Matlab Lecture 3: Finishing with MATLAB

- Part solution to Lab 1:-

```matlab
function [t, sinewave] = sinegen(fsig, fsamp, ncycle)
% Sinewave Generation
% fsig = signal frequency
% fsamp = sampling frequency
% ncycle = number of cycles to generate
% Peter Cheung
% 15th October 1998.
% calculate angular increment per sample
delta_angle = 2*pi*fsig/fsamp;
% create angle vector for ncycle cycles
t = 0:delta_angle:ncycle*(2*pi);
% create sine wave
sinewave = sin(t);
% convert angle to time: time = angle/(2*pi*fsig)
t = t/(2*pi*fsig);
end
```

Solution to Lab 1 (con’t)

```matlab
% Model answer to Lab Session 1
% Exercise 2 - file: lab1_2.m
% define sampling frequency
fs = 44100;
% define signal frequency
f = 1000;
% create sine wave
[t,sinewave]=sinegen(f,fs,4);
% plot it
plot(t,sinewave);
grid
% label axes
xlabel('Time (in sec)');ylabel('Amplitude');
title('Sinewave at 1kHz');
```

Must use Add Path (or Set Path)

- Must use Set Path manual or addpath command to make new .m files visible!

Lab 1 (con’t) - Noisy Sinewave

```matlab
% Program 2 - file: lab1_2.m
% define signal frequency
f = 1000;
% create sine wave
[sinewave, t] = sinegen(f, fs);
% add noise
noisy_signal = sinewave + randn(size(sinewave));
% plot it
plot(t, noisy_signal);
grid
% label axes
xlabel('Time (in sec)'); ylabel('Amplitude');
title('Noisy Sinewave at 1kHz');
```
Logical Subscripting

- The logical vectors created from logical and relational operations can be used to reference subarrays.
- Suppose $X$ is an ordinary matrix and $L$ is a matrix of the same size that is the result of some logical operation. Then $X(L)$ specifies the elements of $X$ where the elements of $L$ are nonzero.
- Suppose:

  $$x = 2.1 \ 1.7 \ 1.6 \ 1.5 \ NaN \ 1.9 \ 1.8 \ 1.5 \ 5.1 \ 1.8 \ 1.4 \ 2.2 \ 1.6 \ 1.8$$

  ```
  >> x = finite(x)
  x = 2.1 \ 1.7 \ 1.6 \ 1.5 \ 1.9 \ 1.8 \ 1.5 \ 5.1 \ 1.8 \ 1.4 \ 2.2 \ 1.6 \ 1.8
  ```

Logical Subscripting in action

- Now there is one observation, 5.1, which seems to be very different from the others. It is an outlier. The following statement removes outliers, in this case those elements more than three standard deviations from the mean.

  ```
  x = x(abs(x-mean(x)) <= 3*std(x))
  x = 2.1 \ 1.7 \ 1.6 \ 1.5 \ 1.9 \ 1.8 \ 1.5 \ 1.8 \ 1.4 \ 2.2 \ 1.6 \ 1.8
  ```

Structures in MATLAB

- Structures are multidimensional MATLAB arrays with elements accessed by textual field designators. For example,

  ```
  S.name = 'Ed Plum';
  S.score = 83;
  S.grade = 'B+'
  ```

  creates a scalar structure with three fields.

  ```
  S =
  name: 'Ed Plum'
  score: 83
  grade: 'B+'
  ```

- an entire element can be added with a single statement.

  ```
  S(3) = struct('name','Jerry Garcia',
  'score',70,'grade','C')
  ```

Assignment: Image Warping

- Four Tasks:
  - Image rotation
  - Image shearing
  - Edge detection
  - Image blurring
- Deadline
  - See Assignment sheet - submit to Level 6 Teaching Office
- Deliverables:-
  - Well commented listing of your MATLAB files
  - Evidence that it works (i.e. hardcopy for each of the special effects)
  - Floppy disk containing a ready-to-try copy of your programmes
**Problem 1: Rotation (1)**

```
Show(clown)
Show(rotate(clown, pi/3))
```

**Problem 1: Rotation (2)**

Forward Mapping:

\[
\begin{align*}
(x_{destination}, y_{destination}) &= (x_{source}, y_{source}) + (X_{centre}, Y_{centre}) \\
&= \cos(\theta) x_{source} - \sin(\theta) y_{source} + X_{centre} \\
&\quad + \sin(\theta) x_{source} + \cos(\theta) y_{source} + Y_{centre}
\end{align*}
\]

**Problem 1: Rotation (3)**

**For each pixel in the source image:**

1. Work out the destination pixel location using the forward mapping equation.
2. Paint that destination pixel with the source image value.

**Pixel Number**

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
</tr>
<tr>
<td>9</td>
<td>10</td>
<td>11</td>
<td>12</td>
</tr>
<tr>
<td>13</td>
<td>14</td>
<td>15</td>
<td>16</td>
</tr>
</tbody>
</table>

**Forward Mapping**

<table>
<thead>
<tr>
<th>1,2</th>
<th>3,4</th>
</tr>
</thead>
<tbody>
<tr>
<td>5,9</td>
<td>6,7</td>
</tr>
<tr>
<td>10</td>
<td>11</td>
</tr>
<tr>
<td>8,12</td>
<td>14,15</td>
</tr>
</tbody>
</table>

**Reverse Mapping**

<table>
<thead>
<tr>
<th>Pixel Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 2 3 4</td>
</tr>
<tr>
<td>5 6 7 8</td>
</tr>
<tr>
<td>9 10 11 12</td>
</tr>
<tr>
<td>13 14 15 16</td>
</tr>
</tbody>
</table>

```
Source Image

For each pixel in the destination image:

1. Work out where the pixel maps to in the source image, using the reverse mapping equation.
2. Paint the destination pixel with that source pixel value.

```
Destination Image

```
Problem 2 & 3: Shearing & Edge Detection