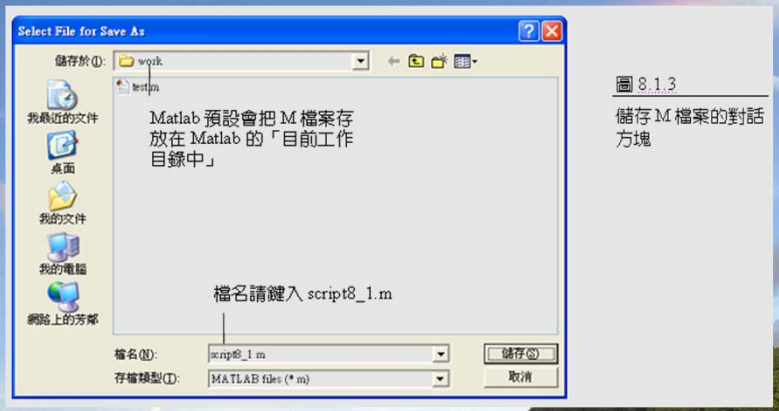
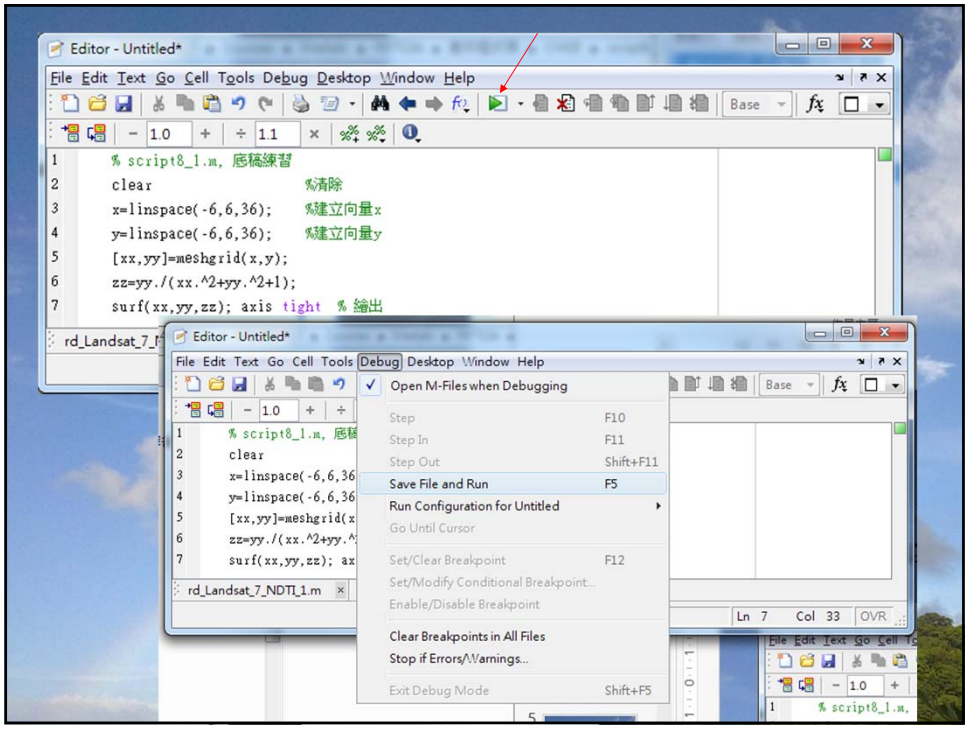


圖 8.1.3 儲存 M 檔案的對話方塊



執行M檔案

- 於指令視窗中鍵入底稿名稱 script8_1 ，即可執行它
  >> script8_1
- 或是按下F5鍵也可以執行底稿

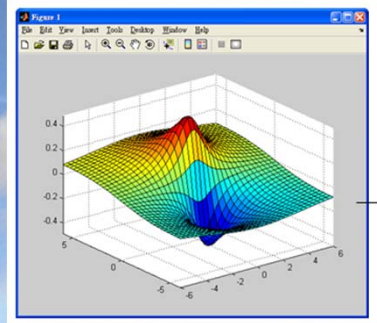


圖 8.1.4
執行底稿 script8_1.m 的結果

執行底稿 script8_1，即可繪出 $f(x,y)=y/(x^2+y^2+1)$ 的三維函數圖

設計Matlab的函數

- 函數（function）也是M檔案的一種，它可用來完成某個特定的工作
- 函數可以接收引數，也可以把運算結果傳回工作區
- 在函數內使用的變數是區域變數（local variable），因此即使工作區內已使用相同名稱的變數，它們彼此之間還是不會混淆

函數的基本架構

- 一個完整的Matlab函數包括函數定義列、H1列（唸成H-one line）、函數說明文字區，以及函數的主體四個部分

```
function 輸出變數 = 函數名稱(引數 1,引數 2,…)   — 函數定義列
%H1 列，此行可用來簡述函數的功用                — H1 列
%此區是函數的說明文字，可用來註解               — 函數說明文字區
%函數的語法、注意事項等
函數的主體                                      — 函數的主體
```

圖 8.2.1 函數架構圖

指令	說明
type function_name.m	查閱M檔案function_name的內容
which function_name.m	查閱存放M檔案function_name的資料夾

查閱M檔案的內容

```
>> type linspace.m
```

```
function y = linspace(d1, d2, n)        ——— 函數定義列
%LINSPACE Linearly spaced vector.       ——— H1 列
%   LINSPACE(X1, X2) generates a row vector of 100 linearly
%   equally spaced points between X1 and X2.
%
%   LINSPACE(X1, X2, N) generates N points between X1 and X2.
%   For N < 2, LINSPACE returns X2.
%
%   Class support for inputs X1,X2:
%      float: double, single
%
%   See also LOGSPACE, COLON.
```

函數說明文字區

```
%   Copyright 1984-2011 The MathWorks, Inc.
%   $Revision: 5.12.4.7 $  $Date: 2011/12/16 16:32:58 $

if nargin == 2
    n = 100;
end
n = double(n);
... (後面的程式碼略去)
```

函數的主體

9

簡單的範例 func8_1()

```
function total=func8_1(x,y)
%FUNC8_1 sum of two numbers or vectors.
%FUNC8_1(X,Y) computes X+Y and returns the result.
%X and Y can be scalars or vectors.
%function's body starts here
total=x+y;
```

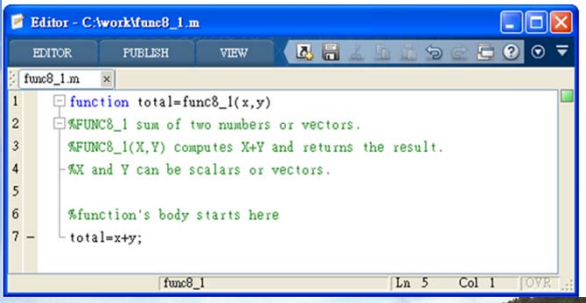


圖 8.2.2
編輯完成的函數 func8_1()

```
>> func8_1(3,5)
ans =
     8
```

10

試試查詢的功能

- 在Matlab的指令視窗裡輸入

```
>> help func8_1
```

Matlab會回應

```
func8_1 sum of two numbers or vectors.
func8_1(X,Y) computes X+Y and returns the result.
X and Y can be scalars or vectors.
```

這三行文字正是在程式碼 func8_1.m 裡2~4行的註解

11

函數的引數與傳回值

```
function total = func8_1(x,y)
```

輸出引數（傳回值）　函數名稱　輸入引數

圖 8.2.3
函數的輸入引數與輸出引數

表 8.2.2　函數定義列的幾種範例

函數定義列的格式	說 明
`function [x,y]=myfun(a)`	**myfun** 有一個輸入引數 *a*，有兩個輸出引數 *x* 與 *y*
`function [x]=myfun(a)` `function x=myfun(a)`	**myfun** 有一個輸入引數 *a*，有一個輸出引數 *x*
`function [x,y]=myfun()` `function [x,y]=myfun`	**myfun** 沒有輸入引數，但有兩個輸出引數 *x* 與 *y*。在沒有輸入引數的情況下，函數名稱後面的括號可有可無
`function []=myfun(a)` `function myfun(a)`	**myfun** 沒有輸出引數，但有一個輸入引數 *a*。當函數沒有輸出引數時，方括號與等號可以省略不寫

有兩個傳回值的函數

func8_2() 可接收一個向量*v*，並可傳回*mn*與*mx*，分別代表向量*v*的最小值與最大值

```
function [mn,mx]=func8_2(v)
mn=min(v);
mx=max(v);
```

```
>> [x,y]=func8_2([8 7 3 9 1])
x =
     1
y =
     9
```

```
>> vec=func8_2([8 7 3 9 1])
vec =
     1
```

```
>> func8_2([8 7 3 9 1])
ans =
     1
```

13

不需傳入引數的函數

定義函數func8_3()。因func8_3()沒有任何引數，所以func8_3()後面的括號內保留空白即可。事實上，這個括號也可以省略。

```
function num=func8_3()
num=length(primes(100));
```

```
>> func8_3()
ans =
    25
```

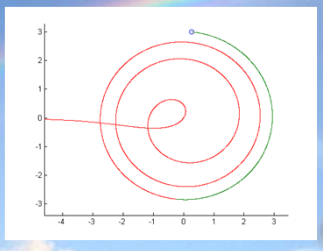
```
>> func8_3
ans =
    25
```

14

沒有傳回值的函數

```
function func8_4(n)
t=linspace(0.01,10*pi,n);
r=log(t);
comet(r.*cos(t),r.*sin(t));
```

```
>> func8_4(10000)
```



因為函數func8_4() 沒有傳回值，所以函數定義列不需要寫上輸出引數與等號。

```
>> x=func8_4(10000)
Error using func8_4
Too many output arguments.
```

追蹤函數的執行與偵錯 (1/2)

- 在函數執行時列印訊息

```
function func8_5(n)
if mod(n,2)==0
    fprintf('%d is even\n',n);
else
    fprintf('%d is odd\n',n);
end
```

在函數內安插一些敘述來列印程式裡的部份訊息

```
>> func8_5(14)
14 is even
```

```
>> func8_5(63)
63 is odd
```

16

追蹤函數的執行與偵錯 (2/2)

- Matlab的M檔案偵錯環境

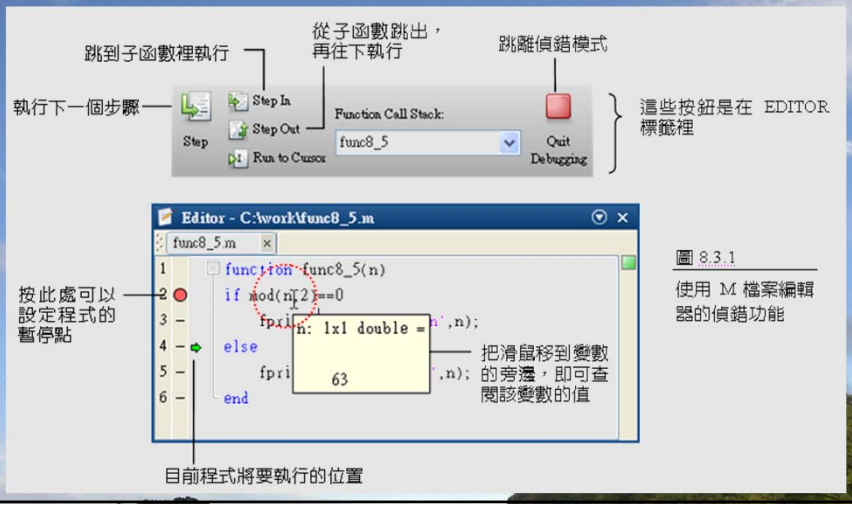


圖 8.3.1 使用 M 檔案編輯器的偵錯功能

函數的進階-不需括號的呼叫

```
axis([-4,4,0,20]);      % 引數有括號
axis off;               % 引數沒有括號
axis on;                % 引數沒有括號
```

- 函數如果沒有輸出引數，就不需要將輸入的引數括起來
- 例如，如果 my_func(*a*, *b*) 沒有輸出引數，可用下面兩種方法呼叫：

```
my_func(a,b);     % 需要括號的呼叫方式
my_func a b;      % 不需括號的呼叫方式
```

18

不需括號的呼叫的範例

定義函數func8_6()，它可接收一個字串，然後把該字串印出。注意func8_6()沒有輸出引數。

```
function func8_6(str)
fprintf('You input
''%s''.\n',str)
```

```
>> func8_6('sss')
You input 'sss'.
```

```
>> func8_6 'sss'
You input 'sss'.
```

```
>> func8_6 sss
You input 'sss'.
```

19

函數的進階-引數的個數

- plot(*y*) % 只有一個輸入引數，可繪出*x*間距為1的*x*-*y*二維圖形
- plot(*x*,*y*) % 有兩個輸入引數，可繪出*x*-*y*二維圖形
- plot(*x*1,*y*1,*x*2,*y*2) % 有四個輸入引數，可繪出*x*1-*y*1與*x*2-*y*2兩個二維圖形
- *zz*=peaks; % 不需輸入引數，以一個變數接收輸出引數
- [*xx*,*yy*,*zz*]=peaks(*n*) % 有一個輸入引數，以三個變數接收輸出引數

表 8.4.1 nargin 與 nargout 變數

變數名稱	說 明
nargin	函數裡輸入引數的個數
nargout	函數裡輸出引數的個數

nargin是number of argument input的縮寫

nargout是number of argument output的縮寫

20

nargin與nargout的使用範例

nargin	函數裡輸入引數的個數
nargout	函數裡輸出引數的個數

```
function [x1,x2,x3]=func8_7(a1,a2)
fprintf('nargin = %d, ',nargin)
fprintf('nargout= %d\n',nargout)
x1=a1+a2;
x2=a1-a2;
x3=(a1+a2)/2;
```

```
>> [x,y,z]=func8_7(6,12)
nargin = 2, nargout= 3
x =
   18
y =
   -6
z =
    9
```

```
>> [x,y]=func8_7(6,12)
nargin = 2, nargout= 2
x =
    18
y =
    -6
```

```
>> total=func8_7(6,12)
nargin = 2, nargout= 1
total =
    18
```

```
>> total=func8_7(6)
nargin = 1, nargout= 1
Error using func8_7 (line 4)
Not enough input arguments.
```

- Func8_7([5 8 3],[3 2 1]), 有幾個輸入引數和輸出引數?結果為何?
- [x,y,z]=Func8_7([5 8 3],[3 2 1]), 有幾個輸入引數和輸出引數?結果為何?

22

nargin與nargout的應用(1/2)

可以依據nargin與nargout的值，適時的在函數裡加入某些判斷的敘述，以便讓函數更有彈性

```
function [x1,x2,x3]=func8_8(a1,a2)
if nargin==1
    a2=0;
end
x1=a1+a2;
x2=a1-a2;
x3=(a1+a2)/2;
```

```
>> [x,y,z]=func8_8(6)
x =
     6
y =
     6
z =
     3
```

23

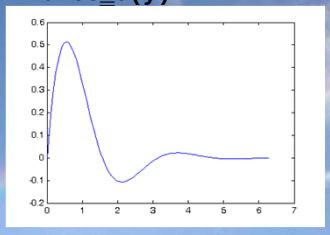
nargin與nargout的應用(2/2)

```
function func8_9(a,b)
if nargin==1
    xx=1:length(a);
    plot(xx,a)
else
    plot(a,b)
end
```

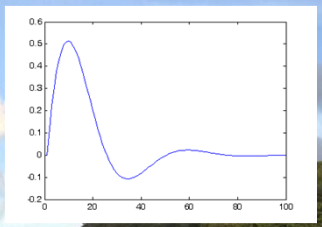
```
>> x=linspace(0,2*pi,100);

>> y=sin(2*x)./exp(x);
```

```
>> func8_9(y)
```



```
>> func8_9(x,y)
```



24

My_main8_9

```
x=linspace(0,2*pi,100);
y=sin(2*x)./exp(x);
func8_9(y)
figure
func8_9(x,y)
```

25

函數內變數的等級

- 在Matlab函數內部的變數均是區域變數(local variable)
- 利用 global 關鍵字可宣告成全域變數

表 8.4.2 使用全域變數

語 法	說 明
global var_1 var_2 …	宣告全域變數 $var_1, var_2, \cdots$
whos global	查詢工作區內的全域變數
clear global var_1 var_2 …	刪除全域變數 $var_1, var_2, \cdots$

26

使用全域變數的範例

```
function func8_10(num)
global VAR;
VAR=VAR+num;
fprintf('在函數內，
VAR=%d\n',VAR);
```

```
>> global VAR
```

```
>> VAR=10;
```

```
>> func8_10(5)
在函數內，VAR=15
```

```
>> func8_10(5)
在函數內，VAR=20
```

```
>> VAR
VAR =
     20
```

```
>> num
Undefined function or variable
'num'.
```

```
>> clear global VAR
```

27

子函數

- 同一個M檔案裡可以撰寫多個函數
- 撰寫在最上方的函數稱為主函數(main function)
- 其它的函數則稱為子函數(sub function)
- 一個M檔案只能有一個主函數，但可以有多個子函數

```
function func8_11(v)  % 主函數func8_11
subf(v);
fprintf('End of main function\n');

function subf(n)        % 子函數subf
fprintf('sum(n)=%d\n',sum(n))
fprintf('prod(n)=%d\n',prod(n))
```

```
>> func8_11([1 2 3 4 5])
sum(n)=15
prod(n)=120
End of main function
```

```
>> subf([1 2 3 4 5])
Undefined function 'subf' for input
arguments of type 'double'.
```

28

私有化目錄(private directory)

1. 在目前主函數所存在的目錄底下再建立一個名稱為private的子目錄
2. 然後把所有想讓主函數呼叫的子函數（每一個子函數存成一個檔案）都存放在這個private子目錄內

- 私有化目錄的好處
  - 不必設定路徑
  - 不用把子函數撰寫在與主函數同一個M檔案內
  - 可利用主函數來呼叫存放在 private 子目錄內的函數

29

建立私有化目錄的範例

```
function func8_12(v) %主函數 func8_12
subf(v);
fprintf('End of func8_12\n')
```

```
function subf(n)   % 子函數 subf
fprintf('sum(n)=%d\n',sum(n))
fprintf('prod(n)=%d\n',prod(n))
```

圖 8.4.1 建立私有化目錄的過程

呼叫func8_12()，此時會自動到private資料夾裡找尋子函數subf() 來執行

```
>> func8_12([1 2 3 4 5])
sum(n)=15
prod(n)=120
End of main function
```

定義在private的子函數並不能由外界直接執行。

```
>> subf([1 2 3 4 5])
Undefined function 'subf' for input arguments of type 'double'.
```

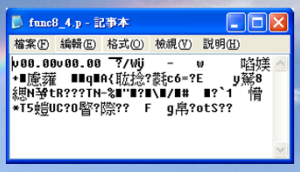
30

保護程式碼—pcode

- pcode 是pseudo-code的縮寫
- pcode 使程式碼可以執行，但無法查閱內容

表 8.4.3 使用 pcode 指令

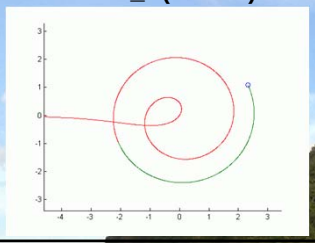
語 法	說 明
pcode *file_name.m*	將 M 檔案轉換成 pcode




```
>> pcode func8_4.m
```

```
>> which func8_4
C:\work\func8_4.p
```

```
>> func8_4(10000)
```



31

設定工作目錄

- Matlab在使用變數，或者是呼叫函數時，都是先在目前工作目錄 (current directory) 先取用函數或變數
- 設定工作目錄 ：

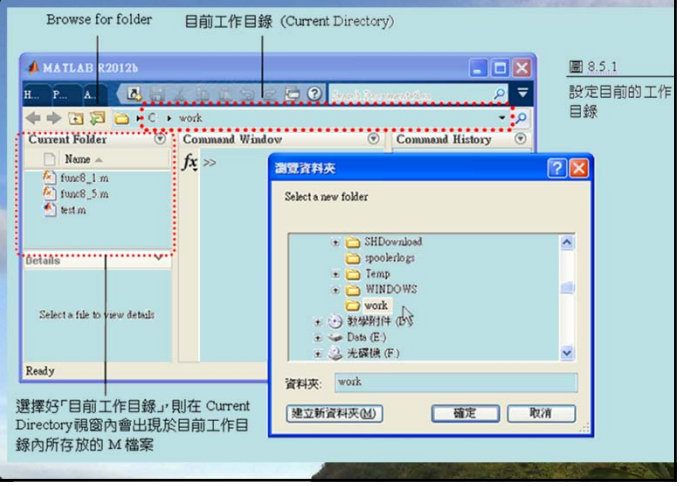


圖 8.5.1 設定目前的工作目錄

設定Matlab搜尋的路徑

圖 8.5.2
Set Path 對話方塊

匿名函數

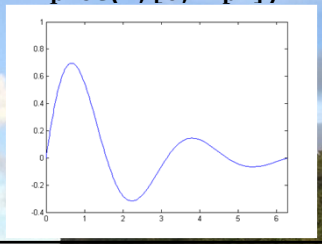
- 匿名函數 (anonymous functions) 可以在指令視窗裡直接定義一個函數，而不用把函數寫在M檔案裡

表 8.6.1 匿名函數的定義

指 令	說 明
fname=@(*arg_list*) *expr*	定義匿名函數，函數名稱為 *fname*，輸入引數為 *arg_list*，函數的內容則定義在 *expr* 的位置

```
>> f=@(x) sin(2*x).*exp(-x/2)
f =
    @(x)sin(2*x).*exp(-x/2)
```

```
>> fplot(f,[0,2*pi])
```

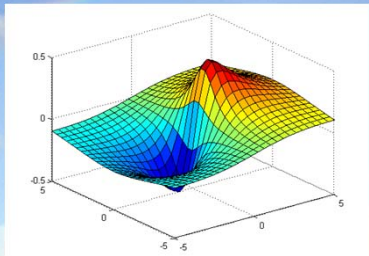


二個引數以上的匿名函數

要定義二個引數以上的匿名函數，只要將每一個引數依序填入匿名函數的括號內即可

```
>> g=@(x,y) x./(x.^2+y.^2+1);
>> [xx,yy]=meshgrid(-5:0.4:5,-5:0.4:5);
>> g(2,3)
ans =
    0.1429
```

```
>> surf(xx,yy,g(xx,yy))
```



35