Lecture 3

Vector representations:

Sounds, Time series

Matrix Representations:

Color Images, Gray Images

sun spot data

Data Analysis for Time Series : interpretation, modeling by mathematical frameworks, forecasting and predicting

Data

sunspot_train

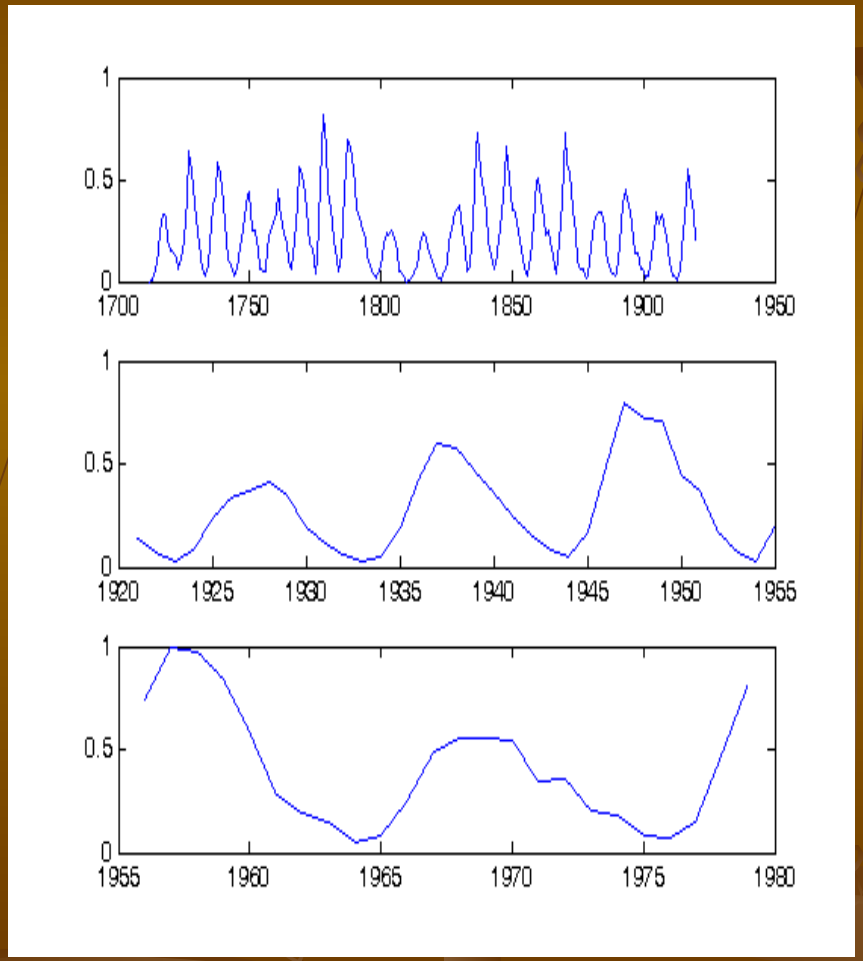
sunspot_test_1.mat

sunspot_test_2.mat

time series: sun spot

Source codes

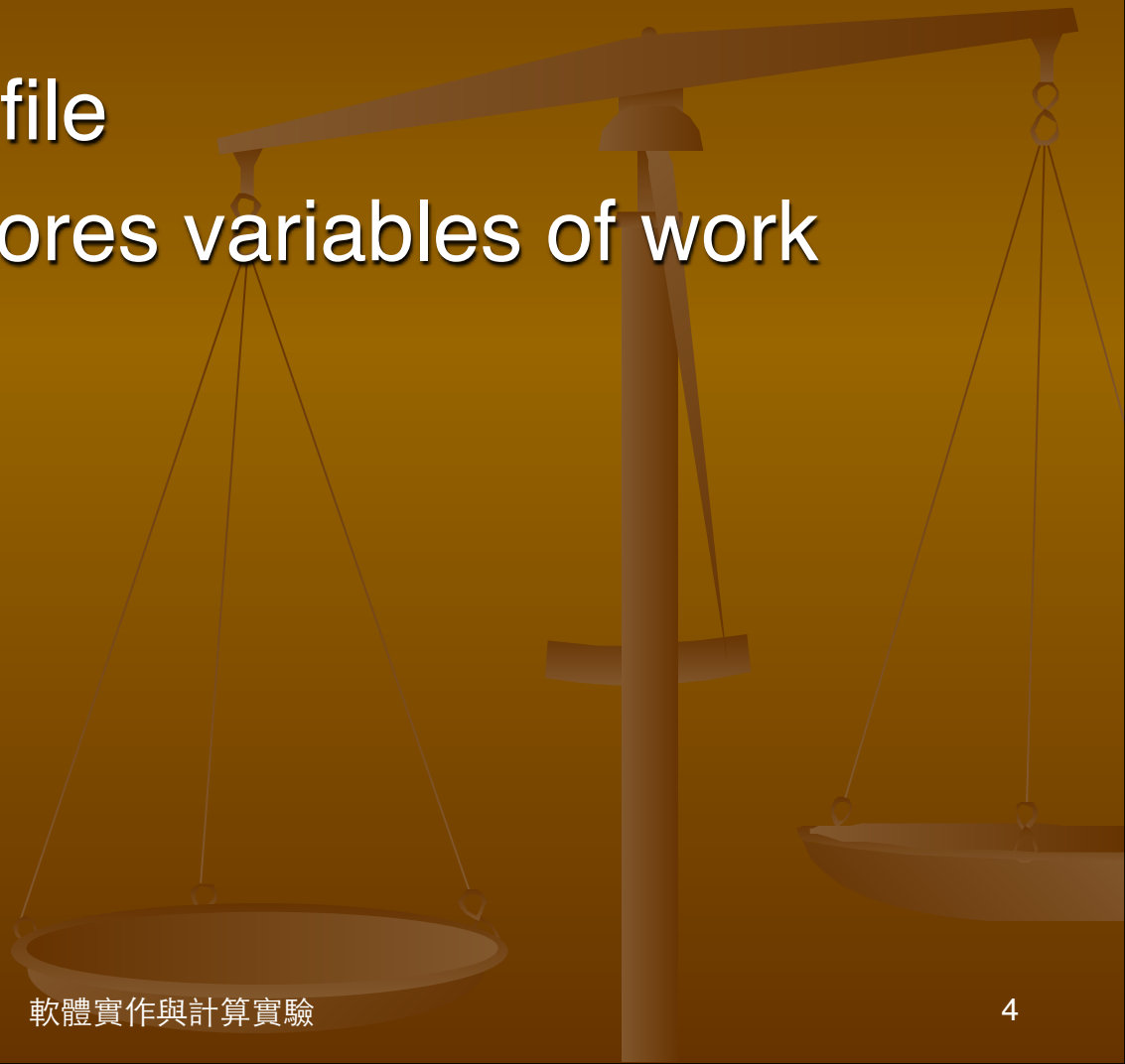
```
load sunspot_train.mat;
subplot(3,1,1);
plot(1712:1920,Y);
load sunspot_test_1.mat;
subplot(3,1,2);
plot(1921:1955,Y);
load sunspot_test_2.mat;
subplot(3,1,3);
plot(1956:1979,Y);
```



load

```
load sunspot_train.mat;
```

- Load an MAT file
- An MAT file stores variables of work space



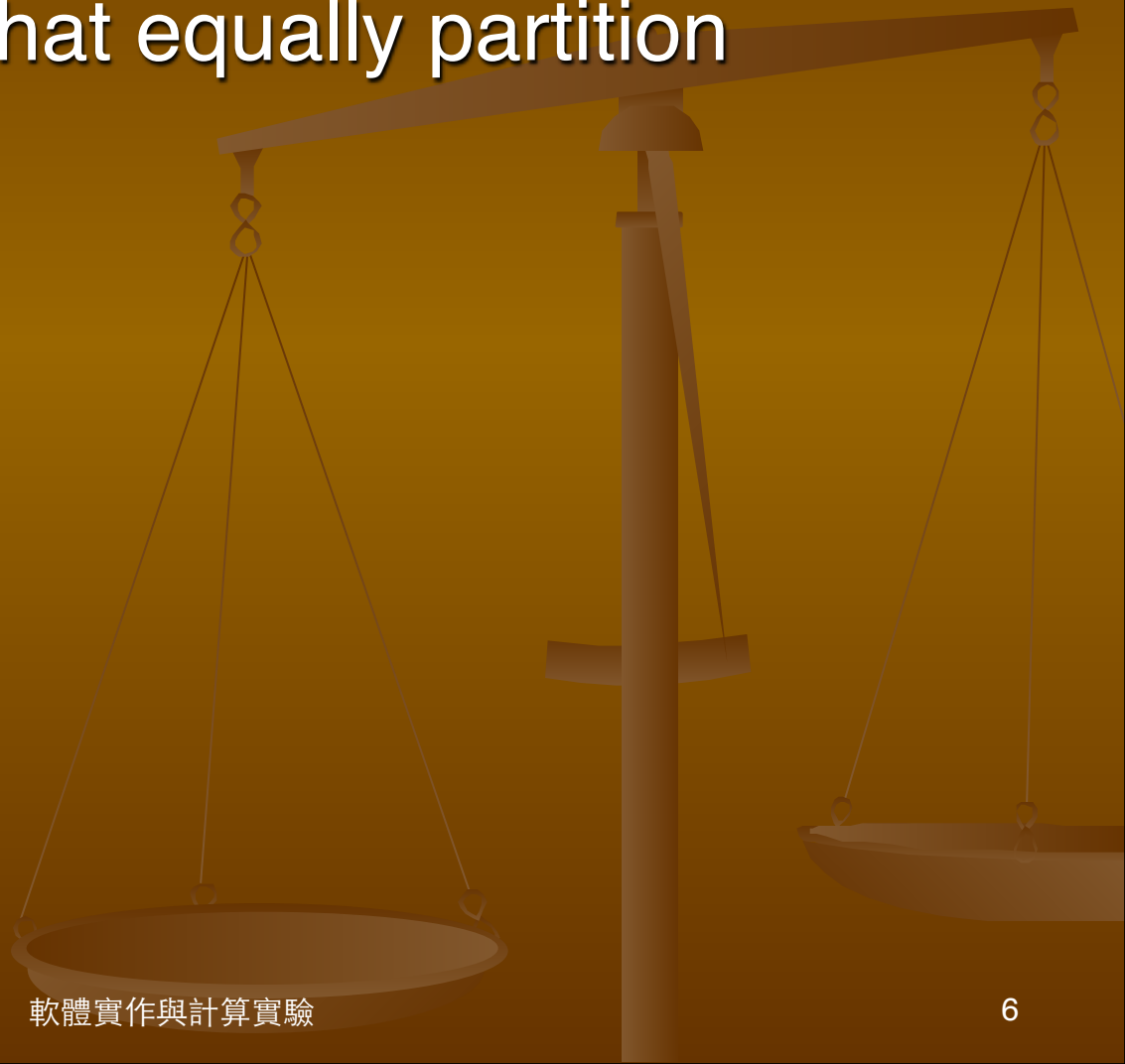
subplot

subplot(m,n,k)

- A figure conists of m rows and n columns of subplots
- Select the the k-th subplot
- Column-major order

Equal Partition: linspace

- Create 5 knots that equally partition interval [0 1]



Sub vector

```
>> a=linspace(0,1,100);
>> a(95:100)

ans =

    0.9495    0.9596    0.9697    0.9798    0.9899    1.0000
```

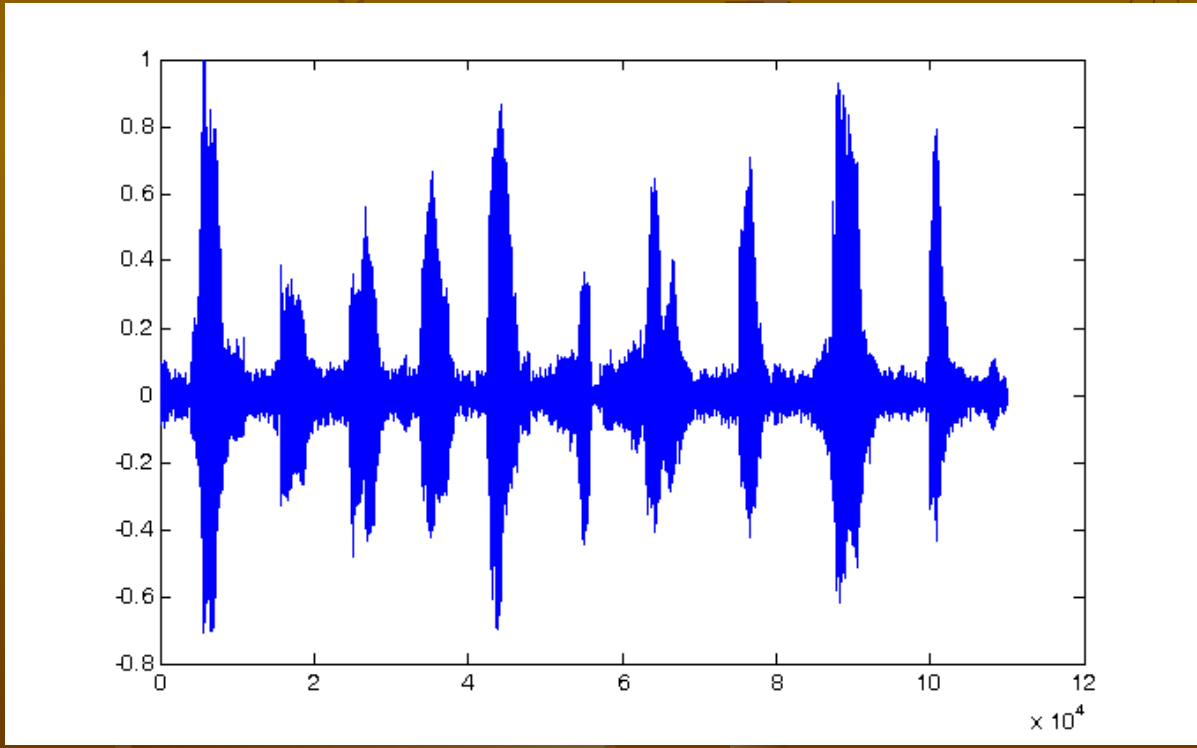
```
>> a=linspace(0,1,100);
>> a(100:-1:95)

ans =

    1.0000    0.9899    0.9798    0.9697    0.9596    0.9495
```

Sound Read

```
[s1,fs]=wavread('ic_a1.wav');
plot(1:length(s1),s1)
```

ic_a1.wav

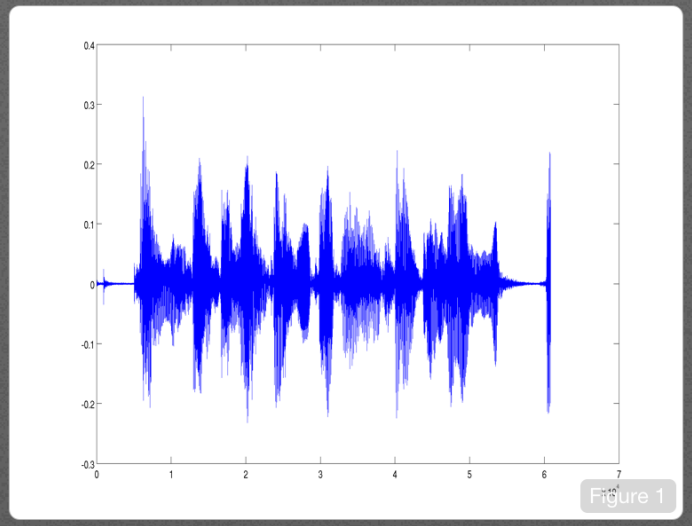
Music and speech



MATLAB Mobile

```
>> dir

.           .session    Shared      song.wav
..          Published   cmw2.bmp

>> [s1,fs]=wavread('song.wav');
plot(1:length(s1),s1)
```

Figures | History




```
[s1,fs]=wavread('song.wav');
plot(1:length(s1),s1);
```

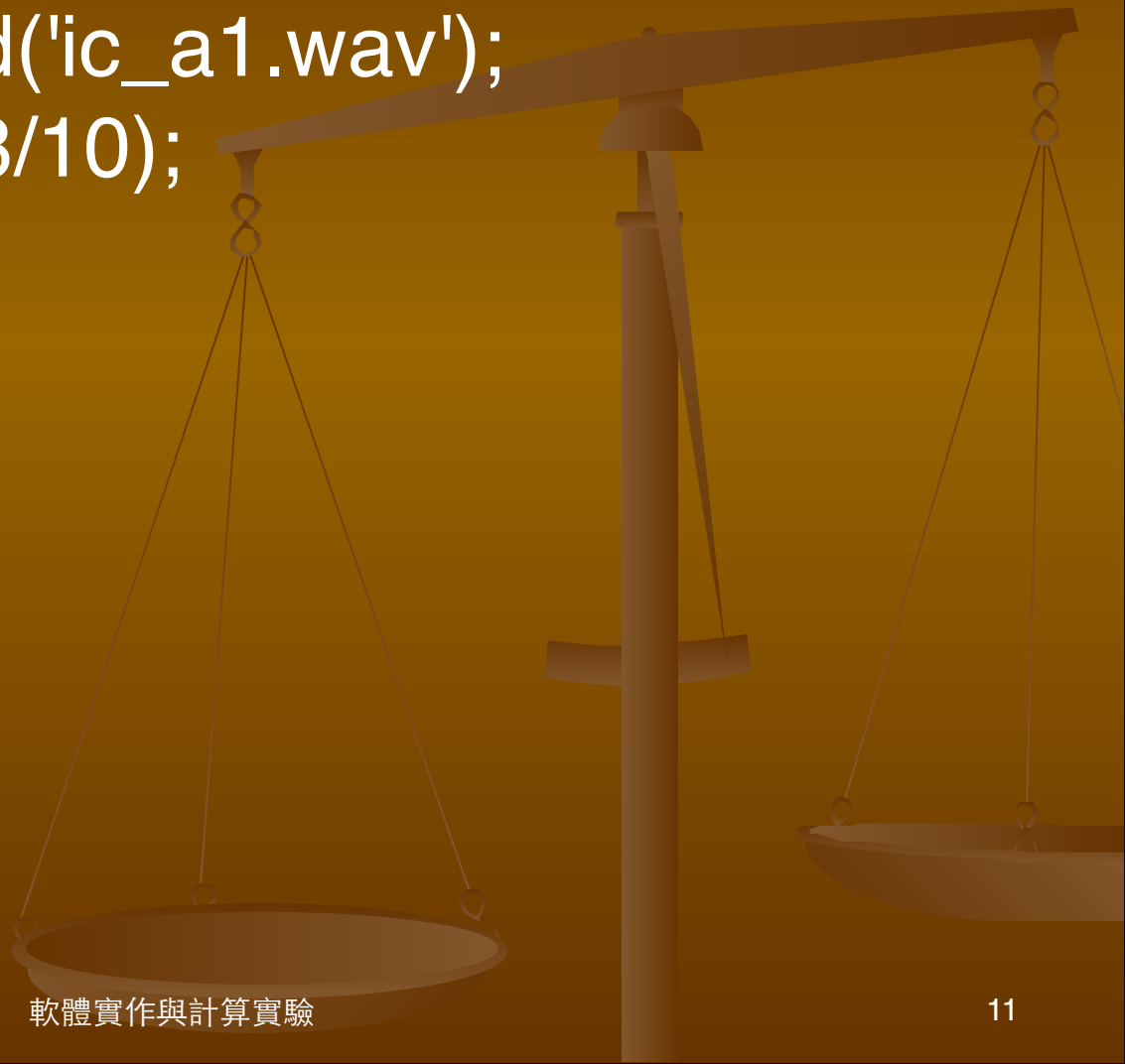
Sound Playing

```
[s1,fs]=wavread('ic_a1.wav');
wavplay(s1,fs);
```

- s1 is a recorded sound vector
- fs denotes the sampling frequency

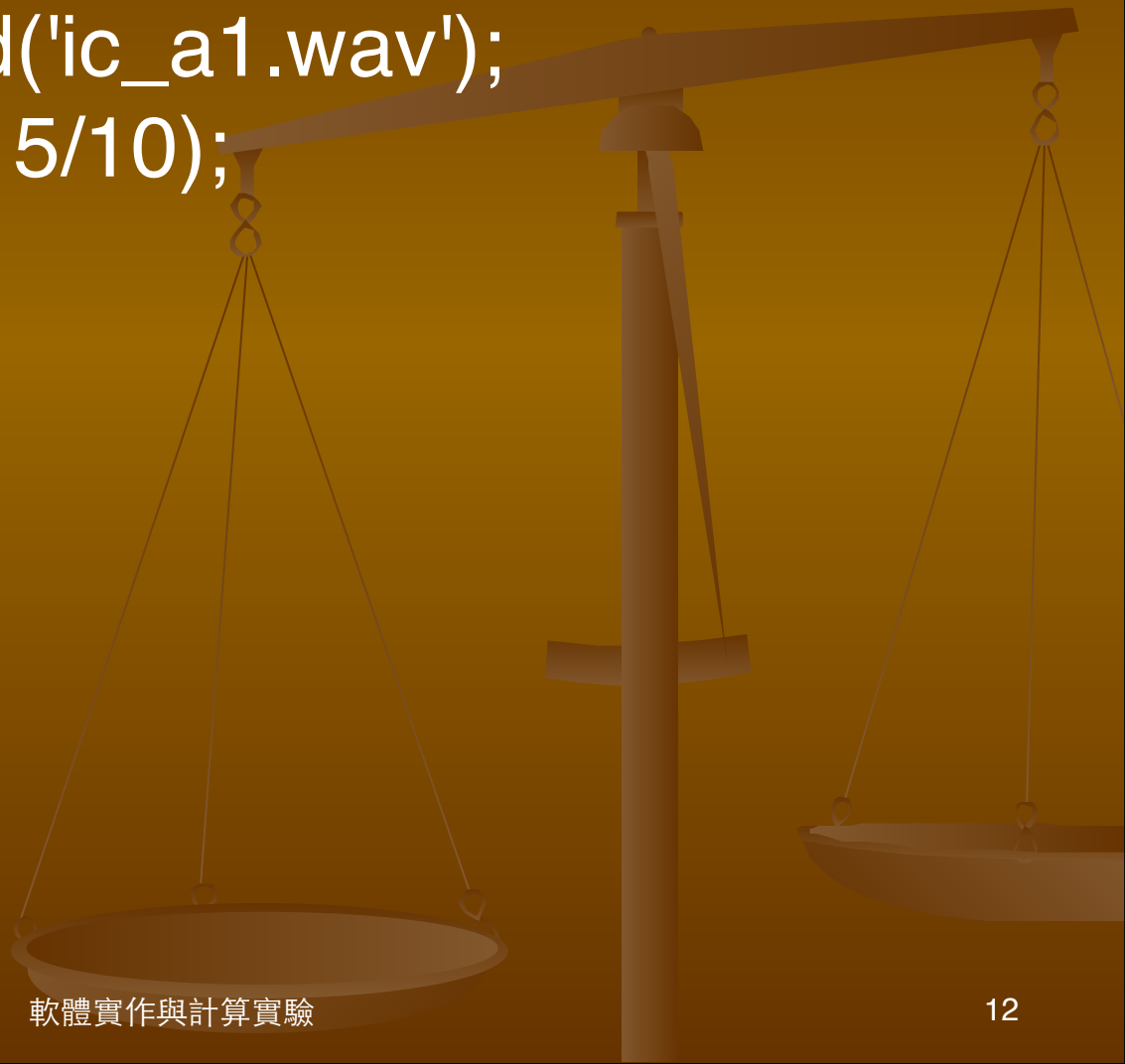
Slow down

```
[s1,fs]=wavread('ic_a1.wav');
wavplay(s1,fs*8/10);
```



Higher frequency

```
[s1,fs]=wavread('ic_a1.wav');
wavplay(s1,fs*15/10);
```



Recording

- Install microphone
- Play sound recorder
- Save as a WMA file

song.wma

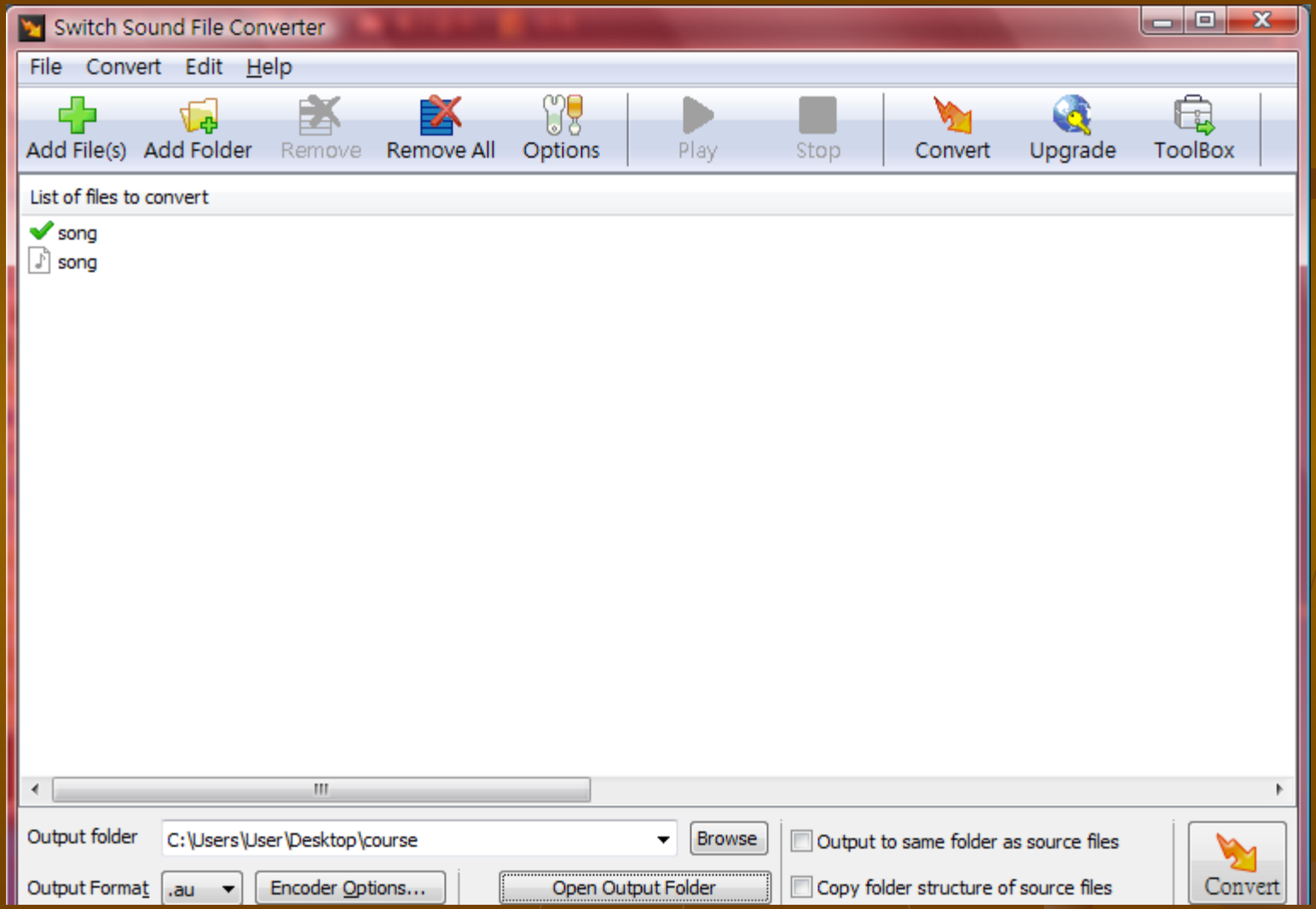
song2.wma


VAIO
solo

Audio converter software

- Switch: convert WMA file to au file

  Audio Sound File Converter Software


Switch Sound File Converter
File Convert Edit Help
Add File(s) Add Folder Remove Remove All Options Play Stop Convert Upgrade ToolBox
List of files to convert
song
song
Output folder C:\Users\User\Desktop\course Browse
Output to same folder as source files
Output Format .au Encoder Options... Open Output Folder
Copy folder structure of source files
Convert

Read and play an au file

```
>> [data,fs]=auread('song.au');
>> sound(data,fs)
```

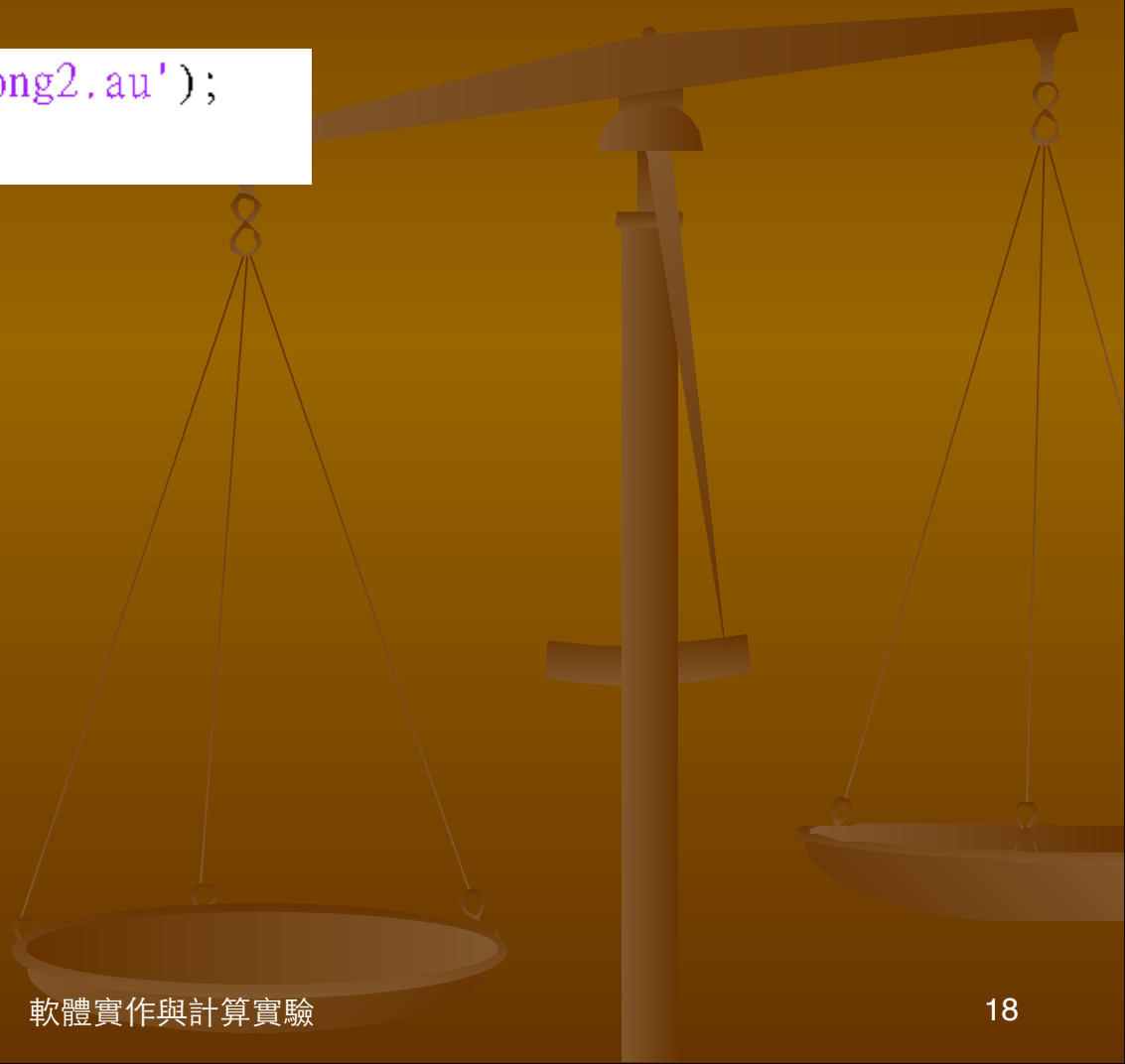
song.au

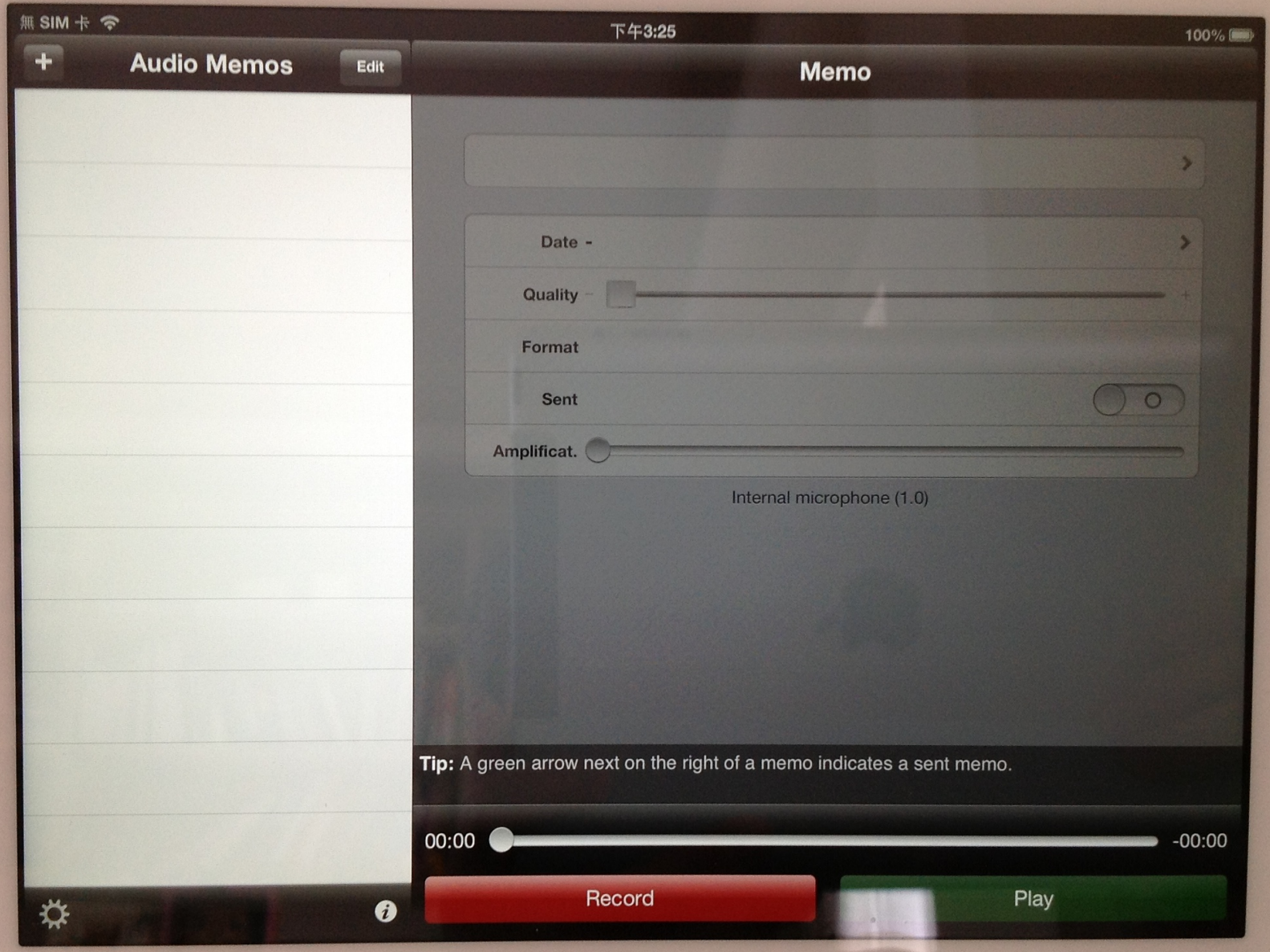
song2.au

Read and play an au file

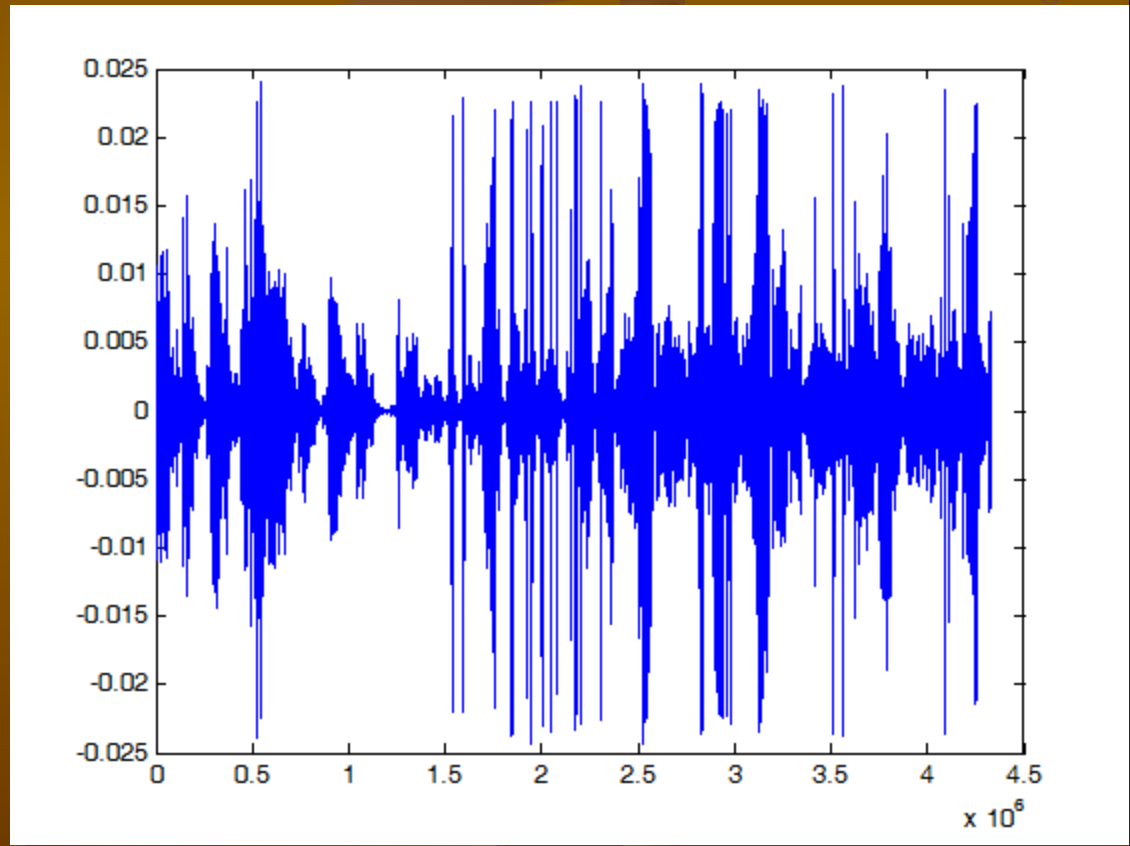
```
>> [data,fs]=auread('song2.au');
>> sound(data,fs)
```

song2.au



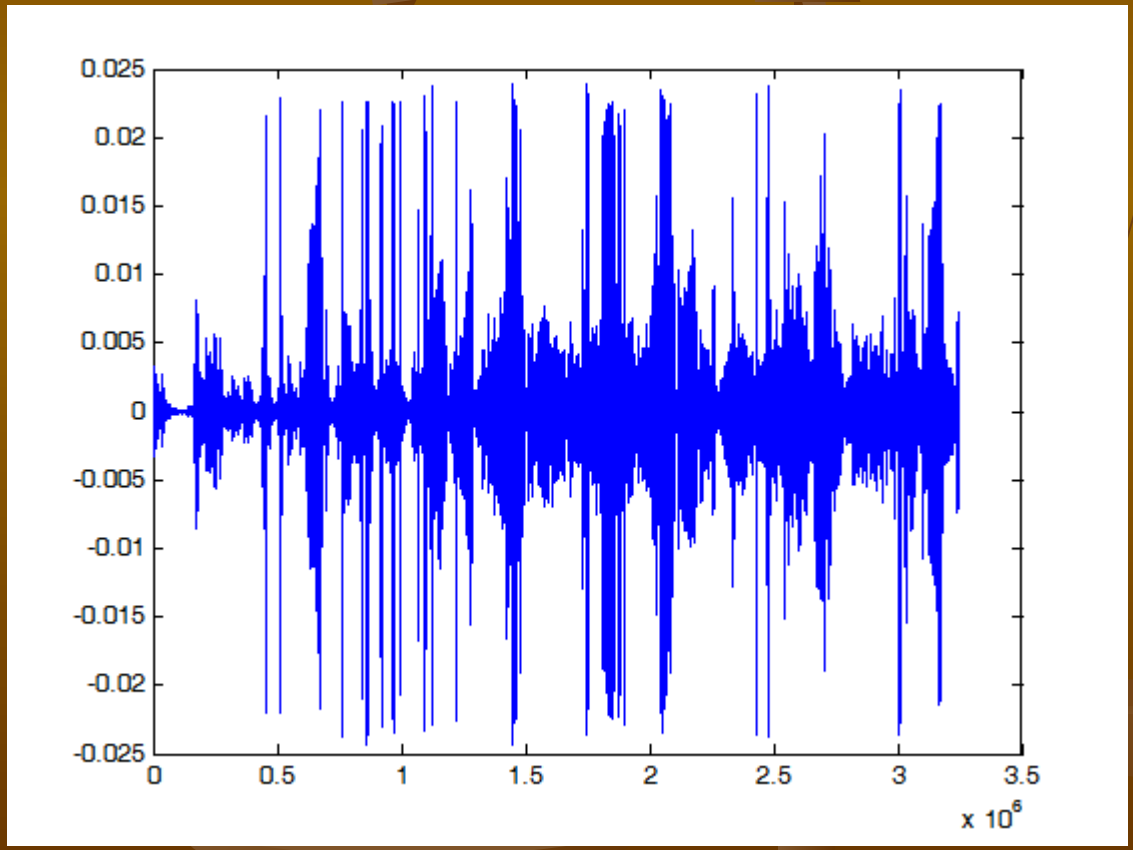

無 SIM 卡
下午3:25
100%
Audio Memos
Edit
Memo
Date -
Quality
Format
Sent
Amplificat.
Internal microphone (1.0)
Tip: A green arrow next on the right of a memo indicates a sent memo.
00:00
-00:00
Record
Play

```
>> [T,d]=size(data)

T =

      4327424


d =

      2

>> plot(1:1:T,data(:,1))
```



Segmentation

```
>> data2=data(floor(T/4):T,:);
>> sound(data2,fs)
>> plot(1:1:size(data2,1),data2(:,1))
```



Transpose of a matrix

```
A=[1 2 3;4 5 6;7 8 9];
A'
```

```
ans =

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

```
A =

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

Matrix and gray-level Image

- Represent an image by a matrix
- Each entry of a matrix represents the gray level of a pixel in an image
- Image
  - Rotation
  - Mirror
  - Transpose

Image Read

- I=imread('filename')
  - Read a color image from a file
  - Store to matrix I
- Each pixel in a color image is with three values for representing Red, Green and Blue color intensities

Color images

CMW2.jpg

```
A=imread('CMW2.jpg');
image(A);
```

取自網路

A color image

- >> size(I)

ans =

235 275 3

- There are 235 rows and 275 columns of pixels in the image
- RGB intensities at each pixel are represented by three values

Gray image

cmw1.bmp

```
I=imread('cmw1.bmp');
image(I)
```

A gray-level image

- >> size(I)

ans =

235 275

- There are 235 rows and 275 columns of pixels in the images
- One gray-level intensity at each pixel of the image

Transpose of an image

iPad 下午6:05

MATLAB Mobile

```
>> J=I(:,:,1);
>> imagesc(J')
>> colormap(gray)
```

```
J=I(:,:,1);
imagesc(J');
colormap(gray)
```

Transpose

Example

```
A=[1 2 3;4 5 6;7 8 9];
A(:,3:-1:1)
```

```
ans =

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

```
A =

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

Mirror of an image

iPad 下午6:07

MATLAB Mobile

```
>> J=I(:,:,1);
>> n=size(J,2);
>> imagesc(J(:,n:-1:1))
>> colormap(gray)
```

```
J=I(:,:,1);
n=size(J,2);
imagesc(J(:,n:-1:1))
colormap(gray)
```

Mirror

Rotation

```
A=[1 2 3;4 5 6;7 8 9];
B=A';
B(:,3:-1:1)
```

```
ans =

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

```
A =

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

Clockwise Rotation of an image

```
>> J=I(:,:,1)';
>> n=size(J,2);
>> imagesc(J(:,n:-1:1))
>> colormap(gray)
```

J=I(:,:,1)';
n=size(J,2);
imagesc(J(:,n:-1:1))
colormap(gray)

Clockwise Rotation

rgb2gray

```
A=imread('CMW2.jpg');
image(A);
IG=rgb2gray(A);
imshow(IG);
imwrite(IG,'cmw2.bmp','bmp')
```

Size of an image

```
>> size(I)

ans =

   235   275
```

Uint8 to double

I=imread('cmw1.bmp');

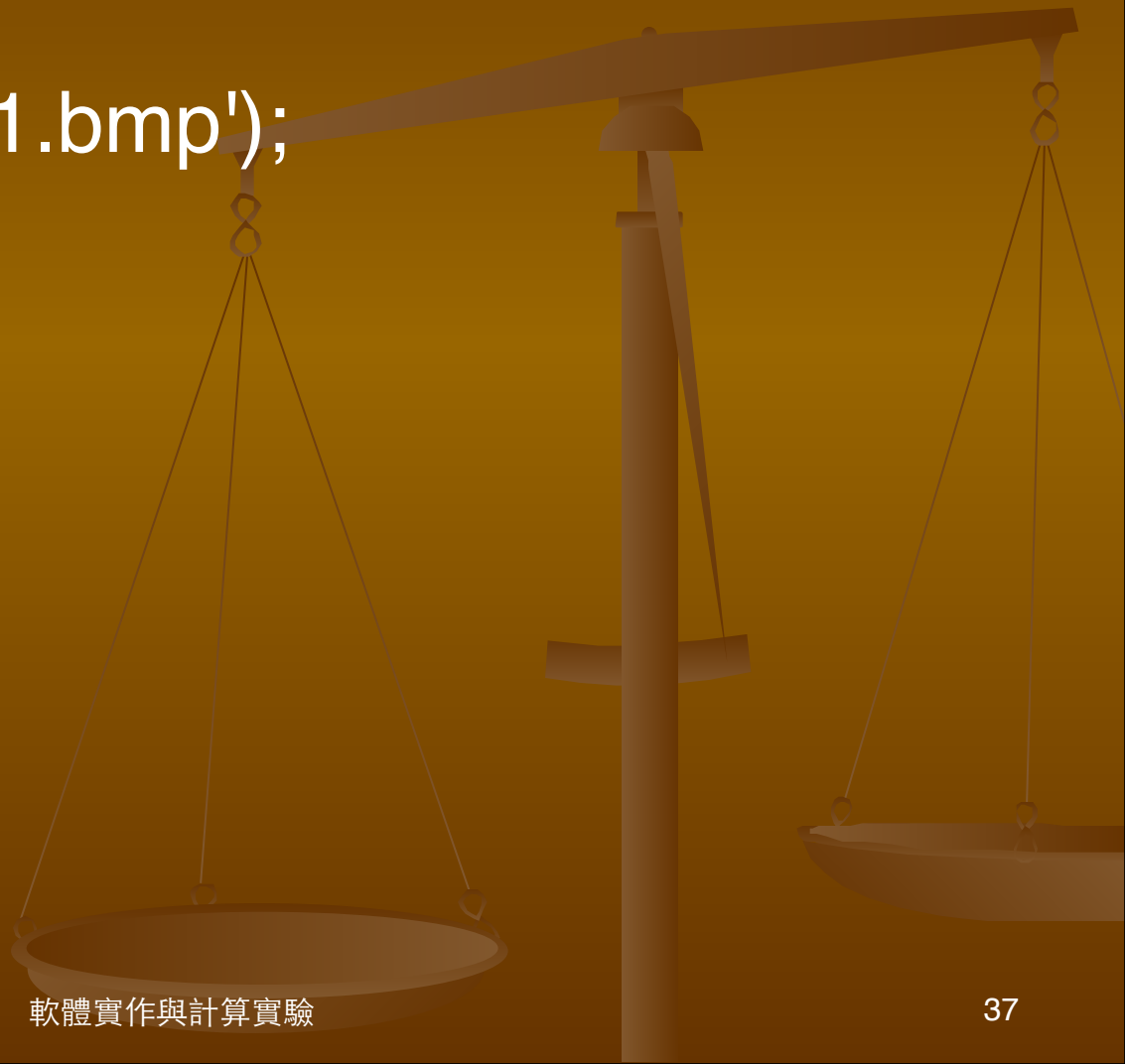
image(I)

J=double(I)

- I is a matrix with elements(uint8)
- J is a matrix with double elements belonging {0,…,255}

Double to uint8

```
I=imread('cmw1.bmp');
J=double(I)
image(uint8(J))
```



Reshape to a vector

```
I=imread('cmw1.bmp');
[m,n]=size(I)
J=double(I); Jv=reshape(I,1,m*n);
```

reshape

```
I=imread('cmw1.bmp');
[m,n]=size(I)
J=double(I); Jv=reshape(I,1,m*n);
vJ=reshape(Jv,m,n);
image(uint8(vJ));
```

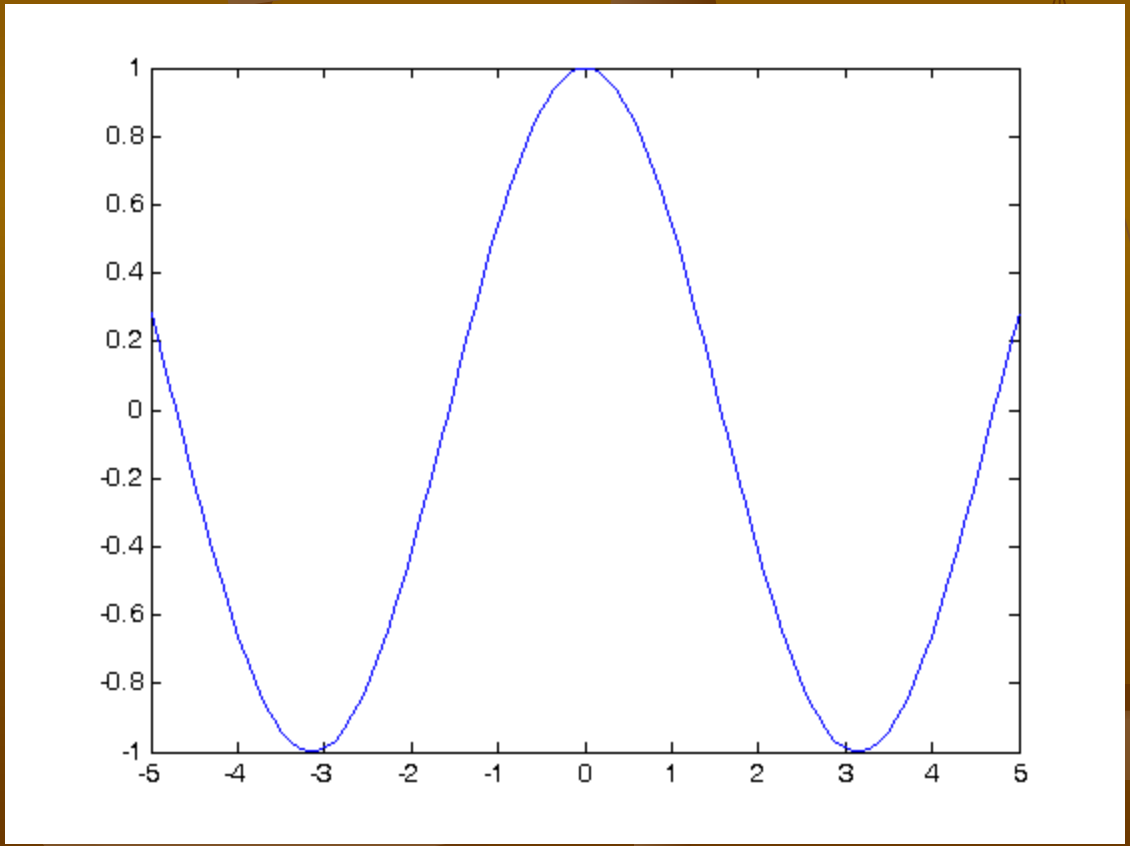
Horizontal mirror

transpose

Vertical mirror

Cosine function

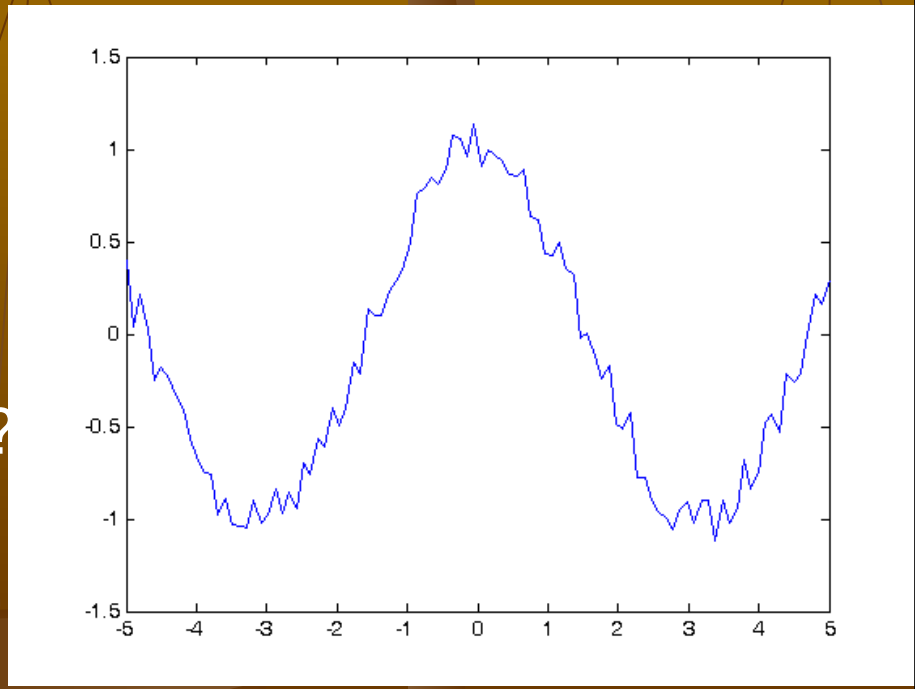
```
x=linspace(-5,5);
plot(x,cos(x));
```



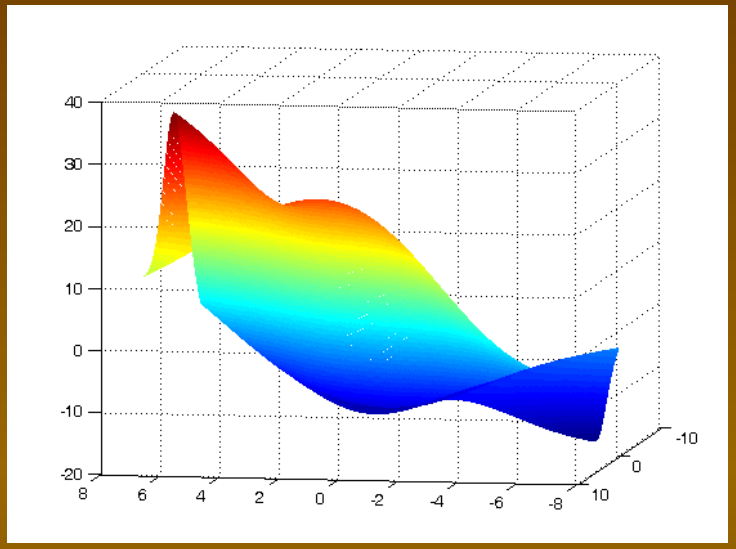
Add noise to cosine

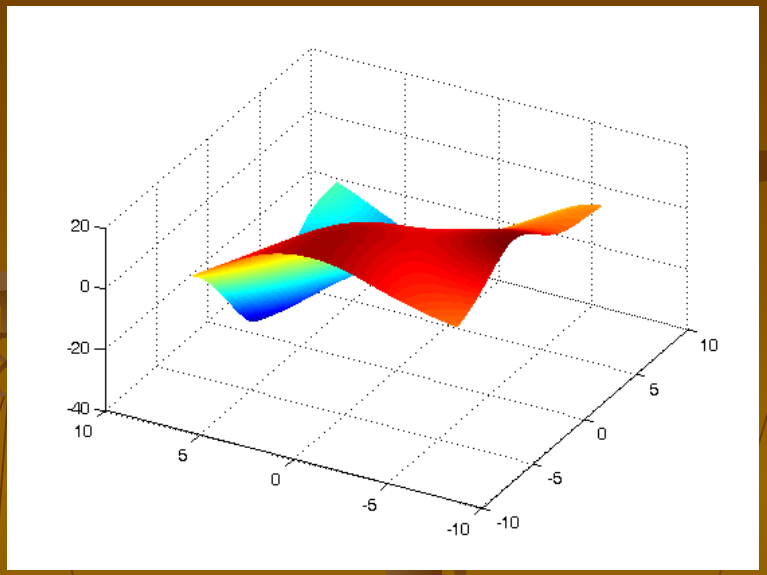
```
x=linspace(-5,5);
L=length(x);
Noise = (rand(1,L)-0.5)*0.3;
plot(x,cos(x)+Noise);
```

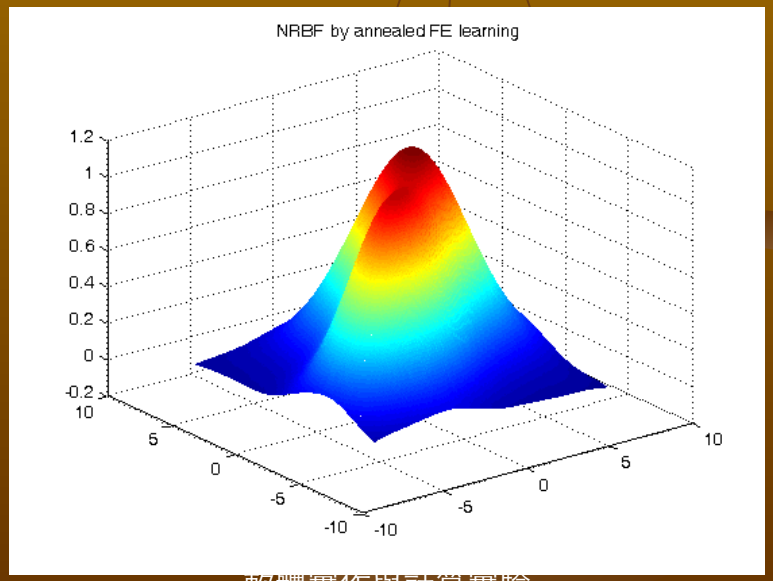
?? How to filter noises for recovering the original function??



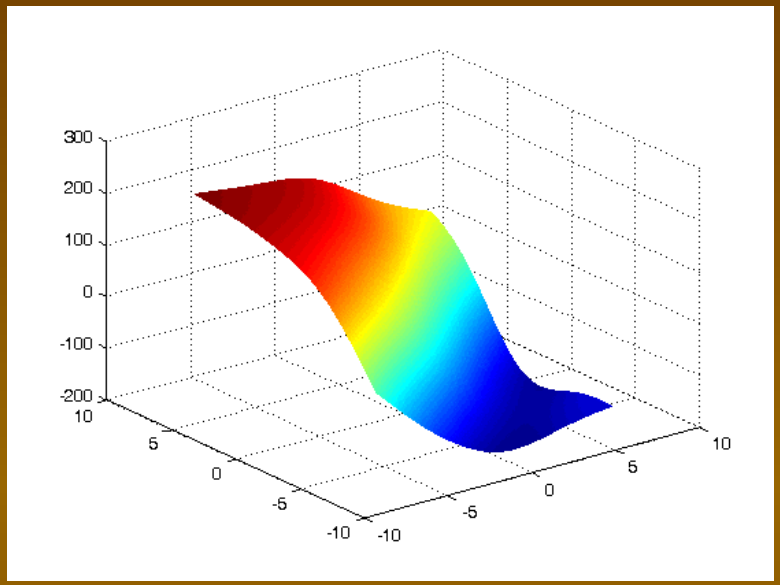
2D functions

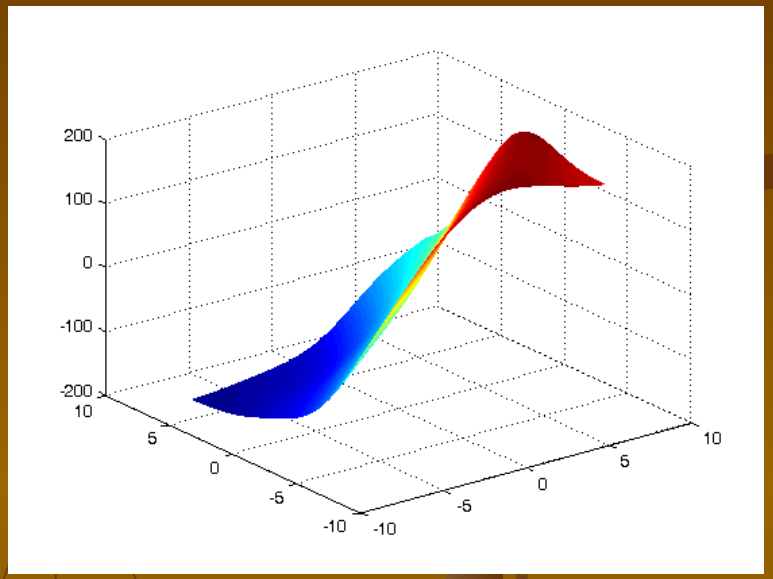


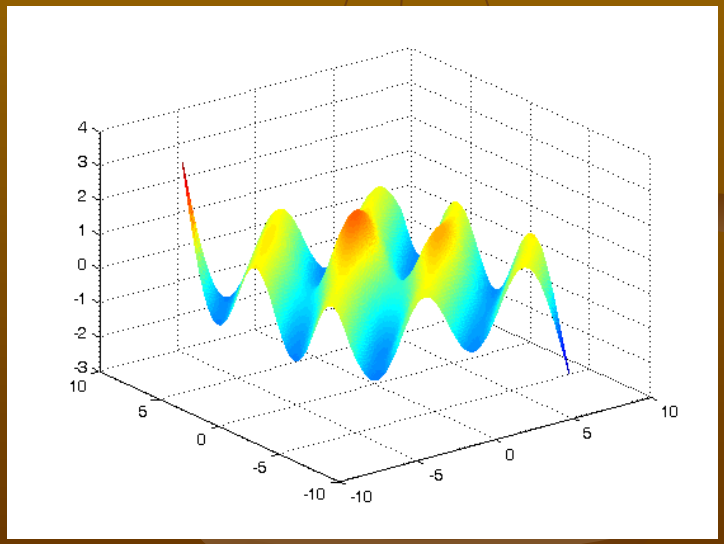




2D functions







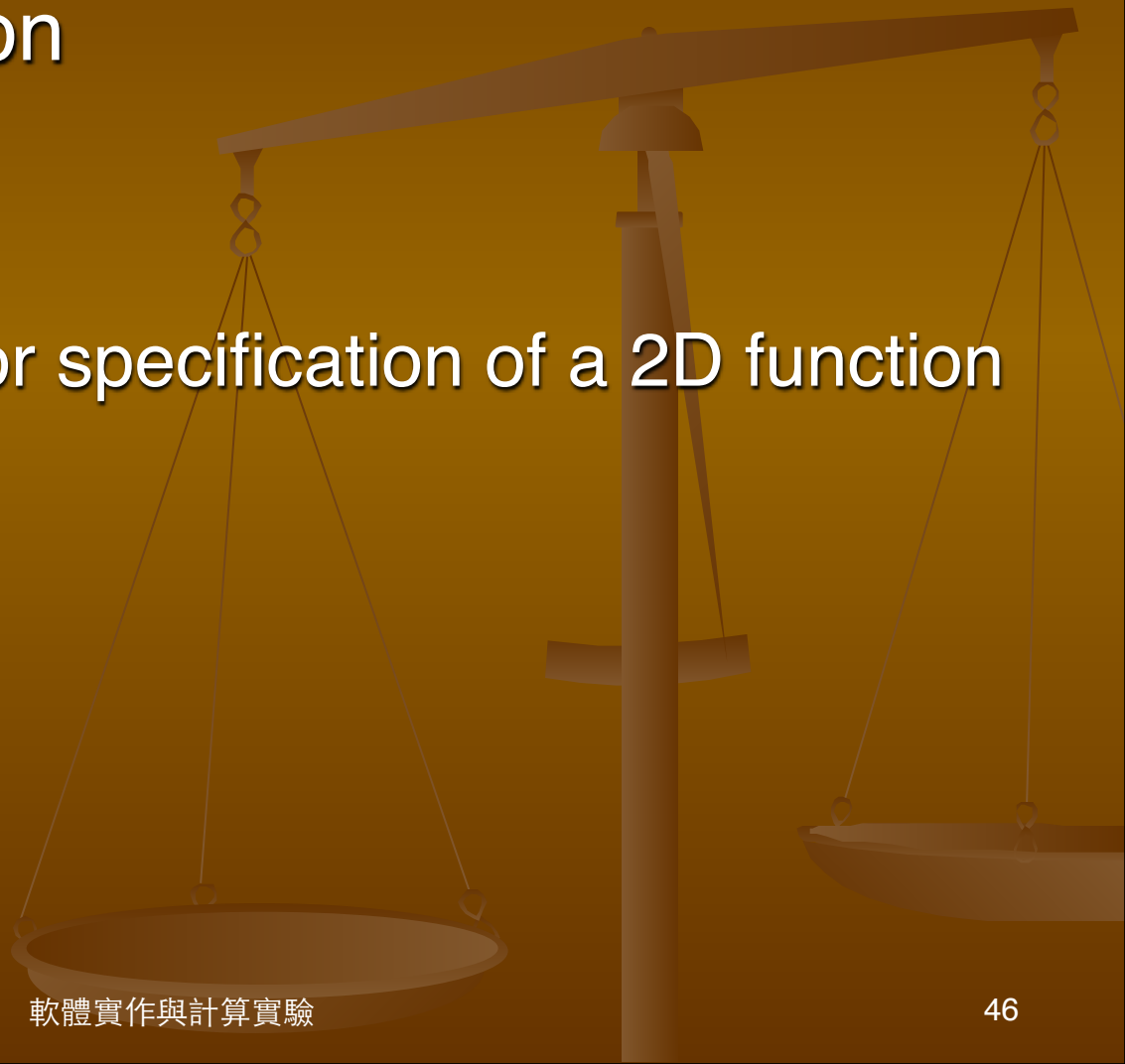
2D function

plot2d.m

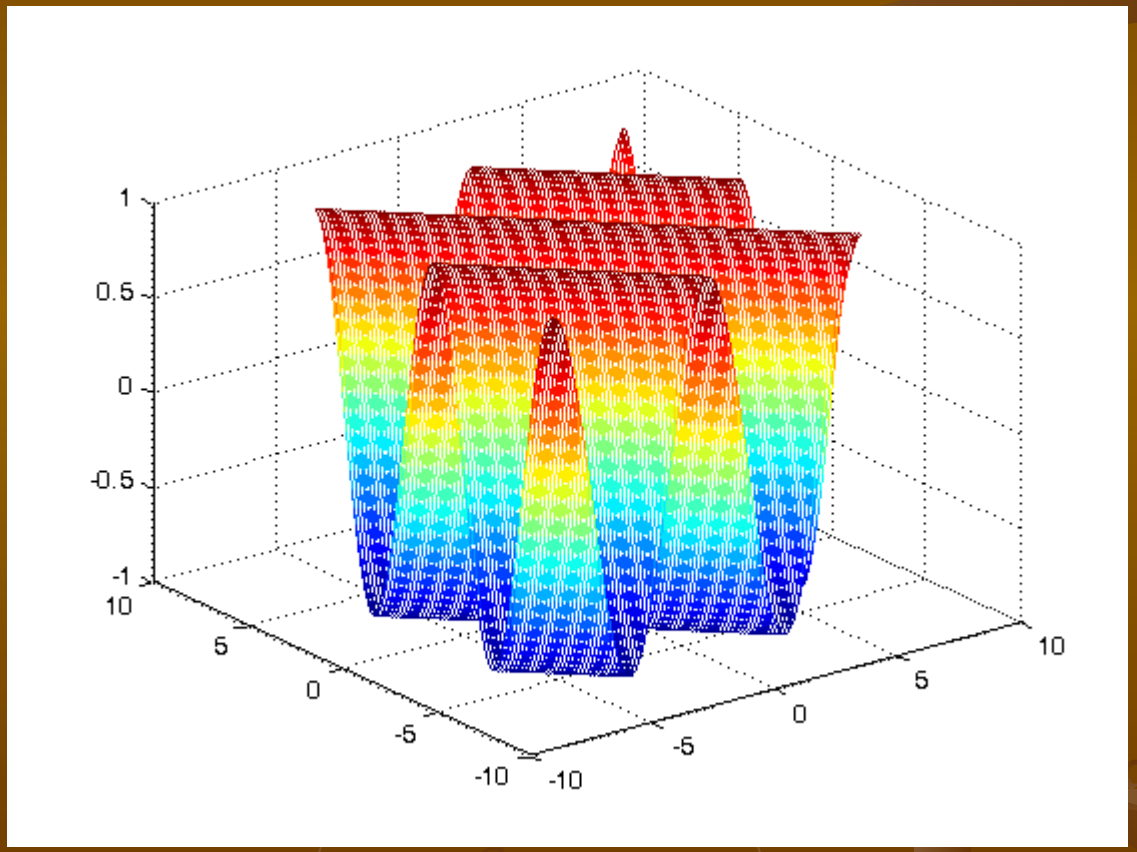
```
fstr=input('input a 2D function: x1.^2+x2.^2+cos(x1) :','s');
fx=inline(fstr);
range=2*pi;
x1=-range:0.1:range;
x2=x1;
for i=1:length(x1)
        C(i,:)=fx(x1(i),x2);
end
mesh(x1,x2,C);
```

Experiment (4)

- Plot a 2D function
  - Load plot2d.m
  - Execute plot2d
  - Keyin a string for specification of a 2D function

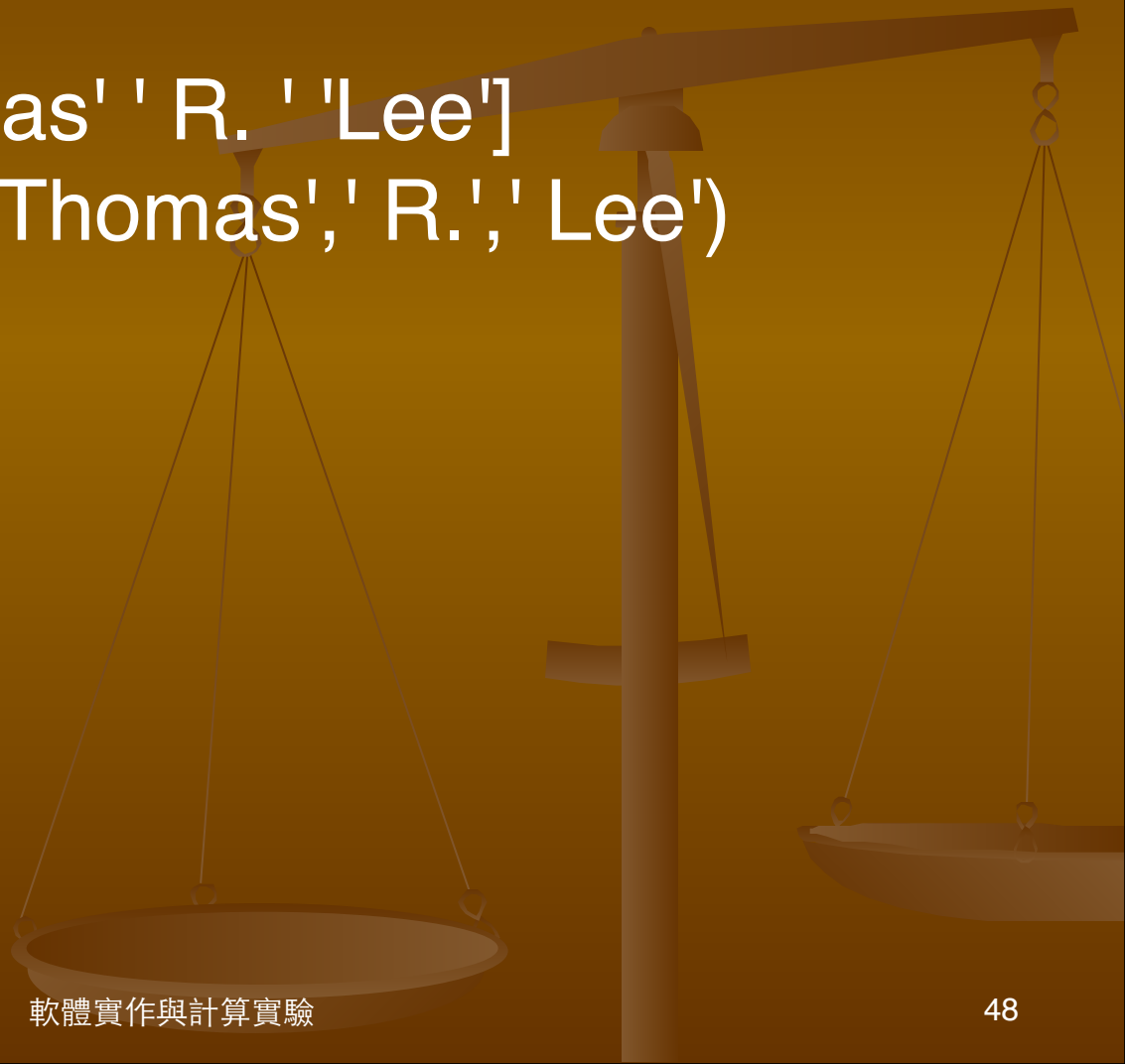


```
>> plot2d
input a 2D function: x1.^2+x2.^2+cos(x1) :cos(x1+x2)
```



Create strings

```
name = ['Thomas' ' R. ' 'Lee']
name = strcat('Thomas',' R.',' Lee')
```



A vertical array of strings

Create a vertical array of strings.
C = strvcat('Hello','Yes','No','Goodbye')

C =
Hello
Yes
No
Goodbye

- C is a matrix
- C(i,:) gets the i-th row

Cell array of strings

Create a cell array of strings.

```
S = {'Hello' 'Yes' 'No' 'Goodbye'}

S =
    'Hello'    'Yes'    'No'    'Goodbye'
```

Cell array of strings

S = {'Hello' 'Yes' 'No' 'Goodbye'}

size(S)

- Return the size of the cell array
- Each cell in S contains a string

S(1,2)

- Specify the string stored at the second column

Example

T={'Mon' 'Tue' 'Wen' 'Thu'}
S = {'Hello' 'Yes' 'No' 'Goodbye'}
ST=[S;T]