UNIT-IV

Topics:-

Manipulating Text

Writing to a text file, Reading from a text file, Randomizing and sorting a list, searching a list.

GUI Interface:-

Attaching buttons to actions, Getting Input, Setting Output..

Manipulating Text

"Manipulating text" in MATLAB refers to a variety of operations that involve **working with string data or text files**, such as reading and writing files, rearranging data, or searching within datasets. These tasks are essential for data analysis, file handling, and automation in programming.

1. Writing to a Text File

Definition:
Writing to a text file means **storing information (such as text, numbers, or data) from a MATLAB program into an external file**, typically with a `.txt` extension. This allows the data to be saved for future use, shared with others, or transferred between systems.

Explanation:
In MATLAB, this is done using the `fopen()` function to create or open the file, `fprintf()` to write formatted data into the file, and `fclose()` to properly close and save the file. Proper file handling ensures data integrity and prevents corruption.

Functions Used:

- `fopen()` – Opens a file for writing or appending
- `fprintf()` – Writes formatted data to the file
- `fclose()` – Closes the file

Example:

```
fileID = fopen('output.txt', 'w');          % Open or create file for writing
fprintf(fileID, 'Hello, World!\n');         % Write string to file
fclose(fileID);                             % Close the file
```

Writing data to a text file means creating a file with data that will be saved on a computer's secondary memory such as a hard disk, CD-ROM, network drive, etc. fprintf() function is used to write data to a text file in MATLAB. It writes formatted text to a file exactly as specified. The different escape sequences used with fprintf() function are:

```
\n  : create a new line
\t  : horizontal tab space
\v  : Vertical tab space
\r  : carriage return
\\  : single backslash
\b  : backspace
%%  : percent character
```

The format of the output is specified by formatting operators. The formatSpec is used to format ordinary text and special characters. A formatting operator starts with a percent sign% and ends with a conversion sign. The different format specifiers used with fprintf() function are:

```
%d or %i: Display the value as an integer
%e : Display the value in exponential format
%f : Display floating point number
%g : Display the value with no trailing zeros
%s : Display string array (Unicode characters)
%c : Display single character(Unicode character)
```

Now let's start writing data to a text file. Before writing to a file, we need to open the text file using fopen() function. To open a file, the syntax is:

f=fopen(File_name, Access_mode)
Here,

fopen() function accepts two arguments:

name of the file or File_identifier.
type of access mode in which the file is to be open. We need to write data, so the access mode can be 'w', 'w+', 'a', 'a+'.

2. Reading from a Text File

Definition:
Reading from a text file is the process of **importing or retrieving text/data stored in a file back into MATLAB** so it can be analyzed, processed, or displayed.

Explanation:
Using file I/O functions like `fopen()` (to open the file), `fgets()`, `fscanf()`, or `fread()` (to read the contents), and `fclose()` (to close the file), you can extract text line-by-line or in full. Reading data from external sources is a key part of data-driven programming.

Functions Used:

- `fopen()`
- `fgets()` / `fscanf()` / `fread()` – Reads content from the file
- `fclose()`

Example:

```
fileID = fopen('output.txt', 'r');          % Open file for reading
line = fgets(fileID);                       % Read one line
disp(line);                                 % Display the line
fclose(fileID);                             % Close the file
```

3. Randomizing a List

Definition:
Randomizing a list means **shuffling the order of elements in an array or list randomly** so that no predictable sequence exists. This is often used in simulations, games, testing, and experiments to eliminate bias.

Explanation:
MATLAB provides the function `randperm(n)` to generate a random permutation of integers from 1 to `n`. You can use this random index order to rearrange elements in any list or dataset.

Function Used:

- `randperm(n)` – Returns a row vector containing a random permutation of integers from 1 to `n`.

Example:

```
list = [1, 2, 3, 4, 5];
randomOrder = list(randperm(length(list)));   % Randomized version
disp(randomOrder);
```

4. Sorting a List

Definition:
Sorting a list refers to the **process of arranging the elements of an array or dataset in a specific order**, typically ascending or descending. Sorting is crucial for searching, reporting, and organizing data.

Explanation:
MATLAB's `sort()` function sorts numeric or textual data. By default, it sorts in ascending order, but you can specify descending order as well. Sorting helps with tasks like ranking, grouping, and preparing data for visualization.

Function Used:

- `sort()` – Sorts data in ascending or descending order

Example:

```
list = [42, 7, 19, 3];
sortedAsc = sort(list);                % Ascending sort
sortedDesc = sort(list, 'descend');    % Descending sort
```

5. Searching in a List

Definition:
Searching in a list is the action of **finding the presence or position of a particular value or pattern** within a dataset. It helps in data lookup, filtering, and decision-making processes.

Explanation:
MATLAB uses `find()` to return the indices of elements that satisfy a condition and `ismember()` to check if a value exists within a list. Searching is used to detect specific entries or patterns in large datasets.

Functions Used:

- `find()` – Returns the index of elements that meet a condition
- `ismember()` – Checks if a value exists in a list

Examples:

```
list = [10, 20, 30, 40];
index = find(list == 30);    % Returns index of 30
disp(index);
```

```
isPresent = ismember(25, list);  % Returns 0 (false) since 25 is not in the list
disp(isPresent);
```

Text Manipulation in MATLAB

Operation	Function(s)	Purpose
Writing to a file	`fopen`, `fprintf`, `fclose`	Save text or data from MATLAB to a `.txt` file
Reading a file	`fopen`, `fgets`, `fclose`	Load or import data from a `.txt` file into MATLAB
Randomizing a list	`randperm()`	Shuffle or mix the order of elements randomly
Sorting a list	`sort()`	Arrange elements in ascending or descending order
Searching a list	`find()`, `ismember()`	Locate a value or its position in a list or check if a value exists

GUI Interface

A **Graphical User Interface (GUI)** in MATLAB allows users to interact with programs through visual components like buttons, text boxes, sliders, and more — rather than writing code in the command window. GUIs are built using **App Designer** (modern method) or **GUIDE** (legacy method).

In MATLAB, GUI (Graphical User Interface) development is used to build **interactive applications** where users can provide input, control processes, and view output — all visually. This allows for **user-friendly applications** without requiring the user to interact with the command line.

The most modern and powerful way to build GUIs in MATLAB is using **App Designer**, which provides a drag-and-drop interface and auto-generates an object-oriented code structure.

Instead of writing code in the command window, users can **click buttons**, **enter text**, **move sliders**, **select from menus**, and **see plots** — all inside a window (figure).

MATLAB provides **three main ways** to create a GUI:

1. **App Designer** (modern method)
   - Drag-and-drop interface to design apps.
   - Automatically generates the underlying code.
2. **Programmatic GUI** (writing code manually)
   - Use `uicontrol`, `figure`, and other functions to create GUI elements.
3. **GUIDE** (Graphical User Interface Development Environment)
   - Old drag-and-drop tool (deprecated after MATLAB R2019b).

✨ Why Use a GUI in MATLAB?

- To make your programs easier to use.
- To build interactive tools, simulations, or dashboards.
- To let non-programmers use your algorithms.

Here's a breakdown of the core concepts:

Attaching Buttons to Actions (Event-Driven Programming)

Definition:

In GUI design, attaching a button to an action refers to the process of **binding an event listener** (typically a *callback function*) to a GUI component (e.g., a button). When the user clicks the button, MATLAB invokes the **callback function**, triggering some functionality.

Attaching buttons to actions means **linking a button (like "Submit", "Calculate", "Start") to a specific function or block of code**. When the user clicks the button, MATLAB executes the associated callback function.

How It Works:

- Every button has a **Callback Function** — this function contains the action to be performed.
- In **App Designer**, you can double-click a button to auto-generate the callback.

Each UI component in App Designer is an object with **properties** and **methods**. When a button is clicked:

- An **event** (like `ButtonPushed`) is fired.
- The corresponding **callback method** is executed.

Example (App Designer):

```
% Inside the button callback
function ButtonPushed(app, event)
    app.Label.Text = 'Button was clicked!';
end
```

Advanced Usage:

```
% Button callback function inside the app class
methods (Access = private)
```

```
    function StartButtonPushed(app, event)
        % Reading from another component
        data = app.InputField.Value;

        % Performing computations or operations
        result = advancedFunction(data);

        % Output
        app.OutputLabel.Text = sprintf('Result: %.2f', result);
    end

end
```

Why It's Important:

- Without a callback, the button would just sit there and do nothing.
- A callback brings the GUI to life by adding **interactivity**.

Getting Input

Definition:

Getting input means **retrieving information** that the user has entered into an input field (like a text box or an editable field) inside the GUI.

Why It's Important:

- A GUI is interactive — it should react based on **what users provide**.
- Inputs allow users to **customize** the program behavior (e.g., type a number, name, choice).

How It Works:

- In MATLAB, the user types something into an `edit` component (text box).
- To read that information, you use the `get` function with the `'String'` property.

Example Code:

```
inputBox = uicontrol('Style', 'edit', 'Position', [50 150 200 30]);

% Later, inside a callback:
userInput = get(inputBox, 'String');
```

Key Points:

- `get(component, 'String')` pulls the **current value** from the input box.
- Data retrieved is usually a **string**; you might need to convert it if you expect numbers (`str2double`).

Setting Output

Definition:

Setting output means **displaying results or messages** back to the user after some action, like after clicking a button or entering data.

Why It's Important:

- Users need **feedback** to know what happened — did their action work? Was it correct?
- Output can be shown as text, graphics, plots, etc.

How It Works:

- Typically, you create a `text` component (non-editable).
- Then you update the text dynamically by using the `set` function.

Example Code:

```
outputLabel = uicontrol('Style', 'text', 'Position', [50 90 200 30], 'String', '');

% Later, inside a callback:
set(outputLabel, 'String', ['You entered: ' userInput]);
```

Key Points:

- `set(component, 'String', newText)` **updates** what is displayed on the GUI.
- The output is immediately visible to the user.

How All Three Concepts Work Together

Step	Action in GUI	What Happens Behind
1	User clicks a button	Button triggers its callback function
2	Callback reads input	`get(inputBox, 'String')` fetches user's data
3	Callback updates output	`set(outputLabel, 'String', new text)` displays response