Plotting in MATLAB

- `linspace(a, b, n)`
  - Generates linearly spaced vectors between “a” and “b” with “n” number
- `x = linspace(0, 2*pi, 100)`
- `y = sin(x)`
- `plot(x, y)`
  - Plots functions with x and y
plot(x,y) result

Plotting in MATLAB

- $z = \cos(x)$
- `plot(x,y,x,z)`
Title and Labels of Graph

- `title('Trigonometric functions');`
- `xlabel('x-axis');`
- `ylabel('y-axis')`

help plot command

```
>> help plot
plot linear plot.
plot(x,y) plots vector Y versus vector X. If X or Y is a matrix,
   then the vector is plotted versus the rows or columns of the matrix.
   whichever line up. If X is a scalar and Y is a vector, disconnected
   line objects are created and plotted as discrete points vertically at
   X.

plot(Y) plots the columns of Y versus their index.
If Y is complex, plot(Y) is equivalent to plot(real(Y),imag(Y)).
In all other uses of plot, the imaginary part is ignored.

Various line types, plot symbols and colors may be obtained with
plot(X,Y,S) where S is a character string made from one element
from any or all the following 3 columns:
```

<table>
<thead>
<tr>
<th>Color</th>
<th>Point style</th>
<th>Line style</th>
</tr>
</thead>
<tbody>
<tr>
<td>b blue</td>
<td>. point</td>
<td>- solid</td>
</tr>
<tr>
<td>g green</td>
<td>o circle</td>
<td>: dotted</td>
</tr>
<tr>
<td>r red</td>
<td>x x-mark</td>
<td>-- dashed</td>
</tr>
<tr>
<td>cyan</td>
<td>+ plus</td>
<td>-- dashdot</td>
</tr>
<tr>
<td>m magenta</td>
<td>* star</td>
<td>none (no line)</td>
</tr>
<tr>
<td>y yellow</td>
<td>s square</td>
<td></td>
</tr>
<tr>
<td>k black</td>
<td>v triangle (down)</td>
<td></td>
</tr>
<tr>
<td>y white</td>
<td>&gt; triangle (right)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&lt; triangle (left)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>p pentagram</td>
<td></td>
</tr>
<tr>
<td></td>
<td>h histogram</td>
<td></td>
</tr>
</tbody>
</table>

Özgür ZEYDAN 5
Ploting with Styles

- `plot(x,y,'r:o',x,z,'c:x')`

- Or

- `plot(x,y,'r:o')`  
- `hold`  
- `plot(x,z,'c:x')`

Sub Plots

- `subplot(1,2,1);`  
- `plot(x,y);`  
- `xlabel('x-axis');`  
- `ylabel('y-axis');`  
- `ylabel('y-axis');`  
- `legend('sinus');`  
- `title('Sinus Function')`

- `subplot(1,2,2);`  
- `plot(x,z,'r');`  
- `ylabel('y-axis');`  
- `xlabel('x-axis');`  
- `legend('cosinus');`  
- `title('Cosinus Function')`
Sub Plots

Axis Commands

- `axis([xmin xmax ymin ymax])`
- `axis off`
  - Removes axis
- `axis on`
  - Shows axis
Systems of Linear Equations

- Suppose you have a system of linear equation
  - $2x - y + z = 8$
  - $-x + y - 2z = -9$
  - $x + y - 3z = -8$

- $A = \begin{bmatrix} 2 & -1 & 1 \\ -1 & 1 & -2 \\ 1 & 1 & -3 \end{bmatrix}$
- $b = [8; -9; -8]$

- System is defined as: $Ac = b$
- Solution is: $c = A \backslash b$
  - $c = [x \ y \ z]$

Solving Polynomials

- Let $f(x) = ax^4 + bx^3 + cx^2 + dx + e = 0$
  - $N^{th}$ order polynomial
- Define $F = [a \ b \ c \ d \ e]$ matrix
  - $N+1$ length matrix
- $y = \text{roots}(F)$ gives roots of $f(x)$

- Ex: solve $x^3 - 4x^2 + x + 6 = 0$
Evaluating Polynomials

- Evaluate polynomial at a certain ‘\(x_0\)’ point or points \((x_1, x_2, x_3, x_4\) etc)
  \[
y = \text{polyval}(F, x_0)
\]
- \(A = [x_1, x_2, x_3, x_4];\)
  \(Y = \text{polyval}(F, A)\)
- Or
  \(Y = \text{polyval}(F, [x_1, x_2, x_3, x_4])\)

Programming in MATLAB

- Variables
- Input and Output Commands
- Decision (Selections) Structures
  - If
  - Switch
- Loop (Repetition) Structures
  - For
  - While
**Variable Names**

- A valid variable name starts with a letter, followed by letters, digits, or underscores.
- A valid variable names must be written with English alphabet.
- MATLAB is case sensitive
  - *name* and *NAME* are different variables!
- Valid variable examples:
  - Student
  - student_1
  - ENVE_student
- Invalid variable examples:
  - student-1
  - student.1
  - 1.student
  - Student first

**Variable Types**

Variable Limits

<table>
<thead>
<tr>
<th>Class</th>
<th>Max value</th>
<th>Min Value</th>
<th>Bytes</th>
<th>Smallest difference</th>
</tr>
</thead>
<tbody>
<tr>
<td>logical</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>int8</td>
<td>127</td>
<td>-128</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>int16</td>
<td>32767</td>
<td>-32768</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>int32</td>
<td>2.14e+09</td>
<td>-2.14e+09</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>int64</td>
<td>9.22e+18</td>
<td>-9.22e+18</td>
<td>8</td>
<td>1</td>
</tr>
<tr>
<td>uint8</td>
<td>255</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>uint16</td>
<td>65535</td>
<td>0</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>uint32</td>
<td>4.29e+09</td>
<td>0</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>uint64</td>
<td>1.84e+19</td>
<td>0</td>
<td>8</td>
<td>1</td>
</tr>
<tr>
<td>single</td>
<td>3.40e+038</td>
<td>-3.40e+038</td>
<td>4</td>
<td>1.1755e-38</td>
</tr>
<tr>
<td>double</td>
<td>1.79e+308</td>
<td>-1.79e+308</td>
<td>8</td>
<td>2.2251e-308</td>
</tr>
</tbody>
</table>

http://www.matlabtips.com/you-must-choose/

Input and Output Commands

- `your_number=input('Write a number:')`
  - The class of “your_number” variable is double
- `name=input('Write your name:','s')`
  - The class of “name” variable is char

```matlab
>> your_number=input('Write a number:')
Write a number:25
your_number =
25

>> name=input('Write your name:','s')
Write your name:Özgür
name =
Özgür
```
Input and Output Commands

- `disp` function is used as output command in MATLAB
- `disp('MATLAB Programming')`

```
>> disp('MATLAB Programming')
MATLAB Programming
```

If Command

```
if condition
    commands
end

if condition
    commands-1
else
    commands-2
end

if condition-1
    commands1
elseif condition-2
    commands2
else
    commands3
end

>> x=9;
 >> if (x<10) disp('X is smaller than 10')
    end
    X is smaller than 10

>> x=9;
 >> if x>10 disp('X is bigger than 10')
    else disp('X is smaller than 10')
    end
    X is smaller than 10

>> x=10;
 >> if (x>10) disp('X is bigger than 10')
    elseif (x<10) disp('X is smaller than 10')
    else disp('x is equal to 10')
    end
    x is equal to 10
```
## Relational Operators

<table>
<thead>
<tr>
<th>Operator</th>
<th>Symbol</th>
</tr>
</thead>
<tbody>
<tr>
<td>equal</td>
<td>==</td>
</tr>
<tr>
<td>not equal</td>
<td>~=</td>
</tr>
<tr>
<td>greater than</td>
<td>&gt;</td>
</tr>
<tr>
<td>less than</td>
<td>&lt;</td>
</tr>
<tr>
<td>greater or equal</td>
<td>&gt;=</td>
</tr>
<tr>
<td>less or equal</td>
<td>&lt;=</td>
</tr>
</tbody>
</table>

## Logical Operators

<table>
<thead>
<tr>
<th>Operator</th>
<th>Symbol</th>
</tr>
</thead>
<tbody>
<tr>
<td>And</td>
<td>&amp;</td>
</tr>
<tr>
<td>Or</td>
<td></td>
</tr>
<tr>
<td>Not</td>
<td>~</td>
</tr>
</tbody>
</table>
If Example

- Check whether the given number is odd or even?

```matlab
number=23;
if (rem(number,2) == 1) disp('number is odd')
else disp('number is even')
end
```
or

```matlab
if (rem(number,2) ~= 1) disp('number is even')
else disp('number is odd')
end
```

If Example

- Using If structure in MATLAB
- Display the letter of a student with respect to his/her grade

<table>
<thead>
<tr>
<th>Letter</th>
<th>Grade range</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>90 – 100</td>
</tr>
<tr>
<td>B</td>
<td>80 – 89</td>
</tr>
<tr>
<td>C</td>
<td>70 – 79</td>
</tr>
<tr>
<td>D</td>
<td>60 – 69</td>
</tr>
<tr>
<td>F</td>
<td>0 – 59</td>
</tr>
</tbody>
</table>
Solution

```matlab
>> grade = 85;
if grade >= 90, letter = 'A'
el elseif grade >= 80, letter = 'B'
el elseif grade >= 70, letter = 'C'
el elseif grade >= 60, letter = 'D'
el else letter = 'F'
end

letter = B
```

Switch

```plaintext
switch condition
  case x1, commands-1
  case x2, commands-2
  case x3, commands-3
  ...
  case xn, commands-n
  otherwise commands-n+1
end
```
Switch Example

- Dice game

- Roll 2 dices by using `rand` and `floor` functions in Matlab.
  - \( d1 = \text{floor}(6*\text{rand}) + 1; \)
  - \( d2 = \text{floor}(6*\text{rand}) + 1; \)
- You will win if sum of 2 dices are 7, lose otherwise.
- Use `switch` to check win condition.

```
>> d1 = floor(6*rand)+1;
   d2 = floor(6*rand)+1;
   switch d1+d2
     case 7
        disp( '7 - You win' );
     otherwise
        disp( 'You lose' );
     end
   You lose
```
For

- In for loop, number of iterations are known.
- Loop variable is a vector.

```matlab
>> for i=1:5
    disp('Matlab Programming')
end
Matlab Programming
Matlab Programming
Matlab Programming
Matlab Programming
Matlab Programming
```

For Example

- Calculate \( \sum_{i=1}^{10} i^2 \)
- using `for` loop

```matlab
>> sumA=0;
>> for i=1:10
    sumA=sumA+i*i;
end
>> sumA
sumA =
    385
```
While

- Loop is executed as long as condition gives true.
- Programmer should check condition to avoid infinite loops.

```
while condition
    commands
end
```

While Example

- Create $\sin(x)$, $\sin(2x)$, $\sin(3x)$, $\sin(4x)$ graphs by using 
  `subplot` and `while` commands.

```matlab
x=linspace(-2*pi,2*pi,100);
i=1;
while i<=4
    subplot(2,2,i), plot(x,sin(i*x)),
    title(['Sinus ',num2str(i),'x']),
    i=i+1;
end
```
While Example