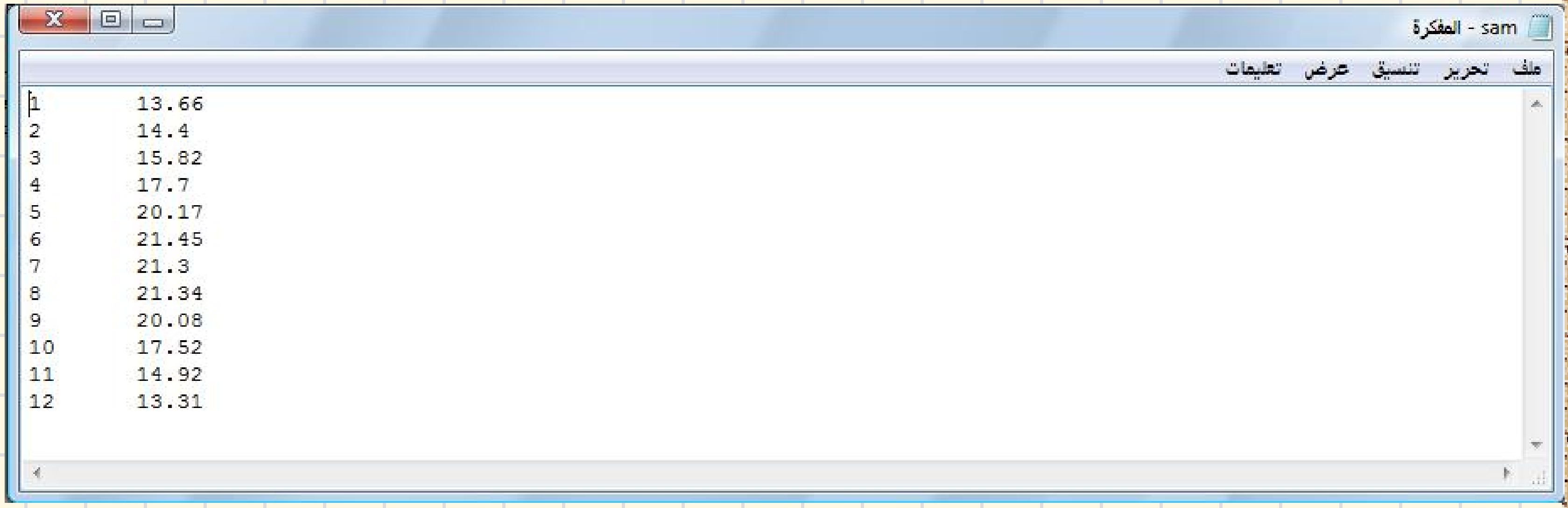
MATLAB

Lecture 5

Loading Data into MATLAB for Plotting

This example show you how to load a simple data set and plot it .

The sam.dat file contains two columns of numbers. The first is the number of the month, and the second is the mean Temperatures recorded at the King Khalid International Airport between 1961 and 1990.



```
1        13.66
2        14.4
3        15.82
4        17.7
5        20.17
6        21.45
7        21.3
8        21.34
9        20.08
10       17.52
11       14.92
12       13.31
```

Here are the MATLAB commands to create a symbol plot with the data from sam.dat .

```
>> load sam.dat;                    %  read data into sam matrix
>> month = sam(:,1);                %  copy first column of sam into month
>> Temp  = sam(:,2);                %  and second column into Temp.
>> plot(month,Temp,'ro');           %  plot Temp vs. month with circles
>> xlabel('month of the year');% add axis labels and plot title
>> ylabel('mean Temperatures (%)');
>> title('Mean monthly Temperatures at King Khalid International
              Airport');
```

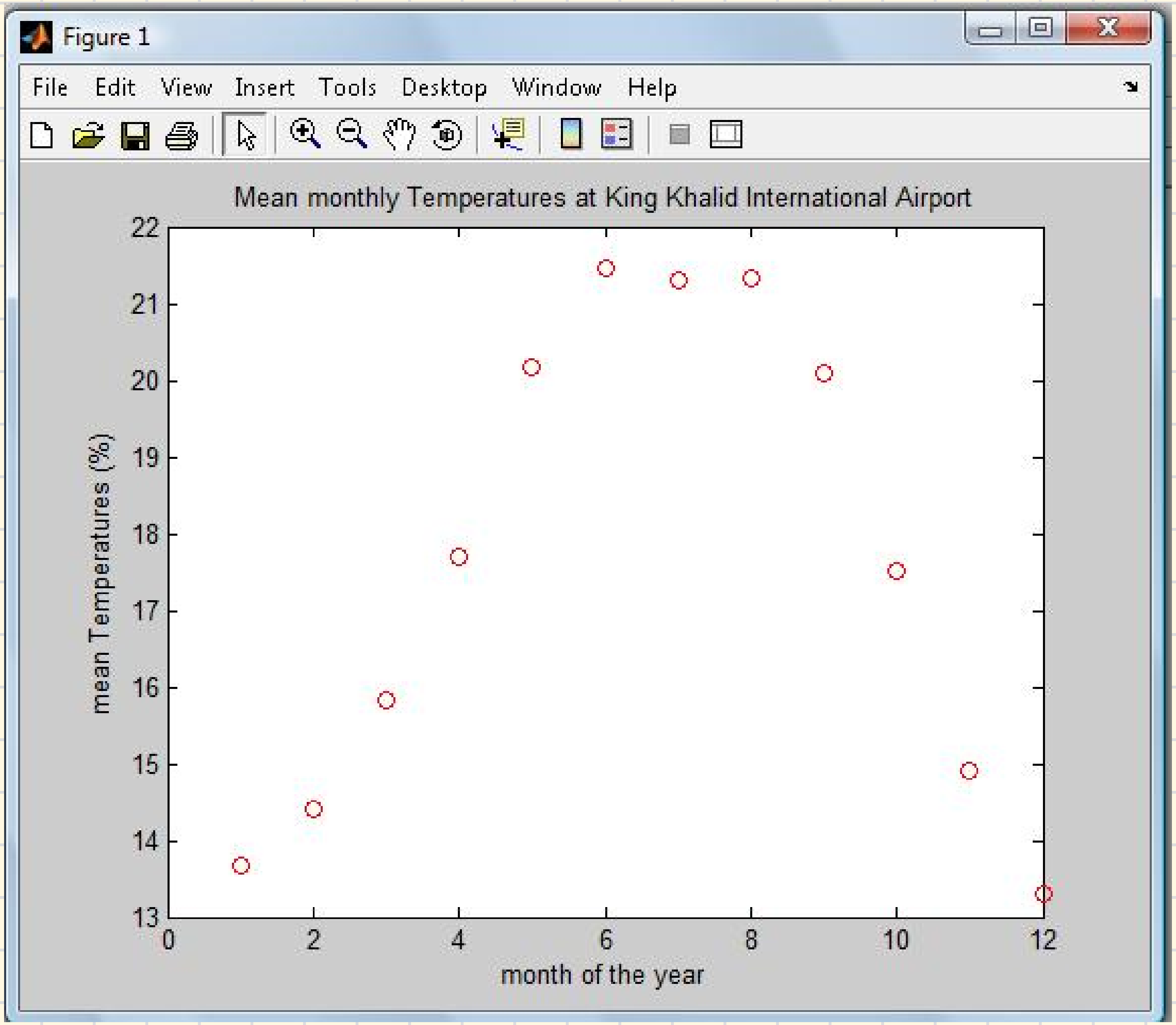


Figure 1
File Edit View Insert Tools Desktop Window Help
Mean monthly Temperatures at King Khalid International Airport
mean Temperatures (%)
month of the year
22
21
20
19
18
17
16
15
14
13
0
2
4
6
8
10
12

Using vector operations instead of loops:

Consider the following loop:

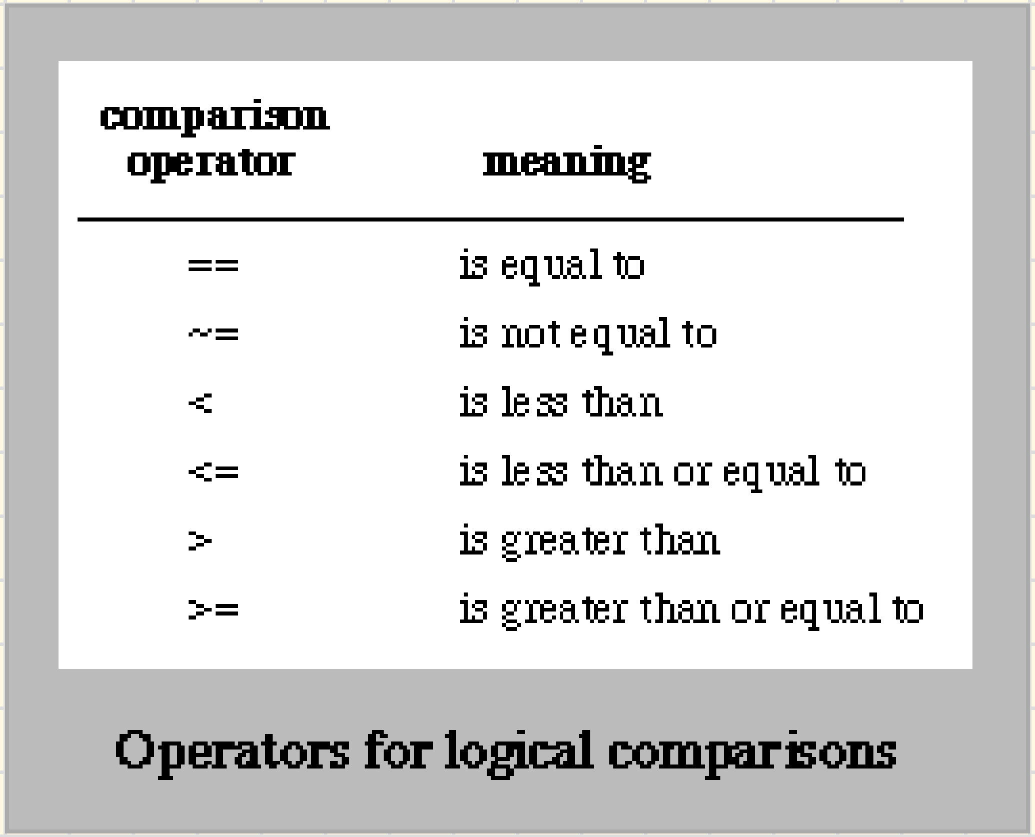
```
dx = pi/30;
nx = 1 + 2*pi/dx;
for i = 1:nx
    x(i) = (i-1)*dx;
    y(i) = sin(3*x(i));
end
```

Pre-allocating memory for vectors and matrices:

Though MATLAB will automatically adjust the size of a matrix (or vector) it is usually a good idea to pre-allocate the matrix. Pre-allocation incurs the cost of memory allocation just once, and it guarantees that matrix elements will be stored in contiguous locations in RAM (by columns).

```
dx = pi/30;
nx = 1 + 2*pi/dx;
nx2 = nx/2;

for i = 1:nx2
   x(i) = (i-1)*dx;
   y(i) = sin(3*x(i));
end

for i = nx2+1:nx
   x(i) = (i-1)*dx;
   y(i) = sin(5*x(i));
end
```

```
if <condition>, <program> end

if <condition1>, <program1> else <program2> end

if <condition1>, <program1>
elseif <condition2>, <program2>
end
```



comparison operator	meaning
==	is equal to
~=	is not equal to
<	is less than
<=	is less than or equal to
>	is greater than
>=	is greater than or equal to

Operators for logical comparisons

Example 1:

```
x=2
If x==0,
   y=x+1;
   else y=x/2;
end
```

Example 2:

A simple warning

```
>> d = b^2 - 4*a*c;
>> if d<0
>>   disp('warning: discriminant is negative, roots are
           imaginary');
>> end
```

Example 3:

Or a warning plus extra notification

```
>> d = b^2 - 4*a*c;
>> if d<0
>>   disp('warning: discriminant is negative, roots are
           imaginary');
>> else
>>   disp('OK: roots are real, but may be repeated');
>> end
```

Example 3:

Or, no secrets whatsoever

```
>> d = b^2 - 4*a*c;
>> if d<0
>>   disp('warning: discriminant is negative, roots are
           imaginary');
>> elseif d==0
>>   disp('discriminant is zero, roots are repeated');
>> else
>>   disp('OK: roots are real and distinct');
>> end
```

Sometimes you will have spent much time creating matrices in the course of your MATLAB session and you would like to use these same matrices in your next session. You can save these values in a file by typing

save filename
This creates a file
filename.mat

- save filename x y z
- which will save the variables x,y,z in the file filename.mat. The variables can be reloaded in a future session by typing

- load filename
- When you are ready to print out the results of a session, you can store the results in a file and print the file from the operating system using the "print" command appropriate for your operating system. The file is created using the command