EEE 208 – Programming for EEE

Assist. Prof. Dr. Engin Mendi

How to display numbers in a string?

- Use the `num2str` function! It converts numbers to a string.
- `T = num2str(X)` converts the matrix X into a string representation T with about 4 digits and an exponent if required. This is useful for labeling plots with the TITLE, XLABEL, YLABEL, and TEXT commands.

- `T = num2str(X,N)` converts the matrix X into a string representation with a maximum N digits of precision. The default number of digits is based on the magnitude of the elements of X.

- `x = 1+2i`
- `disp(['The real part of x is ' num2str(real(x)) ' and the imaginary part is ' num2str(imag(x))])`

Conversion Functions – Check the output's type

- Always check the help file to make sure you are using the right function.

- **datestr**
- **num2str**
- **int2str**: rounds the elements of the matrix to integers and converts the result into a string matrix.

```
>> A = int2str(3+4.6)
>> B = int2str(pi)
```

- **mat2str**: converts a matrix to a string

```
>> C = mat2str(magic(4)^2)
```

- **str2num**: converts a string matrix to numeric array

```
>> str2num(C)
```

Two Math Functions: Rem and Mod

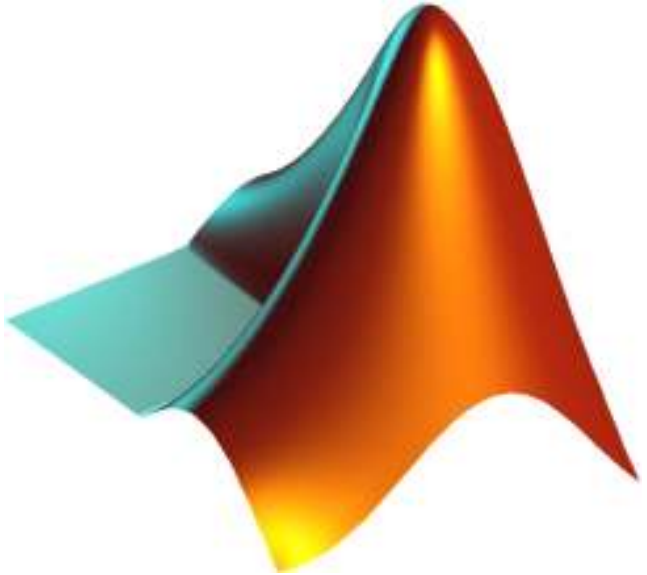
- `M=mod(x,y)` modulus after dividing x by y.
  - `mod(x,y)` is x - n.*y where n = floor(x./y) if y ~= 0
  - `>> mod(3,2)` is 1, and `mod(-3,2)` is 1
  - `>> mod(3,-2)` is -1, and `mod(-3,-2)` is -1
  - `>> mod(x,x) is 0`
  - `>> mod(x,0) is x`
- `R=rem(x,y)` reminder after dividing x by y.
  - `rem(x,y)` is x - n.*y where n = fix(x./y) if y ~= 0
  - `>> rem(3,2)` is 1, and `rem(-3,2)` is -1
  - `>> rem(3,-2)` is 1, and `rem(-3,-2)` is -1
  - `>> rem(x,0)` is always a NaN

Don't memorize! Just learn how to use them.

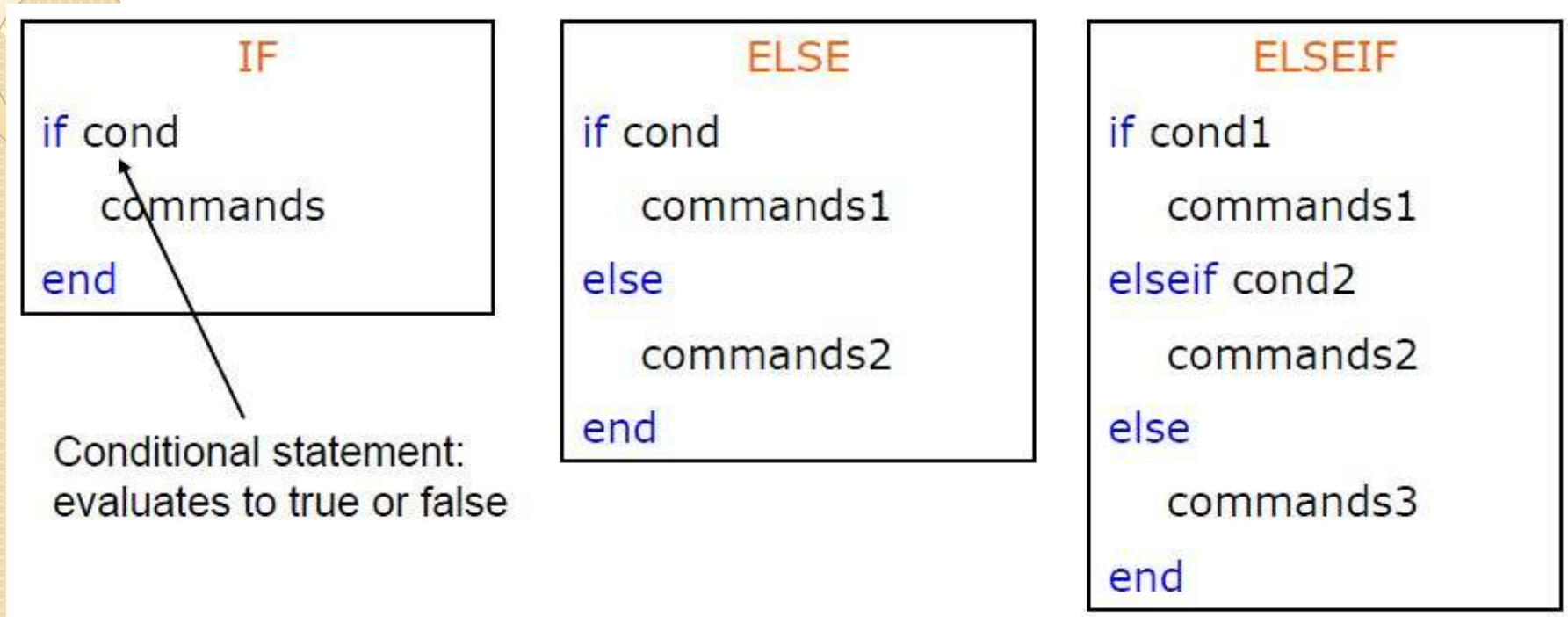
Application? Finding if a number is odd or even, or divisible by a specific integer

Statements for the Flow Control

- Conditional Control
  - `if/elseif/else`
  - `switch/case/otherwise`
- Loop Control
  - `while, for, break`
- Error Control
  - `try, catch`



Flow Control: IF

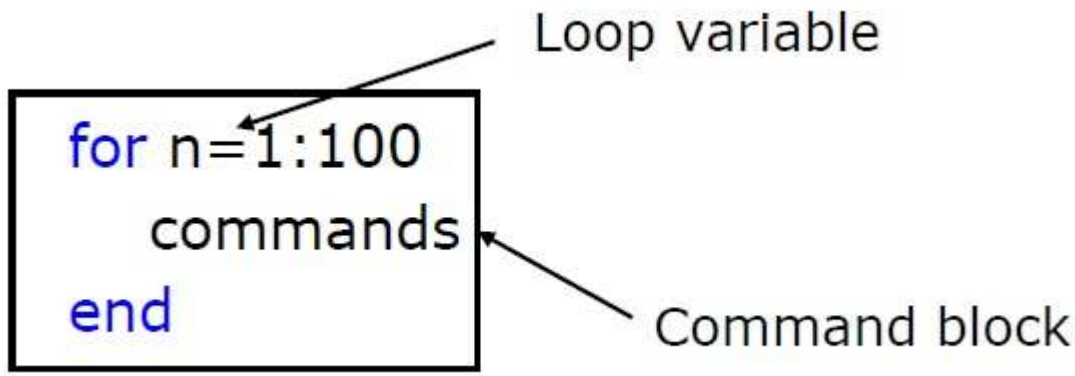


- Use relational operators: ==, ~=, >, <, >=, <=, & and | (for elements), || and && (scalars) , ~, xor, all, any

- No need to use parentheses in the conditions

Flow Control: For Loop

- It's used for a known number of iterations (repeating)
- The loop variable (n for example)
  - Is defined as a vector
  - But it's used like a scalar inside the command block
  - Does not have to have consecutive values (but it's usually nicer if they're consecutive



Flow Control: While loop

- The **while** loop is like a more general **for** loop, because it does not need the number of iterations
- The commands will run as long as the command is True (or 1)
  - `while i< length(x)+1` ... `while x>-1 | x<5`
-
- If the condition is always true, we have an **infinite loop**. To stop it, you have to press Ctrl + C

Example: Conditional Control Statements

- if, elseif and else

```
if n < 0                  % If n negative, display error message.
    disp('Input must be positive');
elseif rem(n,2) == 0 % If n positive and even, divide by 2.
    A = n/2;
else
    A = (n+1)/2;      % If n positive and odd, increment and divide.
end
```

- switch, case and otherwise

```
switch input_num
    case -1
        disp('negative one');
    case 0
        disp('zero');
    case 1
        disp('positive one');
    otherwise
         disp('other value');
end
```

Example: Loop Control Statements

- `while` (conditional loop)

```
n = 1;
while prod(1:n) < 1e100
    n = n + 1;
end
```

- `for` (iterative loop), also nested loop

```
for index = start:increment:end
    statements
end
```

```
for m = 1:5
    for n = 1:100
        A(m, n) = 1/(m + n - 1);
    end
end
```

- `continue, break`

Input-Output Commands

COMMAND	DESCRIPTION
break	exits while or for loops
disp	displays text or matrix
echo	displays m-files during execution
error	displays error messages
format	switches output display to a particular format
fprintf	displays text and matrices and specifies format for printing values
input	allows user input
keyboard	invokes the keyboard as an m-file
pause	causes an m-file to stop executing. Pressing any key cause resumption of program execution.

Class Exercise 1: Loops

Problem: Ask the user to enter a positive integer, and save it as mynumber. Then write a code with a loop for all the values n from 1 to mynumber. For each number, the code should calculate and display 'n is divisible only by 2', 'n is divisible only by 3', 'n is divisible by 2 AND 3' or 'n is NOT divisible by 2 or 3'.

Instructions: Use a for loop, the function mod or rem to figure out if a number is divisible by 2 or 3, and num2str to convert each number to a string for displaying. You can use any combination of if, else, and elseif.

Plots: Advanced Topics 1

- We can set the edge color, face color and size of markers.
  - `>> x = -pi:pi/10:pi;`
  - `>> y = tan(sin(x)) - sin(tan(x));`
  - `>> plot(x,y,'--rs','LineWidth',2, 'MarkerEdgeColor','k', 'MarkerFaceColor','g', 'MarkerSize',10)`
- We can set the font size and color of the text in xlabel, ylabel and title, or use variables in them with num2str, int2str, etc.
- Read `help title` and `help plot` for more examples.

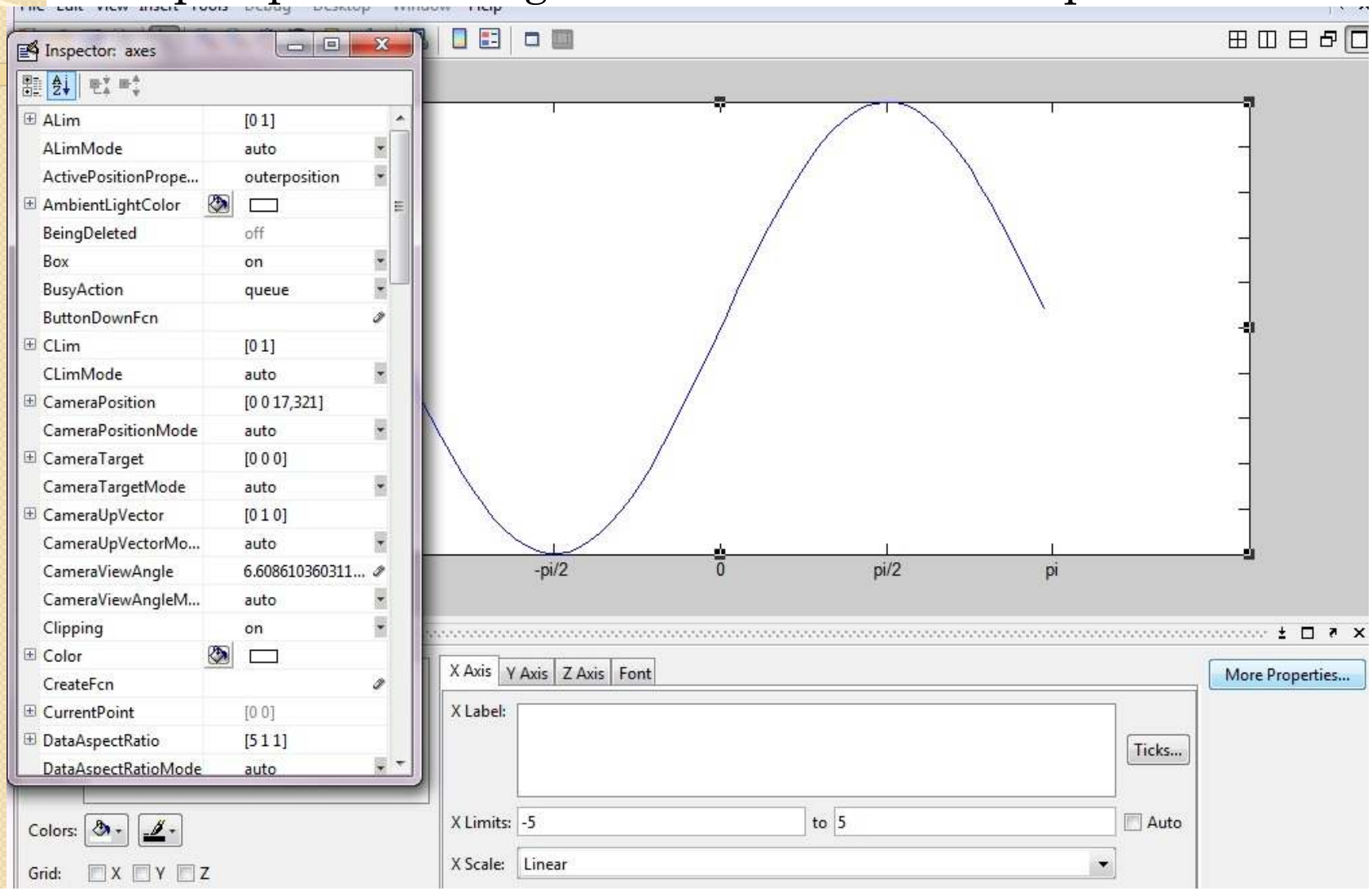
Plots: Advanced Topics 2

- We can set limits for x and y axis, and set the thickness of tick marks.
  - >> example_plot_tick.m

```
clear all;
x = -pi:.1:pi;
y = sin(x);
plot(x,y)
set(gca,'XTick',-pi:pi/2:pi) % No other value in between
set(gca,'XTickLabel',{'-pi','-pi/2', '0', 'pi/2', 'pi'}) % gca means get current axes handle
ax=axis % Reads the current axes values
axis([-5, 5, ax(3)
```

Plot Properties

- What about the other properties? Click on the arrow on top of plot > the figure borders > More Properties…



Example: Adding a text to the Plot

- We can also write a text and set its location on the graph using the **`text`** function. Open and read `example_disp_peak.m` and explain what it does.

Functions for Plotting 2D Graphs

FUNCTION	DESRIPTION
axis	freezes the axis limits
bar	plots bar chart
contour	performs contour plots
ginput	puts cross-hair input from mouse
grid	adds grid to a plot
gtext	does mouse positioned text
histogram	gives histogram bar graph
hold	holds plot (for overlaying other plots)
loglog	does log versus log plot
mesh	performs 3-D mesh plot
meshdom	domain for 3-D mesh plot
pause	wait between plots
plot	performs linear x-y plot
polar	performs polar plot
semilogx	does semilog x-y plot (x-axis logarithmic)
semilogy	does semilog x-y plot (y-axis logarithmic)
shg	shows graph screen
stairs	performs stair-step graph
text	positions text at a specified location on graph
title	used to put title on graph
xlabel	labels x-axis
ylabel	labels y-axis

Plotting Functions: polar(theta,rho)

```
>> theta=10*pi/180;
>> r=1.2;
>> angle=0:theta:2*pi;
>> mag=(r^2) .* (cos(2*angle));
>> polar(angle,mag); grid
```

Plotting Functions: semilogx, semilogy

- `semilogx(t)` creates a plot using a base 10 logarithmic scale for the x-axis and a linear scale for the y-axis. It plots the columns of t versus their index if t contains real numbers.
- `semilogx(t)` is equivalent to `semilogx(real(Y),imag(Y))` if t contains complex numbers.
- `semilogy(t)` creates a plot using a base 10 logarithmic scale for the y-axis and a linear scale for the x-axis.

Plotting Functions: loglog

- `loglog(t)` creates a plot using a base 10 logarithmic scale for the x-axis and a base 10 logarithmic scale for the y-axis. It `plots the columns of t versus their index if t contains real numbers. If t contains complex numbers, loglog(t) and loglog(real(t),imag(t)) are equivalent. loglog ignores the imaginary component in`