MATLAB

Building a graphical user interface

note:
The GUI is contained in two files:
- figure file (.fig): Contains the graphical layout information.
- m-file (.m): Contains the main GUI <u>function</u> & some subfunctions.

• To open the GUIDE window:
write >> <u>guide</u> in the command window

• MATLAB class called <u>uicontrol</u> → Contains most of these GUI objects

• the name of each object is the value of the Tag property.
& it is a unique value for each object.

note:

Callback functions: each object has a callback function & is executed when an object is activated.

for example, a button is activated when the user presses and releases it.

the name of the callback function has the form:

TagValue_Callback

~~note:~~

important note:

the handle of the object: is the address of the object in the memory

note:

The functions str2double & num2str are frequently used when coding in the m-file.

- m-file has code contains:

untitledTool → the main function that creates the tool itself.

$\boxed{\text{untitledTool\_OpeningFcn}}$ → ~~this~~ this function is executed once the program is run. it is like constructors in C++.
just before the tool is made visible.

untitledTool_OutputFcn → advanced function. ignore it.

+

$\boxed{\text{objectName\_Callback}}$ → executed when the object is activated.

- The used functions for the programmer

$\boxed{\text{in the GUIDE window}}$

To edit an object → use Property inspector window by double-clicking on it.

note: the Tag property is an important property.

important note: during writing the code, we deal with Property-Value pairs. it with have the notation that the name of the property is capitalized.

in the m-file

To Communicate (edit or get value) with a GUI object, we need to Know the handle of the object.

① To Know the handle of an object:

to Know the handle of an object, we search for the handle of the object whose <Property> matches the specified <value>

→ To do that we use the handles structure:

handles.xSlider → returns the handle of the slider with Tag "xSlider"

② Communicating with the GUI object:

To do that we use the get & set Command

→ The general form of the get Command:

<var> = get(<handle>, <Property name>);

ex.

get(htext,'String');

returns the value of the String Property of the text object

→ The general form of the set Command:

set(<handle>, <Property name>, <Property value>);

ex.

set(htext, 'String', num2str(x));

set(hslider, 'Value', 9);

Operations on Vectors & matrices

working with matrices makes you encounter two forms for arithmetic operations on matrices & vectors:

matrix arithmetic operator scalar or matrix arithmetic operator matrix

Types of arithmetic operations in matlab:

in order 1) matrix operations

operations that follow the rules of Linear algebra.

special case:

in the case of matrix ± scalar, the scalar will be treated as a matrix of the same size as the other one with all elements equal the scalar.

Arithmetic operators:

\+ − * / ^

operands:

the operand could be a matrix or scalar as the two forms show above.

2) array operations

operations that execute element by element.

Arithmetic operators:

we add (.) before the array operators to distinguish them from the matrix operators.

$\boxed{+ \quad - \quad .* \quad ./ \quad .\hat{}}$

operands:

the operand could be a matrix or scalar as the two forms show above.

File input/output

The basic idea

using functions to make some operations on files of different types (which use different filename extensions)

types of operations: writing, appending, reading

our current types of files: .dat or .txt

the functions:

1) using data of a matrix format (the same kind of data on each line and in the same format on every line) & files of types .dat or .txt

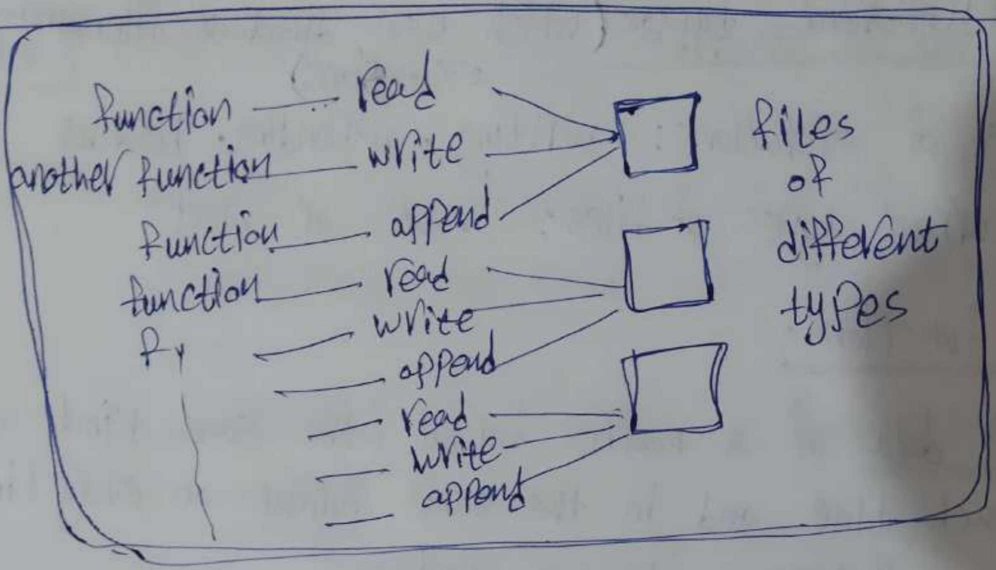
for reading, use Load function
(read from the file, then create a matrix of the same name)

syntax: Load filename.fileextension

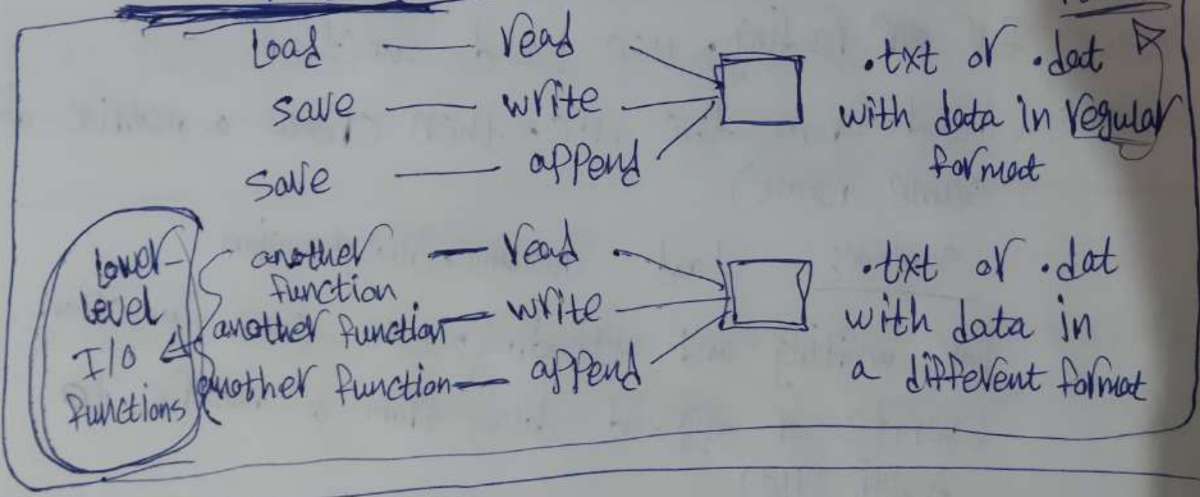
for writing and appending, use save function
(write or append data from a matrix to Ascii file)

syntax: save filename.fileextension matrixvariablename -ascii -append

2) using data in a different format & files of types .txt or .dat,
then we use lower-level file I/O functions.



in our case:



matlab functions

<u>syntax</u>

type filename.fileextension → display the content of the file.

who → Lists the variables currently in the workspace.

whos → Lists the variables currently in the workspace and their sizes and types.

get(h) → returns all properties of the graphics object identified
→ only graphics object

- get(h,'PropertyName') → returns the value of the property of the graphics object identified.

Definition

vectorization → means turning the code into vectors instead of using loops.

Accessing elements of a matrix

we use () for indexing while we use [] for the array itself.

the general form

mymat(□ , □)

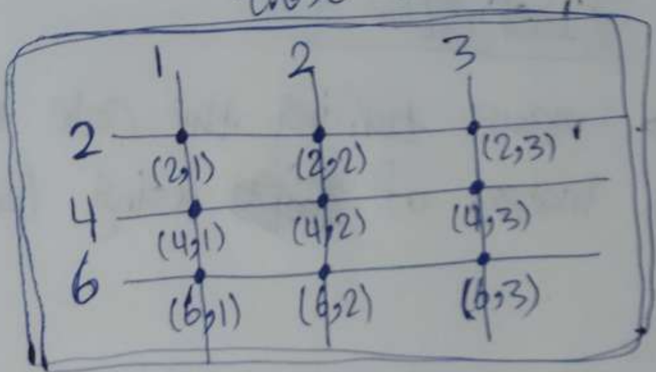
this could be a scalar or matrix so that each value in the first square □ matches all the values in the second square □.

<u>example</u>

mymat([2,4,6],[1,2,3])

= mymat( )

those indices



not only (2,1), (4,2), (6,3) as I was expecting before.

. function types → file function
→ anonymous function

. function function: is the function that accepts another function as input argument.

$\boxed{\text{function name \& function handle}}$

$\boxed{\text{function name}}$ → is used <u>only</u> to call the function.

<u>function handle:</u> is a matlab data type where variable of this type represents pointer to a function.

- we use @ operator to get the function handle.

$\boxed{\text{function handle variable}}$ → is used to call the function by using ()
→ or used as a variable like any other variable (without using ())

To be studied in the future:

- <style-option string> ⟷ 'Linewidth', 2 وحاجات كدة
- figure – subplot – axes handles – set ⟶ $\boxed{\text{essential matlab for engineers and scientists}}$
- Plot & set & بتعمل إيه → بت clear ال current figure وترسم الجديد
- updating the data of a plot in matlab
- axes & axis

Basic animation

• main concept:

same way as a flip_book animation which is a series of still pictures with small differences between them are rapidly displayed to give the illusion of ~~[illegible]~~ motion.

- Command window Prompt → >>

The Matlab returns to the Command window Prompt when the run is over.

- To close the run → ctrl + C (to return to matlab Prompt)

- Colon operator (:) & Linspace (start, end, # of Points) are similar

the interval [start, end]

Plot Command

- general form:

Plot(<vector of x-values>, <vector of y-values>, <style-option string>)

• multiple plots in the same axes:

using plot command:

① use plot with multiple arguments

```
plot(x1vec, y1vec, stylestring1, ...
     x2vec, y2vec, stylestring2, ...
     :
     xnvec, ynvec, stylestringN)
```

② use hold command between multiple plot commands

using fplot command:

① use hold command between multiple fplot commands

note: using fplot with multiple arguments is sth that doesn't exist.

trial division algorithm for Primality test

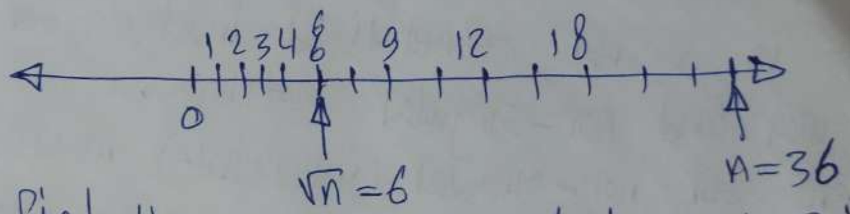
the concept:

if we have a number (n) that is not a prime number, then it can be factored into $n = a * b$.

$$\because \quad n = (\sqrt{n})(\sqrt{n})$$
$$\therefore \quad \min(a,b) \leq \sqrt{n}$$

Then if (n) is not a prime number, we must find at least one factor less than or equal to $\sqrt{n}$.

Ex. $36 \rightarrow \{1,36\}, \{2,18\}, \{3,12\}, \{4,9\}, \{6,6\}, \{36,1\}, \{18,2\}, \{12,3\}, \{9,4\}$



we always find the smallest element in each pair less than or equal to $\sqrt{n}$.

the algorithm:

1) input the number (n)
2) for x from 2 to $\sqrt{n}$, if any x divides n, then n is composite else n is prime.

(:=) is a mathematical notation or convention that means (is defined as equal to) which can be used like many other conventions like ($\stackrel{def}{=}$) or ($\equiv$)

You know the convention you are working with from the context (as explained or shown in that specific context)

~~note:~~

~~In matlab, we have two functions to calculate the inverse:~~

~~pinv → calculates the Moore-Penrose pseudo inverse (this is a more generalized inverse for both singular and non-singular matrices. where for non-singular (invertible) matrices, the matrix inverse = the matrix pseudo inverse)~~

~~inv → calculates the inverse of invertible square matrices. (because only some of the square matrices are invertible)~~

Time complexity of an algorithm

<u>definition:</u>

Time complexity of an algorithm is the amount of time required to run an algorithm.

<u>time complexity analysis:</u>

First, we determine the running time of an algorithm as a function of the input size → $T(n)$
then find the upper bound using big-O notation (for asymptotic behavior)

<u>Calculating the running time:</u>

we assign a time constant for each code fragment then the number of times of excecution for this fragment, which is a function of the input size n,
(in other words, the running time is a function of # of operations and # of operations is a function of the input size)
then to calculate the total running time
assuming:
1) large-size input
2) worst case scenario

we follow these general rules:
1) Running time = $\sum$ Running time of all fragments
2) for If-else statement, choose the worst case

~~Algorithm Performance~~

algorithm running time:

$$O(1) < O(\log n) < O(n) < O(n \log n) < O(n^2)$$

logarithmic complexity:

$\log_2(8) \xrightarrow[\text{form}]{\text{another}} 2^3 = 8$ → n

So this means that how many times you have to divide n (8) by the base (2) to get to 1

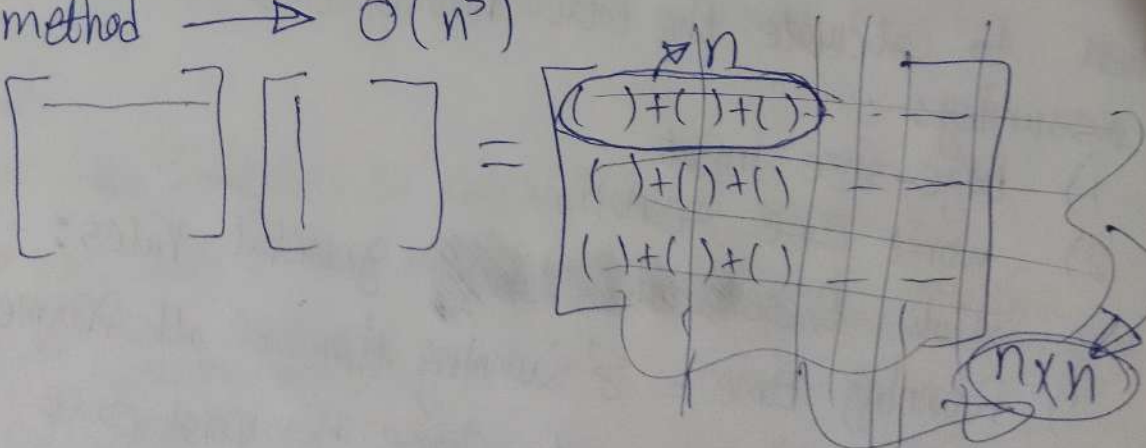
or

how many times you have to multiply 2 (base with 1 to get to n (8).

matrix multiplication algorithm time complexity

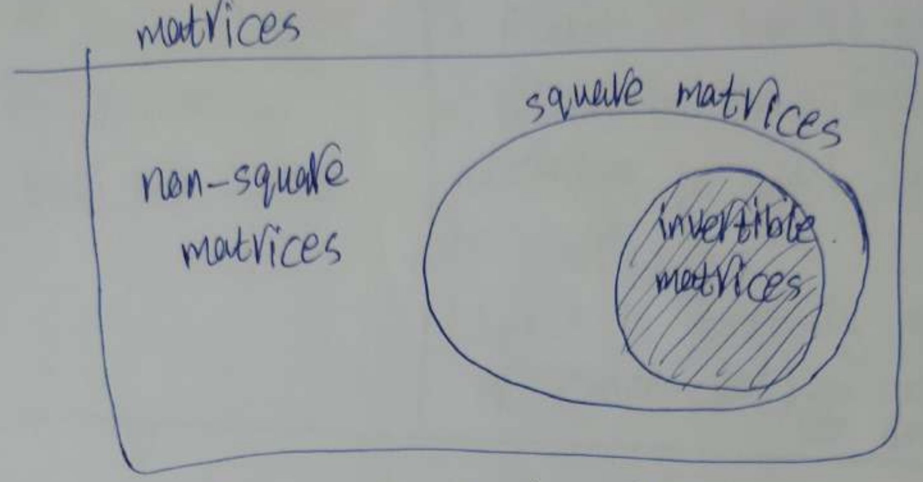
Naive method → $O(n^3)$

as



inverse and Pseudo inverse

in mathematics,



- non-square matrices are not invertible.
- invertible matrices are only some of the square matrices that verifies some conditions.
- we have Pseudo inverse which is a more generalized inverse for both invertible and non-invertible (singular) matrices

where for non-singular (invertible) matrices,

the matrix inverse = the matrix Pseudo inverse.

in matlab,

we have two functions to calculate the inverse:

Pinv → calculates the Moore-Penrose Pseudo inverse for any matrix.

inv → calculates the inverse for square matrices.

(all square matrices even singular ones)

$\begin{bmatrix} inf & inf & - \\ \vdots & \vdots & \end{bmatrix}$