KARPAGAM ACADEMY OF HIGHER EDUCATION

(Deemed to be University)
(Established Under Section 3 of UGC Act 1956)
Eachanari (po), Coimbatore-21

DEPARTMENT OF CS, CA & IT

LECTURE PLAN

SUBJECT NAME: PROGRAMMING IN MATLAB **BATCH: 2017 - 2020**

SUBJECT CODE: 17CTU304B **SEMESTER: II**

STAFF: Dr.D.SHANMUGA PRIYAA **CLASS: II B.Sc. CT**

S.No.	Lecture Duration (Hours)	Topics to be Covered	Support Materials
Unit – I			
1.	1	Components of Computer	W1,W6
2.	1	Working with numbers	W2
3.	1	Machine Code	W3
4.	1	Software hierarchy.	W1
5.	1	Matlab Architecture	W1
6.	1	Recapitulation and Discussion of important questions	
		Total No. of Hours Planned for Unit-I	**6**
Unit – II			
S.No.	**Lecture Duration (Hours)**	**Topics to be Covered**	**Support Materials**
1.	1	MATLAB Windows, A First Program	S1:9–5, W3,W5
2.	1	Expressions	S2: 10-17, W5
3.	1	Constants	S2: 14, W4
4.	1	Variables and assignment statement	S1: 16-18, S2: 6-9, W5
5.	1	Arrays	S1: 35-55,S2: 30-31, W5
6.	1	Recapitulation and Discussion of important questions	
		Total No. of Hours Planned for Unit-II	**6**
Unit – III			
S.No.	**Lecture Duration (Hours)**	**Topics to be Covered**	**Support Materials**
1.	1	Basic plotting, Built in functions	S1: 133-139, S1: 13-16, S2: 14-17, W5
2.	1	Generating waveforms, Sound replay	S2: 393-394, W4
3.	1	Load and save	S1: 111-113, W4
4.	1	Procedure and Functions: Arguments and return values	S1: 219-244, W5
5.	1	M-files, Formatted console input-output	S1: 97-110, R2: 27-29, W5 R3: 5,
6.	1	String handling	S1: 53-54, R2: 125-133, W5
7.	1	Recapitulation and Discussion of important questions	
		Total No. of Hours Planned for Unit-III	**7**

Unit – IV			
S.No.	**Lecture Duration (Hours)**	**Topics to be Covered**	**Support Materials**
1.	1	Conditional statements **if** Statement Representing Logical True and False	S1: 182-189, S2: 82-86, W5
2.	1	**if-Else** Statement Nested **if-Else** Statements	S2: 87-88
3.	1	The **Switch** Statement, **Menu** Function	S1: 190-200,S2:93-96, W5
4.	1	Repetition statements For and Nested For	S2: 110-129,W2
5.	1	While and Multiple Conditions in while	S2: 143-150
6.	1	Recapitulation and Discussion of important questions	
		Total No. of Hours Planned for Unit-IV	**6**
Unit – V			
S.No.	**Lecture Duration (Hours)**	**Topics to be Covered**	**Support Materials**
1.	1	**Manipulating Text** Writing to a text file	S2: 59-62,W4
2.	1	Reading from a text file	S2: 61-63,W1
3.	1	Randomising, Sorting a list	S2: 372-378, W4
4.	1	searching a list	S2: 382-392, W4
5.	1	**GUI Interface** Attaching buttons to actions	R4: 487-488
6.	1	Getting Input	S2: 409-410,W4
7.	1	Setting Output	S2: 409-411,W4
8.	1	Recapitulation and Discussion of Important Questions	
9.	1	Discussion of Previous ESE question papers	
10.	1	Discussion of Previous ESE question papers	
11.	1	Discussion of Previous ESE question papers	
		Total No. of Hours Planned for Unit-V	**11**

Total No. of Hours for the Course: 45

Suggested Readings

S1. MATLAB: An Introduction with Applications, by Amos Gilat, 2nd edition, Wiley, 2004.

S2. C.B. Moler, Numerical Computing with MATLAB, SIAM, 2004.

Reference Books

R1. MATLAB® An Introduction with Applications Fourth Edition Amos Gilat

R2. MATLAB Tutorial

R3. MATLAB Commands and Functions Dr. Brian Vick Mechanical Engineering Department Virginia Tech

R4. Matlab Programming for Engineers, by Stephen J Chapman, 4th edition, 2012

Web Sites

W1. http://oer.nios.ac.in/wiki/index.php/COMPUTER_AND_ITS_COMPONENTS

W2. https://en.wikipedia.org/wiki/MATLAB

W3. https://en.wikipedia.org/wiki/M_code

W4.http://faculty.washington.edu/lum/website_professional/matlab/tutorials/Matlab_Tutorial_Beginner/matlab_tutorial_beginner.pdf

W5. https://in.mathworks.com/help/matlab/learn_matlab/expressions.html

W6. https://www.tutorialspoint.com/matlab/

KARPAGAM ACADEMY OF HIGHER EDUCATION

(Deemed to be University)

(Established Under Section 3 of UGC Act 1956)

Eachanari (po), Coimbatore-21

Semester – III

17CTU304B **PROGRAMMING IN MATLAB** **3H – 3C**

Instruction Hours / week: L: 3 T: 0 P: 0 **Marks:** Int: **40** Ext: **60** Total: **100**

SCOPE

A student who successfully completes this course should be able to learn how to use MATLAB, learn how to program in MATLAB, ability to create a computer program to solve problems in science and engineering.

OBJECTIVE

- To learn fundamental programming concepts using a block-structured language (MATLAB).
- To learn General problem-solving techniques, including the concept of step-wise refinement applied to the development of algorithms.

UNIT-I

Introduction to Programming: Components of a computer, working with numbers, Machine code, Software hierarchy.

UNIT-II

Programming Environment: MATLAB Windows, A First Program, Expressions, Constants, Variables and assignment statement, Arrays.

UNIT-III

Graph Plots: Basic plotting, Built in functions, Generating waveforms, Sound replay, load and save. Procedures and Functions: Arguments and return values, M-files, Formatted console input-output , String handling,

UNIT-IV

Control Statements: Conditional statements: If, Else, Else-if, Repetition statements: While, for loop

UNIT-V

Manipulating Text: Writing to a text file, Reading from a text file, Randomising and sorting a list, searching a list. **GUI Interface**: Attaching buttons to actions, Getting Input, Setting Output

Suggested Readings

1. Amos Gilat (2004). MATLAB: An Introduction with Applications(2nd ed). New Delhi: Wiley.

2. C.B. Moler (2004). Numerical Computing with MATLAB. SIAM.

ESE Pattern	
Part – A (Online)	20 * 1 = 20
Part – B	5 * 2 = 10
Part – C (Either or)	5 * 6 = 30
Total	60 marks

Faculty **HOD**

Unit I

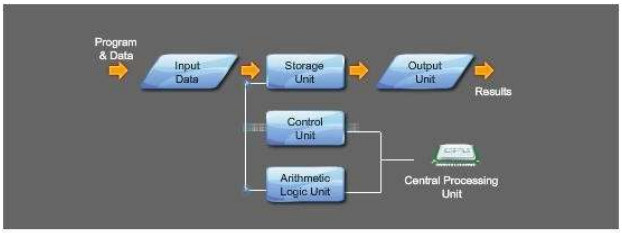
Syllabus

Introduction to Programming: Components of a computer, working with numbers, Machine code, Software hierarchy.

Components of Computer

A computer system consists of mainly four basic units; namely input unit, storage unit, central processing unit and output unit. Central Processing unit further includes Arithmetic logic unit and control unit, as shown in the figure:. A computer performs five major operations or functions irrespective of its size and make. These are

- it accepts data or instructions asinput,
- it stores data andinstruction
- it processes data as per theinstructions,
- it controls all operations inside a computer,and
- itgives results in the form ofoutput.



Functional Units:

a. Input Unit: This unit is used for entering data and programs into the computer system by the user forprocessing.

Basic Computer Organisation

b. Storage Unit: The storage unit is used for storing data and instructions before and after processing.

c. Output Unit: The output unit is used for storing the result as output produced by the computer afterprocessing.

d. Processing: The task of performing operations like arithmetic and logical operations is called processing. The Central Processing Unit (CPU) takes data and instructions from the storage unit and makes all sorts of calculations based on the instructions given and the type of data provided. It is then sent back to the storage unit. CPU includes Arithmetic logic unit (ALU) and control unit(CU)

Arithmetic Logic Unit: All calculations and comparisons, based on the instructions provided, are carried out within the ALU. It performs arithmetic functions like addition, subtraction, multiplication, division and also logical operations like greater than, less than and equal to etc.

• Control Unit: Controlling of all operations like input, processing and output are performed by control unit. It takes care of step by step processing of all operations in side the computer.

Memory

Computer's memory can be classified into two types; primary memory and secondary memory

RAM

a. Primary Memory can be further classified as **RAM andROM**.

• RAM or Random Access Memory is the unit in a computer system. It is the place in a computer where the operating system, application programs and the data in current use are kept temporarily so that they can be accessed by the computer's processor. It is said tobe 'volatile' since its contents are accessible only as long as the computer is on. The contents of RAM are no more available once the computer is turnedoff.

ROM or Read Only Memory is a special type of memory which can only be read and contents of which are not lost even when the computer is switched off. It typically contains manufacturer's instructions. Among other things, ROM also stores an initial program called the 'bootstrap loader' whose function is to start the operation of computer system once the power is turned on.

b. SecondaryMemory

RAM is volatile memory having a limited storage capacity. Secondary/auxiliary memory is storage other than the RAM. These include devices that are peripheral and are connected and controlled by the computer to enable permanent storage of programs and data.

• CD ROM

Secondary storage devices are of two types; magnetic and optical. Magnetic devices include hard disks and optical storage devices are CDs, DVDs, Pen drive, Zip drive etc.

• HardDisk

Hard disks are made up of rigid material and are usually a stack of metal disks sealed in a box. The hard disk and the hard disk drive exist together as a unit and is a permanent part of the computer where data and programs are saved. These disks have storage capacities ranging from 1GB to 80 GB and more. Hard disks are rewritable.

• CompactDisk

Compact Disk (CD) is portable disk having data storage capacity between 650-700 MB. It can hold large amount of information such as music, full-motion videos, and text etc. CDs can be either read only or read writetype.

CD Drive

• Digital VideoDisk

Digital Video Disk (DVD) is similar to a CD but has larger storage capacity and enormous clarity. Depending upon the disk type it can store several Gigabytes of data. DVDs are primarily used to store music or movies and can be played back on your television or the computer too. These are notrewritable.

Hard Disk

Input / Output Devices:

These devices are used to enter information and instructions into a computer for storage or processing and to deliver the processed data to a user. Input/Output devices are required for users to communicate with the computer. In simple terms, input devices bring information INTO the computer and output devices bring information OUT of a computer system. These input/output devices are also known as peripherals since they surround the CPU and memory of a computersystem.

Input Devices

An input device is any device that provides input to a computer. There are many input devices, but the two most common ones are a keyboard and mouse. Every key you press on the keyboard and every movement or click you make with the mouse sends a specific input signal to the computer.

Keyboard

• **Keyboard**: The keyboard is very much like a standard typewriter keyboard with a few additional keys. The basic QWERTY layout of characters is maintained to make it easy to use the system. The additional keys are included to perform certain special functions. These are known as function keys that vary in number from keyboard tokeyboard.

• **Mouse**: A device that controls the movement of the cursor or pointer on a display screen. A mouse is a small object you can roll along a hard and flat surface. Its name is derived from its shape, which looks a bit like a mouse. As you move the mouse, the pointer on the display screen moves in the samedirection.

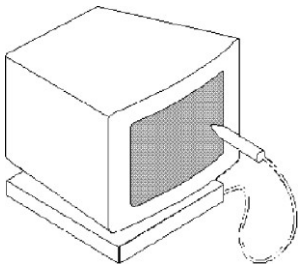
• **Trackball**: A trackball is an input device used to enter motion data into computers or other electronic devices. It serves the same purpose as a mouse, but is designed with a moveable ball on the top, which can be rolled in anydirection.

• **Touchpad**: A touch pad is a device for pointing (controlling input positioning) on a computer display screen. It is an alternative to the mouse. Originally incorporated in

laptop computers, touch pads are also being made for use with desktop computers. A touch pad works by sensing the user's finger movement and downward pressure.

• **Touch Screen:** It allows the user to operate/make selections by simply touching the display screen. A display screen that is sensitive to the touch of a finger or stylus. Widely used on ATM machines, retail point-of-sale terminals, car navigation systems, medical monitors and industrial control panels.

• **Light Pen**: Light pen is an input device that utilizes a light-sensitive detector to select objects on a display screen.



• **Magnetic ink character recognition (MICR)**: MICR can identify character printed with a special ink that contains particles of magnetic material. This device particularly finds applications in bankingindustry.

• **Optical mark recognition (OMR)**: Optical mark recognition, also called mark sense reader is a technology where an OMR device senses the presence or absence of a mark, such as pencil mark. OMR is widely used in tests such as aptitudetest.

• **Bar code reader**: Bar-code readers are photoelectric scanners that read the bar codes or vertical zebra strips marks, printed on product containers. These devices are generally used in super markets, bookshopsetc.

Scanner

Scanner is an input device that can read text or illustration printed on paper and translates the information into a form that the computer can use. A scanner works by digitizing an image.

Output Devices:

Output device receives information from the CPU and presents it to the user in the desired from. The processed data, stored in the memory of the computer is sent to the output unit, which then converts it into a form that can be understood by the user. The output is usually produced in one of the two ways – on the display device, or on paper (hardcopy).

•**Monitor**: is often used synonymously with "computer screen" or "display." Monitor is an output device that resembles the television screen (fig. 1.8). It may use a Cathode Ray Tube (CRT) to display information. The monitor is associated with a keyboard for manual input of characters and displays the information as it is keyed in. It also displays the program or application output. Like the television, monitors are also available in

differentsizes.

• **Printer**: Printers are used to produce paper (commonly known as hard copy) output. Based on the technology used, they can be classified as Impact or Non-impactprinters.

Impact printers use the typewriting printing mechanism wherein a hammer strikes the paper through a ribbon in order to produce output. Dot-matrix and Character printers fall under this category.

Non-impact printers do not touch the paper while printing. They use chemical, heat or electrical signals to etch the symbols on paper. Inkjet, Deskjet, Laser, Thermal printers fall under this category of printers.

• **Plotter**: Plotters are used to print graphical output on paper. It interprets computer commands and makes line drawings on paper using multi colored automated pens. It is capable of producing graphs, drawings, charts, maps etc. • **Facsimile (FAX)**: Facsimile machine, a device that can send or receive pictures and text over a telephone line. Fax machines work by digitizing animage.

• **Sound cards and Speaker(s)**: An expansion board that enables a computer to manipulate and output sounds. Sound cards are necessary for nearly all CD-ROMs and have become commonplace on modern personal computers. Sound cards enable the computer to output sound through speakers connected to the board, to record sound input from a microphone connected to the computer, and manipulate sound stored on adisk.

WORKING WITH NUMBERS

MATLAB(matrixlaboratory) is a multi-paradigm numerical computing environment and fourth-generation programming language. A proprietaryprogramming languagedeveloped by Mathworks, MATLAB allows matrix manipulations, plotting of functions and data, implementation of algorithms, creation of user interfaces, and interfacing with programs written in other languages, including C, C++, C#, Java, Fortran andPython.

Although MATLAB is intended primarily for numerical computing, an optional toolbox uses the MuPADsymbolic engine, allowing access to symboliccomputing abilities. An additional package, Simulink, adds graphical multi-domain simulation and model-based design fordynamicand embeddedsystems.

As of 2017, MATLAB has over 2 million users across industry and academia. MATLAB users come from various backgroundsof engineering, science, andeconomics.

M CODE (MACHINECODE)

M code or **M-Code** may refer to

- Machine Code
- MATLAB programminglanguage
- Military GPS signal (or GPS_signals#Military_.28M-code.29),or
- halfof the **G & M**-Code programming language used in the CNC Machining Industry.

Every processor or processor family has its own machine code instruction set. Instructions are patterns of bits that by physical design correspond to different commands to the machine. Thus, the instruction set is specific to a class of processors using (mostly) the same architecture. Successor or derivative processor designs often include all the instructions of a predecessor and may add additional instructions. Occasionally, a successor design will discontinue or alter the meaning of some instruction code (typically because it is needed for new purposes), affecting code compatibility to some extent; even nearly completely compatible processors may show slightly different behavior for some instructions, but this is rarely a problem. Systems may also differ in other details, such as memory arrangement, operating systems, or peripheral devices. Because a program normally relies on such factors, different systems will typically not run the same machine code, even when the same type of processor isused.

A machine code instruction set may have all instructions of the same length, or it may have variable-length instructions. How the patterns are organized varies strongly with the particular architecture and often also with the type of instruction. Most instructions have one or more opcode fields which specifies the basic instruction type (such as arithmetic, logical, jump, etc.) and the actual operation (such as add or compare) and other fields that may give the type of the operand(s), the addressing mode(s), the addressing offset(s) or index, or the actual value itself (such constant operands contained in an instruction are called *immediates*).

Not all machines or individual instructions have explicit operands. An accumulator machine has a combined left operand and result in an implicit accumulator for most arithmetic instructions. Other architectures (such as 8086 and the x86-family) have accumulator versions of common instructions, with the accumulator regarded as one of the general registers by longer instructions. A stack machine has most or all of its operands on an implicit stack. Special purpose instructions also often lack explicit operands (CPUID in the x86 architecture writes values into four implicit destination

registers, for instance). This distinction between explicit and implicit operands is important in machine code generators, especially in the register allocation and live range tracking parts. A good code optimizer can track implicit as well as explicit operands which may allow more frequent constant propagation, constant folding of registers (a register assigned the result of a constant expression freed up by replacing it by that constant) and other codeenhancements.

Programs

A computer program is a sequence of instructions that are executed by a CPU. While simple processors execute instructions one after another, superscalar processors are capable of executing several instructions at once.

Program flow may be influenced by special 'jump' instructions that transfer execution to an instruction other than the numerically following one. Conditional jumps are taken (execution continues at another address) or not (execution continues at the next instruction) depending on some condition.

Assemblylanguages

Main article: Assemblylanguage

A much more readable renditionof machine language, called assembly language, uses mnemonic codes to refer to machine code instructions, rather than using the instructions' numeric values directly. For example, on the Zilog Z80 processor, the machine code 00000101 , which causes the CPU to decrement the B processor register, would be represented in assembly language as DEC B.

Example

The MIPS architecture provides a specific example for a machine code whose instructions are always 32 bits long. The general type of instruction is given by the *op* (operation) field, the highest 6 bits. J-type (jump) and I-type (immediate) instructions are fully specified by *op*. R-type (register) instructions include an additional field *funct*to determine the exact operation. The fields used in these typesare:

```
  6     5     5     5     5     6bits
[ op | rs | rt | rd |shamt| funct]R-type
[ op |  rs |  rt | address/immediate]  I-type
[ op |        targetaddress         ]J-type
```

rs,*rt*, and*rd*indicate register operands;*shamt*gives a shift amount; and

the *address* or *immediate* fields contain an operanddirectly.

For example, adding the registers 1 and 2 and placing the result in register 6 is encoded:

```
[ op | rs | rt | rd |shamt| funct]
   0    1    2    6    0    32     decimal
000000 00001 00010 00110 00000 100000binary
```

Load a value into register 8, taken from the memory cell 68 cells after the location listed in register 3:

```
[ op | rs | rt | address/immediate]
  35    3    8         68          decimal
100011 00011 01000 00000 00001 000100binary
```

Jumping to the address 1024:

```
[ op |        targetaddress        ]
  2               1024              decimal
000010 00000 00000 00000 10000 000000binary
```

Relationship tomicrocode

In some computer architectures, the machine code is implemented by an even more fundamental underlying layer called microcode, providing a common machine language interface across a line or family of different models of computer with widely different underlying dataflow. This is done to facilitate porting of machine language programs between different models. An example of this use is the IBM System/360 family of computers and their successors. With dataflow path widths of 8 bits to 64 bits and beyond, they nevertheless present a common architecture at the machine language level across the entireline.

Using microcode to implement an emulator enables the computer to present the architecture of an entirely different computer. The System/360 line used this to allow porting programs from earlier IBM machinesto the new family of computers, e.g. an IBM 1401/1440/1460 emulator on the IBM S/360 model40.

Relationship tobytecode

Machine code is generally different than byte code (also known as p-code), which is either executed by an interpreter or itself compiled into machine code for faster (direct) execution. An exception is when a processor is designed to use a particular bytecode directly as its machine code, such as is the case with Javaprocessors.

Machine code and assembly code are sometimes called *native code* when referring to platform-dependent parts of language features or libraries.

Storing inmemory

The Harvard architecture is computer architecture with physically separate storage and signal pathways for the code (instructions) and data. Today, most processors implement such separate signal pathways for performance reasons but actually implement a Modified Harvard architecture,[*citation needed*] so they can support tasks like loading an executable program from disk storage as data and then executing it. Harvard architecture is contrasted to the Von Neumann architecture, where data and code are stored in the same memory which is read by the processor allowing the computer to execute commands.

From the point of view of a process, the *code space* is the part of its address space where the code in execution is stored. In multitasking systems this comprises the program's code segment and usually shared libraries. In multi-threading environment, different threads of one process share code space along with data space, which reduces the overhead of context switching considerably as compared to process switching.

Readability byhumans

It has been said that machine code is so unreadable that the United States Copyright Office cannot identify whether a particular encoded program is an original work of authorship; however, the US Copyright Office *does* allow for copyright registration of computer programs and a program's machine code can sometimes be decompiled in order to make its functioning more easily understandable to humans.

Cognitive science professor Douglas Hofstadter has compared machine code to genetic code, saying that "Looking at a program written in machine language is vaguely comparable to looking at a DNA molecule atom by atom.

Software hierarchy

The lowest level description of a computer program is just the sequence of numbers which encode the basic CPU operations. This level is called **machine code**. Machine code is specific to a given CPU manufacturer and often specific to a given model type

(for example the Pentium CPU has some codes not used by earlier 8086 CPUs). Machine code is very difficult for a human to read or write,so the lowest level of programming done by humans is in a language in which each basic operation is given a mnemonic code called **assembly language**. Humans can read and write using assembly language which can be converted into machine code using an **assembler**. Assembly language, like machine code is often specific to a particular CPU manufacturer ormodel.

The development of **high-level languages** meant that humans could program using a formalism that was closer to their conceptual models of the data being manipulated: characters, real numbers, lists, tables or database records. Such languages are easier for humans to learn and to use, and furthermore they tend to be available across different computers; with each manufacturer supplying a conversion program between the high-level language and the assembly language for their CPU. Examples of high-level languages are Fortran, Pascal, Basic, C, C++, Java andMATLAB.

Modern computer systems need to deal with complex tasks involving multiple programs interacting simultaneously, and the sharing of access to files on disks, to network resources and displays. To cope with these demands, manufacturers supply **operating systems** (e.g. Windows, Linux), which are themselves programs which help the user operate the computer and run other **application** programs. Often individual application programs need to work together to achieve an objective: for example a word processing application might call on a drawing package or on a spreadsheet program to do some specific processing within a document. This idea of combining programs is called **scripting**, where the specifications for which programs are to be executed and how they should interact is specified in a **script**.

Possible Questions

Part-A (1 mark-Online Exam)

Part – B (2 marks)

1. Define machine language
2. Define assembly language
3. List the components of computer

Part – C (6 marks)

1. Discuss about components of computer.
2. Explain in detail working with numbers and softwarehierarchy.
3. Discuss about input, output and storagedevices.
4. Elaborate about high-level language and machinelanguage.
5. Explain in detail about programming in machine level, assembly level and high-levellanguage.
6. Describe about machine code and softwarehierarchy.

Question	Option 1	Option 2	Option 3	Option 4	Answer
Finite sequence of instructions is known as	Program	Flow Chart	Algorithm	Software	Algorithm
An algorithm expressed in a programming language is called as	Expression	Computer program	Instruction	Data	Computer program
An algorithm can be expressed in a graphical form known as	Program	Translator	Flow chart	Bar chart	Flow chart
Data in computers are represented using	Binary Digits	Bits	Bytes	Units	Binary Digits
A set of computer programs and related data that provide the	Hardware	Software	Maleware	Shareware	Software
Software designed to operate the computer hardware and to provide a	System software	Application software	Operating Systems	Utility programs	System software
A collection of programs that form a bridge between user and the hardware	Operating systems	Translators	Software	Program	Operating systems
Program that translates a set of code written in programming language into	Translator	Compiler	Loder	Linker	Translator
Which will translates assembly language programs into machine code	Assembler	Compiler	All the Above	None of the above	Assembler
Compiler translates _____	Object code into	Object code into	Source code to	High level language	High level language
Which type of translator reads only one line at a time and converts it to	Assembler	Compiler	Interpreter	Linker	Interpreter
Software designed to help the user to perform specific tasks is	System software	Application software	Operating Systems	Utility programs	Application software
The programs written in machine language are	Machine independen	Machine dependent	Machine interconne	Machine interface	Machine dependent
Assembly language consists of	Mnemonic s	Source code	Object code	Byte Code	Mnemonic s
Assemblers perform a _____	One to One translation	One to Many	Many to Many	Many to One	One to One translation
Tool that produces input for compilers	Loader	Interpreter	Preprocess or	Wordproce ssor	Preprocess or
Computer program that links and merges various object files together in	Dynamic Linker	Linker	Loader	Static Linker	Linker
High Level Languages are belongs to which generation	First generation	Second generation	Fourth generation	Third generation	Third generation
Functional languages treat procedures like	Mathemati cal	English like words	Symbols	Charaters	Mathemati cal
An example of Functional language	COBOL	FORTRAN	PASCAL	C++	PASCAL
The subprogram may be called by using a Simple	Instrution	Task	Symbols	Classes	Instruction
In functional language, each module has its own set of	Variable	External Variable	Dynamic Variable	Local Variable	Local Variable
The role of a compiler is to translate source program statements to	Program code	Object code	Octal code	Decimal code	Object code
An interpreter reads the source code of a program	one line at a time	two line at a time	complete program in	n number of code	one line at a time
Computer is a digital device which have only two state	COMPILE &	ON & OFF	START & STOP	TRUE & FALSE	ON & OFF

Machine language is also called as	Original Language	Starting Language	Initial Lanugage	Native Language	Native Language
Instruction in machine language consists of two parts	Character & String	Array & Structure	Operation & Operand	Variable & Symbol	Operation & Operand
Program change the flow of execution by using special instruction	JUMP	MOVE	TRANSFERE	ALTER	JUMP
Which is the only lanaguage that computers can directly executed	Assembly Language	Machine Language	High Level Lanaguage	Very High Level	Machine Language
Assembly language was developed by	HP	Microsoft	Apple	IBM	IBM
Assembly language allows the program to interact directly with	Hardware	Software	Maleware	Shareware	Hardware
The program written in high level language can be used in	Files	Platform	Software	Hardware	Platform
In high level language there is no need to aware of	Networks	Architecture	Hardware	Peripherals	Architecture
MICR stands for	Magnetic Ink	Magnetic Ink Code	Magnetic Ink Cases	Magnetic Ink	Magnetic Ink
Fourth generation languages need large memory capacity and	Disk Space	Resource	Connection	Process speed	Disk Space
Using compiler for translation process execution time get	decreased	increased	doubles	Tripled	increased
Which generation of the language have less number of syntax	Procedural Lanaguage	Query Lanaguage	Functional Language	Structural Language	Query Lanaguage
CD-ROM stands for	Compactable Read	Compact Data Read	Compactable Disk	Compact Disk Read	Compact Disk Read
ALU is	Arithmetic Logic Unit	Array Logic Unit	Application Logic	Arithmetic Local Unit	Arithmetic Logic Unit
VGA is	Video Graphics	Visual Graphics	Volatile Graphics	Video Graphics	Video Graphics
Which of the following is first generation of computer	EDSAC	IBM-1401	CDC-1604	ICL-2900	EDSAC
Chief component of first generation computer was	Transistors	Vacuum Tubes and	Integrated Circuits	Circuits	Vacuum Tubes and
The computer size was very large in	First Generation	Second Generation	Third Generation	Fourth Generation	First Generation
Microprocessors as switching devices are for which generation computers	First Generation	Second Generation	Third Generation	Fourth Generation	Fourth Generation
In latest generation computers, the instructions are executed	Parallel only	Sequentially only	Both sequentiall	All	All
In which type of computer, data are represented as discrete signals.	Analog computer	Digital computer	Analog computer	Hybrid Computer	Digital computer
Daisy wheel, Drum, chain etc are the	Flow chart	Mouse	Key board	Printers	Printers
Trackball is a _____	Input device	Output device	Programming	Software	Input device
The common name for the crime of stealing passwords is	Jacking	Identity theft	Spoofing	Hacking	Spoofing

Unit II

Syllabus

Programming Environment: MATLAB Windows, A First Program, Expressions, Constants, Variables and assignment statement, Arrays.

MATLAB WINDOWS

It is assumed that the software is installed on the computer, and that the user can start the program. Once the program starts, the MATLAB desktop window opens (Figure 1-1). The window contains four smaller windows: the Command Window, the Current Folder Window, the Workspace Window, and the Command History Window. This is the default view that shows four of the various windows of MATLAB. A list of several windows and their purpose is given inTable 1-1. The Start button on the lower left side can be used to access MATLAB tools and features. Four of the windows—the Command Window, the Figure Window, the Editor Window, and the Help Window—are used extensively throughout the book and are briefly described on the followingpages

Command Window: The Command Window is MATLAB's main window and opens when MATLAB is started. It is convenient to have the Command Window as the only visible window, and this can be done by either closing all the other windows (click on the x at the top right-hand side of the window you want to close) or by first selecting the Desktop Layout in the Desktop menu, and then 6 Chapter 1: Starting with MATLAB selecting Command Window Only from the submenu that opens

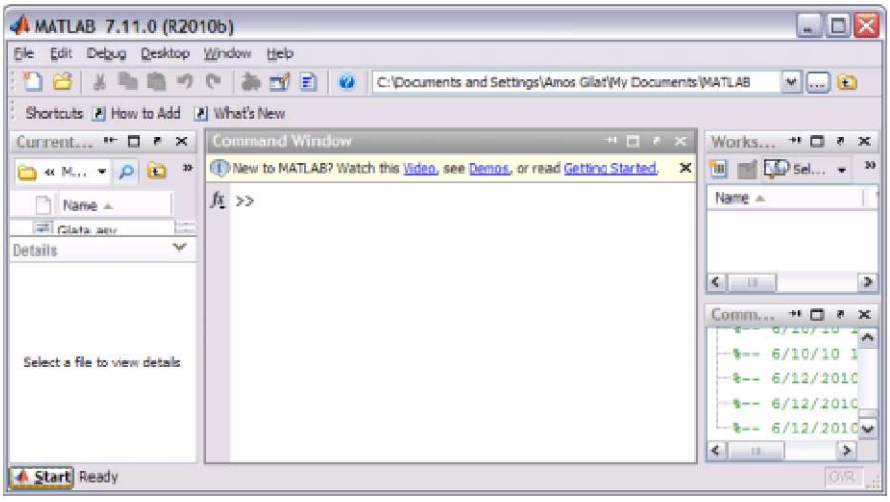


Figure 1-1: The default view of MATLAB desktop.

Table 1-1: MATLAB windows

Window	Purpose
Command Window	Main window, enters variables, runs programs.
Figure Window	Contains output from graphic commands.
Editor Window	Creates and debugs script and function files.
Help Window	Provides help information.
Command History Window	Logs commands entered in the Command Window.
Workspace Window	Provides information about the variables that are used.
Current Folder Window	Shows the files in the current folder.

Figure Window: The Figure Window opens automatically when graphics commands are executed, and contains graphs created by these commands. An example of a Figure Window is shown in Figure 1-2.

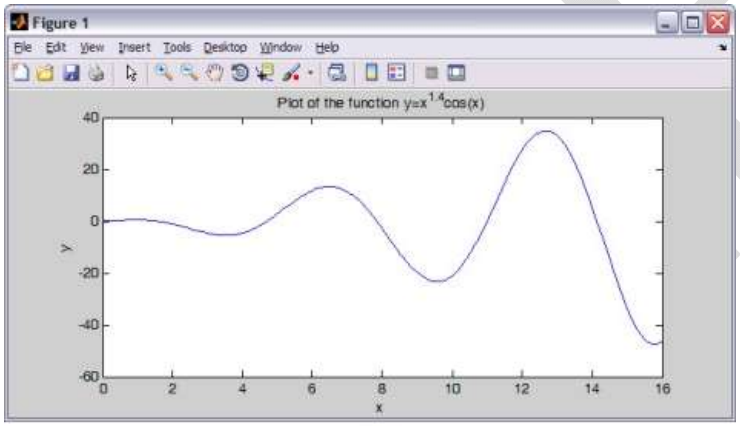


Figure 1-2: Example of a Figure Window.

Editor Window: The Editor Window is used for writing and editing programs. This window is opened from the File menu. An example of an Editor Window is shown in Figure 1-3.

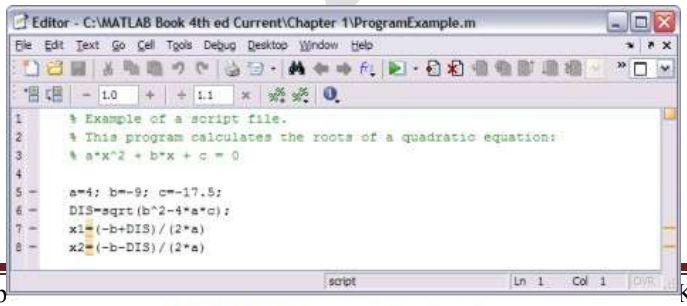


Figure 1-3: Example of an Editor Window.

Help Window: The Help Window contains help information. This window can be opened from the Help menu in the toolbar of any MATLAB window. The Help Window is interactive and can be used to obtain information on any feature of MATLAB. Figure 1-4 shows an open Help Window.

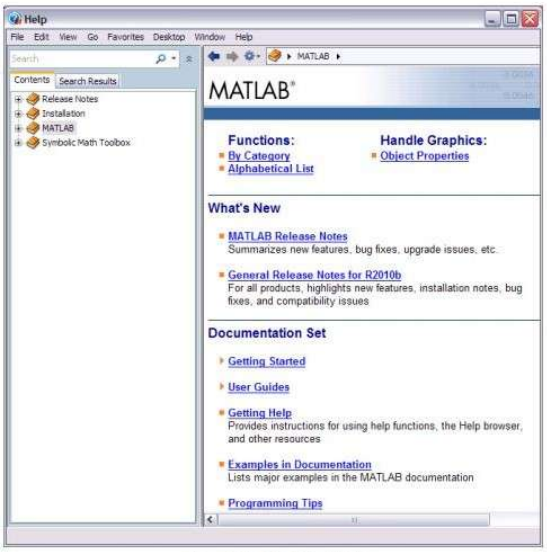


Figure 1-4: The Help Window.

Working In The Command Window The Command Window is MATLAB's main window and can be used for executing commands, opening other windows, running programs written by the user, and managing the software. An example of the Command Window, with several simple commands that will be explained later in this chapter, is shown in Figure1-5.

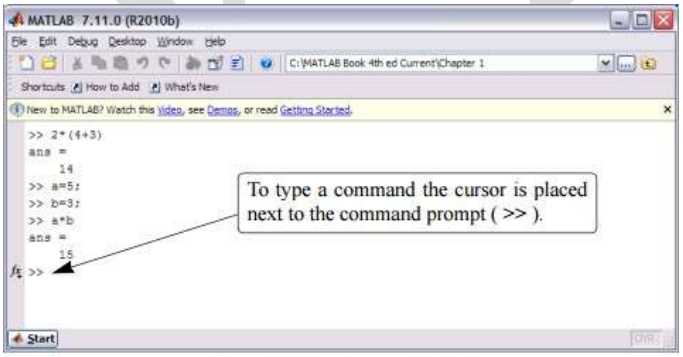


Figure 1-5: The Command Window.

A FIRST PROGRAM

Matlab stores most of its numerical results as matrices. Unlike some languages (C, C++, C#), it dynamically allocates memory to store variables. Therefore, it is not necessary to declare variables before using them. Let's begin by simply adding two numbers. Click in the Command Window. You will see a flashing "|" symbols next to the ">>" symbol. Enter the following commands

1. Type in "x = 3" then hit"enter"
2. Type in "y = 2;" then hit "enter" (note the semicolonhere!)
3. Type "z = x + y" then hit"enter"

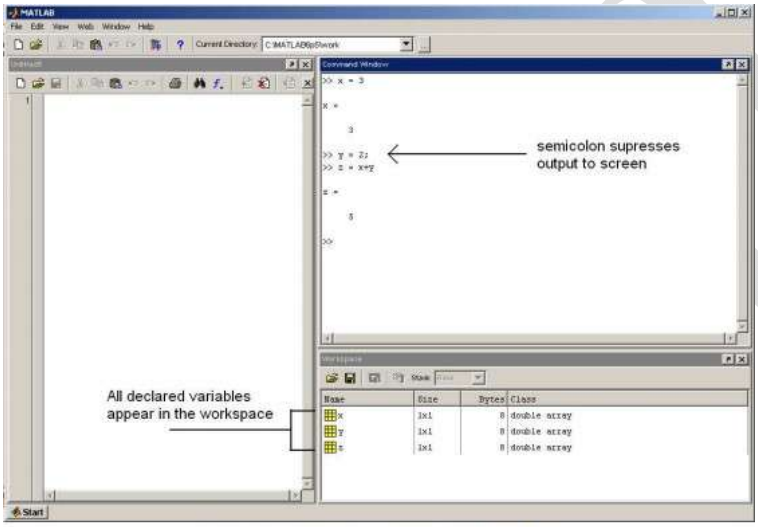


Figure 4: Entering in scalar values into **Matlab**

All declared variables appear in the workspace. Recall that these values are stored as matrices. The "size" column tells us the dimension of the matrix. As expected, all these variables are 1x1 scalar values. To double check on value stored in this matrix, simply double click any of the variables in the Workspace.

Example program

The command

```
disp(argument);
```

displays the value of the argument. This can be a number, a string in single quotes, or an expression. For simple numbers, the arithmetic operators are: **+, -, *,** / and**^.**Try

```
disp(2*3+1);
```

or

```
disp('HelloWorld!');
```

Try these programs out first on the command line; then practise using the editor to enter the commands, saving them to a file, loading the file and running the program from inside the editor.

Expressions

Variables

Like mostother programming languages, the MATLAB language provides mathematical *expressions*, but unlike most programming languages, these expressions involve entirematrices.

MATLAB does not require any type declarations or dimension statements. When MATLAB encounters a new variable name, it automatically creates the variable and allocates the appropriate amount of storage. If the variable already exists, MATLAB changes its contents and, if necessary, allocates new storage. Forexample,

```
num_students = 25
```

creates a 1-by-1 matrix named num_students and stores the value 25 in its single element. To view the matrix assigned to any variable, simply enter the variable name.

Variable names consist of a letter, followed by any number of letters, digits, or underscores. MATLAB is case sensitive; it distinguishes between uppercase and lowercase letters. A and a are *not* the samevariable.

Although variable names can be of any length, MATLAB uses only the first N characters of the name, (where N is the number returned by the function namelengthmax), and ignores the rest. Hence, it is important to make each variable name unique in the first N characters to enable MATLAB to distinguish variables.

```
N = namelengthmax
N =     63
```

Numbers

MATLAB uses conventional decimal notation, with an optional decimal point and leading plus or minus sign, for numbers. Scientific notation uses the letter e to specify a power-of-ten scale factor. Imaginary numbers use either i or j as a suffix. Some examples of legal numbersare

```
3           -99          0.0001
9.6397238   1.60210e-20  6.02252e23
1i          -3.14159j    3e5i
```

MATLAB stores all numbers internally using the *long* format specified by the IEEE® floating-

point standard. Floating-point numbers have a finite *precision* of roughly 16 significant decimal digits and a finite *range* of roughly 10^{-308} to 10^{+308}.

Numbers represented in the double format have a maximum precision of 52 bits. Any double requiring more bits than 52 loses some precision. For example, the following code shows two unequal values to be equal because they are both truncated:

```
x =36028797018963968;
y =36028797018963972;
x == y
ans =
     1
```

Integers have available precisions of 8-bit, 16-bit, 32-bit, and 64-bit. Storing the same numbers as 64-bit integers preserves precision:

```
x = uint64(36028797018963968);
y = uint64(36028797018963972);
x == y
ans = 0
```

Matrix Operators

Expressions use familiar arithmetic operators and precedence rules.

+	Addition
-	Subtraction
*	Multiplication
/	Division
\	Left division
^	Power
'	Complex conjugate transpose
( )	Specify evaluation order

Array Operators

When they are taken away from the world of linear algebra, matrices become two-dimensional numeric arrays. Arithmetic operations on arrays are done element by element. This means that addition and subtraction are the same for arrays and matrices, but that multiplicative operations are different. MATLAB uses a dot, or decimal point, as part of the notation for multiplicative

array operations.

The list of operators includes

+	Addition
-	Subtraction
.*	Element-by-element multiplication
./	Element-by-element division
.\	Element-by-element left division
.^	Element-by-element power
.'	Unconjugated array transpose

If the Dürer magic square is multiplied by itself with array multiplication

```
A.*A
```

the result is an array containing the squares of the integers from 1 to 16, in an unusual order:

```
ans =
  256    9    4 169
   25  100  121   64
   81   36   49144
   16  225  196    1
```

Examples of Expressions

You have already seen several examples of MATLAB expressions. Here are a few more examples, and the resulting values:

```
rho = (1+sqrt(5))/2
rho =
   1.6180


a = abs(3+4i)
a=
    5
```

```
z =sqrt(besselk(4/3,rho-i))
z=
  0.3730+ 0.3214i

huge = exp(log(realmax))
huge =
  1.7977e+308

toobig = pi*huge
toobig =
 Inf
```

Variables and assignment

Variables are named locations in memory where numbers, strings and other elements of data maybestoredwhiletheprogramisworking.Variablenamesarecombinationsoflettersand digits, but must start with a latter. MATLAB does not require you to declare the names of variables in advance of their use. This is actually a common cause of error, since it allows you to refer accidentally to variables that don't exist. To assign a variable a value, use the **assignment statement**. This takes the form

variable=expression;

forexample

```
a=6;
```

or

```
name='Mark';
```

To display the contents of a variable, use

```
disp(variable);
```

Please note that−

- Once a variable is entered into the system, you can refer to itlater.

- Variables must have values before they areused.
- When an expression returns a result that is not assigned to any variable, the system assigns it to a variable named ans, which can be usedlater.

For example,

```
sqrt(78)
```

MATLAB will execute the above statement and return the following result −

```
ans = 8.8318
```

You can use this variable **ans** −

```
sqrt(78);
9876/ans
```

MATLAB will execute the above statement and return the following result −

```
ans = 1118.2
```

Let's look at another example −

```
x = 7 * 8;
y = x * 7.89
```

MATLAB will execute the above statement and return the following result −

```
y = 441.84
```

Multiple Assignments

You can have multiple assignments on the same line. For example,

```
a = 2; b = 7; c = a * b
```

MATLAB will execute the above statement and return the following result −

```
c = 14
```

Arrays

MATLAB is particularly powerful in the way it deals with tables of data, called arrays. An array is simply a variable that can contain a number of values arranged in tabular form. Arrays may be one dimensional (like a list), two dimensional (like a table), or have more dimensions. To set the value of one element of a one dimensional array, use thenotation

variable(*index*)=*expression*;

forexample

```
table(1)=3;
table(2)=6;
```

Note that indexes must be expressions evaluating to positive integers. The smallest index is 1. To access one element from a one dimensional array, use the notation

variable(*index*)

forexample

```
a=table(2);
disp(table(2));
```

For two dimensional arrays, use

variable(*index*,*index*)=*expression*;

to set the valueand

variable(*index*,*index*)

to retrieve its value. You can store strings in tables, but each string occupies a row, and all rows must be the same length (think of a two-dimensional array ofcharacters).

You can assign a whole array in one operation using a notation involving square brackets: for example:

array= [v11 v12 v13; v21v22v23];

where v11 is the value in row 1 col 1; v21 is the value in row 2 col 1; etc. The ';' marks the end of arow.

You can generate arrays containing sequences very easily with the ':' operator. The expression

start:*stop*

generates a sequence of integers from start to stop. The expression

start:*increment*:*stop*

generates a sequence from start to stop with the specfied increment. Try

```
disp(1:10);
disp(1:2:10);
```

You can also select sub-parts of the array with the ':' operator. For example,

x(3:5)

represents the array consisting of the third through fifth elements of x.Also

y(2:2:100)

represents the array containing the even number elements of y below index 100.

You can also add subtract, multiply and divide arrays of data using the operators we've mentioned previously. However MATLAB makes a difference between operations that work on a cell-by-cell basis (so-called "dot" operations) as opposed to operations that work on the arrays as a whole. For example, if you want to multiply two arrays of equal size to give a third array in which each cell contains the product of the corresponding cells in the input, then you need to use the "dot-multiply" operator .* forexample

```
C =A.*B;
```

Finally you can transpose rows and columns of a matrix with the ' operator, for example

```
disp(A')
```

Possible Questions

Part-A (1 mark-Online Exam)

Part – B (2 marks)

1. List the MATLABwindows.
2. List the matrix operators, Arrayoperators.
3. Write a program for matrixmultiplication.

Part – C (6 marks)

1. Explain in detail about MATLABwindows.
2. Discuss about array withexample.
3. Explain about variables andconstants.
4. Write in detail about array functions with example andoutput.

Question	Option 1	Option 2	Option 3	Option 4	Answer
MATLAB stands for _____	Maths Labor	Matrix Labor	Mathematica	Maths Lab	Matrix Laboratory
MATLAB was developed by _____	MathsWorks	Intel	Microsoft	IBM	MathsWorks
In MATLAB the matrix is defined as an _____	vector	scalar	array	integer	array
_____ acts as an outstanding tool for visulaizing technical data	C	C++	Java	MATLAB	MATLAB
The fundamental unit of data in any MATLAB program is the _____	array	vector	scalar	none	array
In command window the _____ are entered	data	values	commands	fields	commands
_____ window displays plots and graphs	command	Edit	Figure	Command hi	Figure
_____ window permits a user to create and modify MATLAB programs	command	Editor	Figure	Command hi	Editor
MATLAB programs are saved with the extensions _____	.m	.mm	.mf	.ml	.m
The _____ window displays a list of commands that a user has entered in the command window	edit	figure	debug	command his	command history
When a window is _____ it appears as a pane within the MATLAB desktop	docked	undocked	removed	deleted	docked
A _____ is a collection of all the variables and arrays that can be used by MATLAB when a particular command is executed	editor	workspace	desktop	none	workspace
_____ command is used to list all the variables and arrays in the current workspace	string	whos	whose	where	whos
A variable can be deleted from the workspace with the _____ command	delete	remove	clear	omit	clear
The _____ command will display a list of possible help topics in the command window	help	helper	lookfor	order	help
The _____ command searches the quick summary information in each function for a match	lookfor	helper	help	order	lookfor
The term _____ is used to describe an array with only one dimension	array	vector	matrix	scale	vector
The term _____ is used to describe an array with two or more dimensions	array	vector	matrix	scale	matrix
A variable o ftype _____ is automatically created whenever a numerical value is assigned to a variable name	double	long	int	short	double
MATLAB is a _____ typed language	strongly	weakly	stronger	thiner	weakly
The _____ operator swaps the row and columns of any array that is given	transpose	concatenates	colon	semicolon	transpose
The _____ function can be used ti create an all zero array	ones	zero	eye	randn	zero
The _____ function can be used to generate arrays containing all ones	ones	zero	eye	randn	ones

The eye function can be used to generate arrays coantining _____ matrices	square	null	identity	none	identity
MATLAB always allocated array elements in _____ major order	row	column	row & columr	none	column
The _____ functions returns the highest value taken on by that subscript	ones	zero	end	repalce	end
pi is an example of _____	operators	functions	plotting	special value	special values
NaN stands for _____	Number and	number	not a number	not and numl	not a number
The default format for displaying the output can be changed by using _____ command	path	format	special	null	format
The _____ function accepts an array arugment and displayus the value of athe array in the command window	disp	format	special	fprintf	disp
the _____ function displays one or more values together with realted text	disp	format	fprintf	special	fprintf
The _____ command loads data from a disk file into the current MATLAB WORKSPACE	save	update	load	open	load
_____ are operations performed between arrays on an element by element basis	matrix operat	array operati	vector operat	arthimetic op	array operations
In _____ the number of rows and columns in both arrays must be the same	matrix operat	array operati	vector operat	arthimetic op	array operations
The MATLAB functions can return _____ results to the calling program	more than on	exactly one	only two	none	more than one
The _____ function rounds x to the nearest integer twords zero	ceil(x)	fix(x)	floor(x)	round(x)	fix(x)
The _____ function rounds x to the nearest integer	ceil(x)	fix(x)	floor(x)	round(x)	round(x)
The _____ command can be used to save a plot as a graphical image by specifying appropriate options and a filename	plot	print	draw	multiple	print
If the result of the MATLAB expression is not assigned to any variable, then itr is stored in default variable _____	result	ans	answer	output	ans
The _____ command clears the screen	clc	clr	cls	cle	clc
The _____ command clears the figure window	clc	clf	cls	cle	clf
the _____ returns tangent of an angle given in degrees	tang	tand	tan	tan2	tand
If a step size is not specified , + ___is taken as default value of the step size	2	1	3	4	1
The _____ gives the number of elements in a row/column vector	len(x)	size(x)	length(x)	none	length(x)
The _____ command returns the number of elements of the matrix in each dimentsions	len(x)	length(x)	size(x)	none	size(x)
The _____ function concatenates a list of arrays along a specified dimension	join	cat	rand	joined	cat
the _____ function gives the minimum value in row/column vector	min	least	max	minum	min

The _____ function gives the maximum value in row/column vector	min	least	max	minum	max
The _____ gives the transpose of x	x'	x''	x'''	x	x'

Unit-III

Syllabus

Graph Plots: Basic plotting, Built in functions, Generating waveforms, Sound replay, load and save. Procedures and Functions: Arguments and return values, M-files, Formatted console input-output , String handling,

Basic Plotting

To create XY graphs, it is easiest to form your data into two row vectors, one for the x co-ordinates, and one for the y co-ordinates. The command

```
plot(x,y)
```

will then create a figure with points at each y value for each matching x value. You can control the style of any line drawn through the points by a third string argument to the plot command:

```
plot(x,y,style);
```

where style is made up from characters as follows:

- Color strings are 'c', 'm', 'y', 'r', 'g', 'b', 'w', and 'k'. These correspond to cyan, magenta, yellow, red, green, blue, white, and black.
- Linestyle strings are '-' for solid, '--' for dashed, ':' for dotted, '-.' for dash-dot, and none for no line.
- The marker types are '+', 'o', '*', and 'x' and the filled marker types 's' for square, 'd' for diamond, '^' for up triangle, 'v' for down triangle, '>' for right triangle, '<' for left triangle, 'p' for pentagram, 'h' for hexagram, and none for no marker.

For example:

```
x = [ 1 2 3 4 ];
y = [ 10 15 20 25 ];
plot(x,y,'g-*');
```

You can plot multiple lines by repeating the arguments:

```
plot(x1,y1,x2,y2,…);
```

or

```
plot(x1,y1,style1,x2,y2,style2,…);
```

You can give the graph a title with the

```
title(label);
```

command, where label is a character string. Likewise you can add labels to the X and Y axes with

```
xlabel(label);
```

and

```
ylabel(label);
```

You can add a legend with

```
legend(label1,label2,label3,…);
```

Built in functions

Generation

zeros()	matrix of specified size filled with zeros
ones()	matrix of specified size filled with ones
rand()	generate pseudo random number(s) between 0 and 1

Arithmetic

rem()	remainder after integer division
abs()	absolute value (also character -> number)
fix()	truncate a value to its integer part (towards zero)
round()	round a value to nearest integer.
sqrt()	square root
sin()	sine (angle in radians)
cos()	cosine (angle in radians)
exp()	exponential
log()	natural logarithm
log10()	logarithm base 10

Status

length()	length of a vector (longest dimension of matrix)
size()	size of a matrix [nrows, ncols]

Miscellaneous

sum()	sum the elements of a vector
mean()	find mean of elements of a vector
sort()	sort the elements of a vector in increasing size
clock()	returns date and time as a vector [year month day hour minute seconds]
date()	returns date as a string dd-mmm-yyyy

Generating waveforms

Waveforms are just long vectors with one number per amplitude sample. Usually they are best kept scaled so that each amplitude is between –1 and 1. To generate a sinewave, first generate a time sequence t representing the times of each sampling instant; for example:

```
t = 0:0.0001:2;
```

would generate a two second sequence with a sampling interval of 0.1ms (i.e. 10,000Hz). You can then generate a sinewave at frequency F with the expression

```
y = sin(2*pi*F*t);
```

You can create a pulse by creating a vector of zeros and setting a single element to one.

A pulse train has a series of elements set to one. If these occurred every 100 elements, you might use the expression

```
y(1:100:10000)=1;
```

To create a simple sawtooth, you can use the remainder function, for example

```
y = rem(1:10000,100)/100;
```

To create a noise waveform, you can use the 'rand(*nrows*,*ncols*)' function, for example

```
y = rand(1,10000);
```

Sound Replay, Load and Save

To replay a waveform, you can use

```
sound(wave,samplerate);
```

To ensure that the waveform is scaled to the range –1 .. +1 before replay, use

```
soundsc(wave,samplerate);
```

instead.

To save a waveform to a file, use

```
save filename variable;
```

To load a waveform from a file, use

```
load filename variable;
```

To save a waveform in a Windows compatible audio file format, use

```
wavwrite(waveform,samplerate,filename);
```

To load a Windows compatible audio file, use

```
[waveform,samplerate,nbits]=wavread(filename);
```

Procedure and Functions: Arguments and return values

Functions

You can define your own functions to complement those provided by MATLAB. Functions are the building blocks of your own programs. You should always try and divide your programming task into separate functions, then design, code and test each one independently. It is common to design from the top down, but build from the bottom up.

It is good practice to store each function in its own source file, with the name of the source file matching the function. Thus a function called "myfunc" will be stored in the file "myfunc.m". This way, both you and MATLAB can easily find the source file for a function given its name. The first line of a function source file should then be the function definition line, which has the format:

```
function outargs=funcname(inargs);
```

The function name can be a mixture of letters and digits but must start with a letter. It is a good idea to avoid names that MATLAB is already using. The *inargs*parameter is a list of variable names separated by commas. These are the dummy names you will use in the code for the function to 'stand for' the actual arguments passed to the function when it is executed. Likewise the *outargs* parameter is a list of variable names separated by

commas which stand for the values returned by the function to the calling program. Note that a function can take zero or more input arguments and return zero or more values. Here are some example function definitions:

```
function y=square(x);
```

```
function av=average(x1,x2,x3,x4,x5);
function printvalue(A);
function B=readvalue();
function [mean,sttdev]=analyse(tab);
```

Following the function line you should write a one line comment that summarises what the function does. For example:

```
% square(x) returns the square of the argument x
```

This line is printed out if the user types

```
lookfor funcname;
```

in the command window. All the comment lines between the function definition and the first executable statement are printed out when the user types

```
help funcname;
```

in the command window. Use this facility to provide some help information to the users of your function.

The body of your function will normally perform some computation based on the input arguments and end by assigning some values to the output arguments. When the function is called from another program, whatever values are supplied to the function are copied into the dummy input arguments, then the function is executed, then the values of the output dummy arguments are inserted into the calculation in the calling program. It is good practice to end each function with the returnstatement to remind you that execution returns to the calling program at this point.

```
function y=cube(x)
% cube(x) returns the cube of x
y = x * x * x;
return;
```

```
a=10;
b=cube(a);
disp(b);            % \
disp(cube(a));      % All display 1000
disp(cube(10));     % /
```

It is good practice to pass all the information you need for a function through the list of input arguments and to receive all the processed results through the output arguments. Although this requires a lot of copying, MATLAB does this quite efficiently. Sometimes however, you may have a number of functions that all require access to the same table of data, and you don't want to keep copying the table into the function and then copying the changes back into your program. Imagine if the table had a million elements! Under these circumstances you can declare variables as 'global'. This means that they can be

accessed both inside your program and inside a function without having to pass the variable as a function argument. Here is an example:

```
function initialisetable(num)
% initialise global variable TAB to all the same value
global TAB;
TAB=num*ones(size(TAB));

% main program
global TAB;
TAB=zeros(1,100);
initialisetable(5);
```

You can also write functions which take a variable number of arguments. In fact MATLAB allows any function to be called with fewer arguments than the definition, so it is a good idea to always check the number of arguments supplied. The built in variable 'nargin' contains the number of input arguments actually supplied, and 'nargout' contains the number of output arguments. You can use the built in function 'error()' to report an error if the number of arguments is incorrect. For example:

```
function m=average(x,y)
if (nargin!=2)
error('two arguments needed in average()');
end
```

M-files

MATLAB allows writing two kinds of program files −

- **Scripts** − script files are program files with **.m extension**. In these files, you write series of commands, which you want to execute together. Scripts do not accept inputs and do not return any outputs. They operate on data in the workspace.

- **Functions** − functions files are also program files with **.m extension**. Functions can accept inputs and return outputs. Internal variables are local to the function.

You can use the MATLAB editor or any other text editor to create your **.m**files. In this section, we will discuss the script files. A script file contains multiple sequential lines of MATLAB commands and function calls. You can run a script by typing its name at the command line.

Creating and Running Script File

To create scripts files, you need to use a text editor. You can open the MATLAB editor in two ways:

- Using the command prompt
- Using the IDE

If you are using the command prompt, type **edit** in the command prompt. This will open the editor. You can directly type **edit** and then the filename (with .m extension)

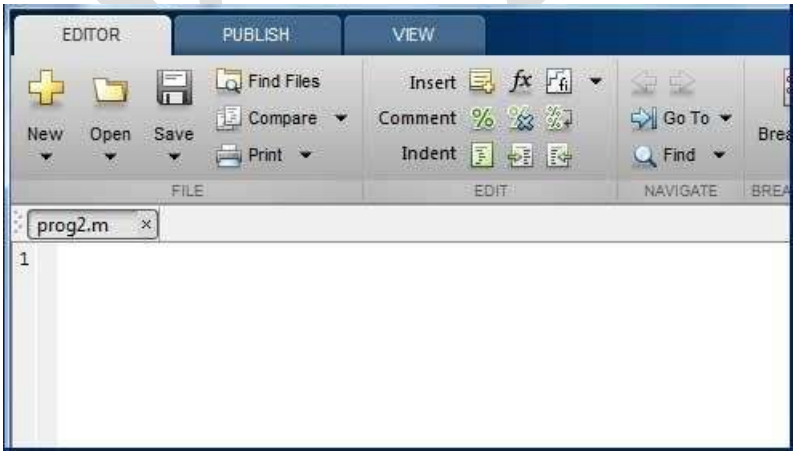
```
edit
Or
edit <filename>
```

The above command will create the file in default MATLAB directory. If you want to store all program files in a specific folder, then you will have to provide the entire path.

Let us create a folder named progs. Type the following commands at the command prompt (>>):

```
mkdir progs    % create directory progs under default directory
chdir progs    % changing the current directory to progs
edit prog1.m % creating an m file named prog1.m
```

If you are creating the file for first time, MATLAB prompts you to confirm it. Click Yes.



Alternatively, if you are using the IDE, choose NEW -> Script. This also opens the editor and creates a file named Untitled. You can name and save the file after typing the

code.

Type the following code in the editor −

```
NoOfStudents = 6000;
TeachingStaff = 150;
NonTeachingStaff = 20;
Total = NoOfStudents + TeachingStaff ...
  + NonTeachingStaff;
disp(Total);
```

After creating and saving the file, you can run it in two ways −

- Clicking the **Run** button on the editor window or
- Just typing the filename (without extension) in the command prompt: >> prog1

The command window prompt displays the result −

```
6170
```

Example

Create a script file, and type the following code −

```
a = 5; b = 7;
c = a + b
d = c + sin(b)
e = 5 * d
f = exp(-d)
```

When the above code is compiled and executed, it produces the following result −

```
c = 12
d =  12.657
e =  63.285
f =    3.1852e-06
```

Formatted console input-output

You can control the exact way in which values are printed to the screen with the 'fprintf()' function (fprintf= "file print formatted"). This function takes one argument repesenting the formatting instructions, followed by a list of values to be printed.

Embedded within the format string are 'percent commands' which control where and how the values are to be written. Here are some examples:

```
fprintf('The answer is %g seconds.\n',nsec);
fprintf('Day of the week = %s\n',dayofweek([7 12 1941]));
fprintf('Mean=%.3f ± %.4f\n',mean,stddev);
```

The command %g represents a general real number, %f means a fixed point number, %d a decimal integer, and %s a string. You can put numeric values between the '%' and the letter to control the field width and the number of digits after the decimal point. For example (□=space):

	fprintf('%5g',10)	□□□10
	fprintf('%10.4f',123.456)	□□123.4560
	fprintf('%10s', 'fred')	□□□□□□fred

You can input a value or a string from the command line with the 'input()' function. This has two forms depending on whether you want to input a number or a string:

```
yval=input('Enter a number: ');
name=input('Enter your name: ', 's');
```

Input and Output Commands

MATLAB provides the following input and output related commands −

Command	Purpose
disp	Displays contents of an array or string.
fscanf	Read formatted data from a file.
format	Controls screen-display format.
fprintf	Performs formatted writes to screen or file.
input	Displays prompts and waits for input.
;	Suppresses screen printing.

The **fscanf** and **fprintf** commands behave like C scanf and printf functions. They support the following format codes

3. String handling

Simple strings are stored as tables with one row and a number of columns: one column

per character. You can concatenate any table or strings simply by making the contents part of one table. For example:

```
str1='Hello';
str2='Mark ';
str=[str1 ' ' str2];
```

You can convert numbers to strings using the 'sprintf()' function, which operates analogously to the fprintf() function but outputs to a string rather than to the screen.

```
str=sprintf('%10.4f',123.45);
```

The 'abs()' function can be used to find the standard character codes for a string:

```
disp(abs('Mark'));
   77    97 114  107
```

The 'char()' function can be used to convert character codes back to a string:

```
disp(char([77 97 114 107]));
Mark
```

The 'eval()' function can be used to evaluate an expression stored in a string. This allows you to execute expressions typed in by the user:

```
expr=input('Enter an expression (e.g. "2+3*4") : ', 's');
disp(eval(expr));
```

Creating a character string is quite simple in MATLAB. In fact, we have used it many times. For example, you type the following in the command prompt −

```
my_string = 'Tutorials Point'
```

MATLAB will execute the above statement and return the following result −

```
my_string = Tutorials Point
```

MATLAB considers all variables as arrays, and strings are considered as character arrays. Let us use the **whos** command to check the variable created above −

```
whos
```

MATLAB will execute the above statement and return the following result −

```
Name         Size          Bytes  Class    Attributes
my_string    1x16             32  char
```

Interestingly, you can use numeric conversion functions like **uint8** or **uint16**to convert the characters in the string to their numeric codes. The **char**function converts the integer

vector back to characters −

Example

Create a script file and type the following code into it −

```
my_string = 'Tutorial''s Point';
str_ascii = uint8(my_string)        % 8-bit ascii values
str_back_to_char= char(str_ascii)
str_16bit = uint16(my_string)       % 16-bit ascii values
str_back_to_char = char(str_16bit)
```

When you run the file, it displays the following result −

```
str_ascii =

  84 117 116 111 114 105 97 108 39 115 32 80 111 105 110 116

str_back_to_char = Tutorial's Point
str_16bit =

  84 117 116 111 114 105 97 108 39 115 32 80 111 105 110 116

str_back_to_char = Tutorial's Point
```

Rectangular Character Array

The strings we have discussed so far are one-dimensional character arrays; however, we need to store more than that. We need to store more dimensional textual data in our program. This is achieved by creating rectangular character arrays.

Simplest way of creating a rectangular character array is by concatenating two or more one-dimensional character arrays, either vertically or horizontally as required.

You can combine strings vertically in either of the following ways −

- Using the MATLAB concatenation operator **[]** and separating each row with a semicolon (;). Please note that in this method each row must contain the same number of characters. For strings with different lengths, you should pad with space characters as needed.

- Using the **char** function. If the strings are of different lengths, char pads the shorter strings with trailing blanks so that each row has the same number of characters.

Example

Create a script file and type the following code into it −

```
doc_profile = ['Zara Ali                    '; ...
               'Sr. Surgeon                 '; ...
               'R N Tagore Cardiology Research Center']
doc_profile = char('Zara Ali', 'Sr. Surgeon', ...
                   'RN Tagore Cardiology Research Center')
```

When you run the file, it displays the following result −

```
doc_profile =
Zara Ali
Sr. Surgeon
R N Tagore Cardiology Research Center
doc_profile =
Zara  Ali
Sr. Surgeon
RN Tagore Cardiology Research Center
```

You can combine strings horizontally in either of the following ways −

- Using the MATLAB concatenation operator, **[]** and separating the input strings with a comma or a space. This method preserves any trailing spaces in the input arrays.

- Using the string concatenation function, **strcat**. This method removes trailing spaces in the inputs.

Example

Create a script file and type the following code into it −

```
name =     'Zara Ali                    ';
position = 'Sr. Surgeon                   ';
worksAt =  'R N Tagore Cardiology Research Center';
```

```
profile = [name ', ' position ', ' worksAt]
profile = strcat(name, ', ', position, ', ', worksAt)
```

When you run the file, it displays the following result −

```
profile = Zara Ali                , Sr. Surgeon                , R N Tagore
Cardiology Research Center
profile = Zara Ali,Sr. Surgeon,R N Tagore Cardiology Research Center
```

Combining Strings into a Cell Array

From our previous discussion, it is clear that combining strings with different lengths could be a pain as all strings in the array has to be of the same length. We have used blank spaces at the end of strings to equalize their length.

However, a more efficient way to combine the strings is to convert the resulting array into a cell array.

MATLAB cell array can hold different sizes and types of data in an array. Cell arrays provide a more flexible way to store strings of varying length.

The **cellstr** function converts a character array into a cell array of strings.

Example

Create a script file and type the following code into it −

```
name =     'Zara Ali                ';
position = 'Sr. Surgeon             ';
worksAt = 'R N Tagore Cardiology Research Center';
profile = char(name, position, worksAt);
profile = cellstr(profile);
disp(profile)
```

When you run the file, it displays the following result −

```
{
  [1,1] = Zara Ali
  [2,1] = Sr. Surgeon
  [3,1] = R N Tagore Cardiology Research Center
```

```
}
```

String Functions in MATLAB

MATLAB provides numerous string functions creating, combining, parsing, comparing and manipulating strings.

Following table provides brief description of the string functions in MATLAB −

Function	Purpose
Functions for storing text in character arrays, combine character arrays, etc.	
blanks	Create string of blank characters
cellstr	Create cell array of strings from character array
char	Convert to character array (string)
iscellstr	Determine whether input is cell array of strings
ischar	Determine whether item is character array
sprintf	Format data into string
strcat	Concatenate strings horizontally
strjoin	Join strings in cell array into single string
Functions for identifying parts of strings, find and replace substrings	
ischar	Determine whether item is character array

isletter	Array elements that are alphabetic letters
isspace	Array elements that are space characters
isstrprop	Determine whether string is of specified category
sscanf	Read formatted data from string
strfind	Find one string within another
strrep	Find and replace substring
strsplit	Split string at specified delimiter
strtok	Selected parts of string
validatestring	Check validity of text string
symvar	Determine symbolic variables in expression
regexp	Match regular expression (case sensitive)
regexpi	Match regular expression (case insensitive)
regexprep	Replace string using regular expression
regexptranslate	Translate string into regular expression
Functions for string comparison	

strcmp	Compare strings (case sensitive)
strcmpi	Compare strings (case insensitive)
strncmp	Compare first n characters of strings (case sensitive)
strncmpi	Compare first n characters of strings (case insensitive)
Functions for changing string to upper- or lowercase, creating or removing white space	
deblank	Strip trailing blanks from end of string
strtrim	Remove leading and trailing white space from string
lower	Convert string to lowercase
upper	Convert string to uppercase
strjust	Justify character array

Examples

The following examples illustrate some of the above-mentioned string functions −

FORMATTING STRINGS

Create a script file and type the following code into it −

```
A = pi*1000*ones(1,5);
sprintf(' %f \n %.2f \n %+.2f \n %12.2f \n %012.2f \n', A)
```

When you run the file, it displays the following result −

```
ans = 3141.592654
```

```
3141.59
+3141.59
   3141.59
000003141.59
```

JOINING STRINGS

Create a script file and type the following code into it −

```
%cell array of strings
str_array = {'red','blue','green', 'yellow', 'orange'};

% Join strings in cell array into single string
str1 = strjoin(str_array, "-")
str2 = strjoin(str_array, ",")
```

When you run the file, it displays the following result −

```
str1 = red-blue-green-yellow-orange
str2 = red,blue,green,yellow,orange
```

FINDING AND REPLACING STRINGS

Create a script file and type the following code into it −

```
students = {'Zara Ali', 'Neha Bhatnagar', ...
        'Monica Malik', 'Madhu Gautam', ...
        'Madhu Sharma', 'Bhawna Sharma',...
        'Nuha Ali', 'Reva Dutta', ...
        'Sunaina Ali', 'Sofia Kabir'};

% The strrep function searches and replaces sub-string.
new_student = strrep(students(8), 'Reva', 'Poulomi')
% Display first names
first_names = strtok(students)
```

When you run the file, it displays the following result −

```
{
  [1,1] = Poulomi Dutta
}
first_names =
{
  [1,1] = Zara
  [1,2] = Neha
  [1,3] = Monica
  [1,4] = Madhu
  [1,5] = Madhu
  [1,6] = Bhawna
  [1,7] = Nuha
  [1,8] = Reva
  [1,9] = Sunaina
  [1,10] = Sofia
}
```

COMPARING STRINGS

Create a script file and type the following code into it −

```
str1 = 'This is test'
str2 = 'This is text'
if (strcmp(str1, str2))
 sprintf('%s and %s are equal', str1, str2)
else
 sprintf('%s and %s are not equal', str1, str2)
end
```

When you run the file, it displays the following result −

```
str1 = This is test
str2 = This is text
ans = This is test and This is text are not equal
```

Possible Questions

Part-A (1 mark-Online Exam)

Part – B (2 marks)

1. What is plotting?
2. Draw a graph that joins the points (4, 3), (5,-2), (2, 0).
3. Write about functions and procedures.
4. What are the colour strings and line styles used in plotting?

Part – C (6 marks)

1. Discuss in detail about plotting and built-in-functions available with the example.
2. Explain in detail about string handling functions with example

Question	Option 1	Option 2	Option 3	Option 4	Answer
When the _____ function is executed, MATLAB opens the Figure window and displays the plot in that window	edit	figure	plotting	plot	plot
Functions receive input data from the program tha tinvokes them through a list of variables called an _____ argument list	input	output	result	fucntion	input
_____ are just collections of MATLAB statements that are stored in a file	function files	script files	legal files	none	script files
A MATLAB function is a special type of _____ that runs in its own independent workspace	G file	M file	MM file	MX file	M file
The _____ statement marks the beginning of the function	structure	function	parameters	none	function
A function is invoked by naming it in an expression together with a list of _____ arguments	formal	informal	argument	actual	actual
The _____ statement is used to terminate the function	stop	finish	end	none	end
The first comment line after the function statement is called the _____ comment line	H1	L1	G1	E1	H1
MATLAB programs communicate with their functionsusing a _____ scheme	pass by value	pass by no values	pass by parameters	none	pass by value
_____ function returns the number of actual input arguments that were used to call the function	nargin	nargout	nargchk	erro	nargin
_____ fucntion returns the number of actual output arguments that were used to call the function	nargin	nargout	nargchk	erro	nargout
_____ funtion returns a standard error message if a function is called with too few or too many arguments	nargin	nargout	nargchk	erro	nargchk
_____ displays warning messasge and continue function excution	nargin	nargout	nargchk	warning	warning
_____ is a special type of memory tha tcan be accessed from any workspace	static memory	dynamic memory	global memory	random memory	global memory
A _____ variable is declared with the global statement	local	global	persistent	protected	global
_____ is a special type of memory that can be accessed only within the function, but is preserved unchanged between calls to the function	static memory	dynamic memory	global memory	persistent memory	persistent memory
_____ are functions whose input arguments include the names of other functions	function files	function functions	sub function	recursive function	function functions
_____ function locates a zero of the function tha tis apssed to it.	fempty	fzero	fnull	fone	fzero
Function _____ evaluates a character string as though it had been typed in the command window	eval	feval	fnull	fone	eval
Function _____ evaluates a named function at a specific input value	eval	feval	fnull	fone	feval
_____ function gives MATLAB functions a chance to construct executable statements during execution	eval	feval	fnull	fone	eval
_____ is a function functions which is used to numerically integrate a function	eval	feval	quad	minbrd	quad
_____ is a function functions which is easy to use function plotter	eval	feval	quad	Ezplot	Ezplot

_____ is a function functions which is used to plot a function by name	eval	feval	fplot	Ezplot	fplot
If more than one function is present in a file, the top function is called as _____	primary function	first function	starting function	sub function	primary function
The _____ function should have the samename as the file it appears in	primary function	first function	starting function	sub function	primary function
_____ are often used to implement utility calcualtions for a main fucntion	primary function	first function	starting function	sub function	sub function
Nested functions are defined within a _____ and they inherit variables defined within the host function	primary function	sub function	host function	recursive function	host function
_____ are functions that reside in subdirectories with the special name private	primary function	private function	host function	recursive function	private function
If a file contains one or more nested functions, then every fucntion in the file must be terminated with an _____ statement	end	close	stop	close all	end
_____ function generates random values from a normal distribution	rand	randn	randu	random	randn
_____ function stops executing a function and return to caller	stop	end	return	back	return
_____ are numbers with both a real and an imaginary component	complex	vector	real	none of the above	complex
the function _____ is used to determine whether a given array is real or complex	isarray	isimg	isreal	real	isreal
If any element of ana rray has an imaginarycomponent then the array is _____	real	imaginary	number	none of the above	imaginary
Function _____ converts the real part of the complex number into the double data type and throws away the imaginary part	imag	real	isreal	numb	real
function _____ converts the imaginary part of a complex number into a real number	imag	real	isreal	numb	imag
A MATLAB string is an array of type _____	int	float	char	double	char
A special function _____ can be used to checkfor character arrys	isstring	ischar	isimg	isint	ischar
Variables may be converted from char dat type to double data type using _____ function	char	int	double	string	double
Variable can be converte dfrom double data type to char data type using _____ function	char	int	double	string	char
_____ function can be used to remove extra blanks from a string when it is extracted from an array	remove	deblank	trim	delete	deblank
Two dimensional character arrays can also be created with function _____	string	character	strvcat	strrev	strvcat
_____ functions concatenates two or more strings ignoring trailing blanks	strrev	strvcat	strcat	strcon	strcat
_____ function determines if two strings are identical	strrev	strcmp	strncmp	stricmp	strcmp
_____ is a type of polar plot in which each value represented by an arrow whose length is proportional to its value	bar plot	compass plot	pie plot	stem plot	compass plot
_____ function determines if the first n characters of two strings are identical	strncmp	strcmp	strcmpi	stricmp	strncmp

_____ determines if the first n characters of two strings are identical ignoring cases	strncmp	strcmp	strncmpi	stricmp	strncmpi
_____ function determines if a character is a letter	isalpha	isletter	ischar	isstring	isletter
A _____ plot is a plot in which each dat avalue is represented by a marker and a line connecting the marker vertically to the x aixs	stair	stem	bar	pie	stem
_____ finds matches for string	strfind	strmatch	strrep	strrrev	strmatch
_____ function replaces onestring with another	strfind	strmatch	strrep	strrrev	strrep
_____ function is used to justify the string	strjust	strmatch	strrep	strrrev	strjust
A ____ plot is a plot in which each point is represented by a vertical bar or horizontal bar	stair	stem	bar	pie	bar
_____ functions removes any extra leading and trailing whitespace from a string	deblank	strtrim	strrev	strrep	strtrim
_____ function converts a double value into a string	num2str	int2str	str2num	none of the above	num2str
dec2hex converts a _____ value into corresponding hexadecimal string	integer	double	long int	none of the above	double
MATLAB function _____ converts an array to a string that MATLAB can evaluate	mat2int	str2mat	mat2str	none of the above	mat2str
In _____ function the output goes into a character string instead of the command window	fprintf	sprintf	printf	print	sprintf

Unit – IV

Syllabus

Control Statements: Conditional statements: If, Else, Else-if, Repetition statements: While, for loop

MATLAB is also a *programming language*. Like other computer programming languages, MATLAB has some decision making structures for control of command execution. These decision making or *control flow* structures include for loops, while loops, and if-else-end constructions. Control flow structures are often used in script M-files and function M-files. By creating a file with the extension .m, we can easily write and run programs.

Control flow

MATLAB has four control flow structures: the if statement, the for loop, the while loop, and the switch statement.

The ‘‘if...end’’ **structure**

MATLAB supports the variants of “if” construct.

if ... end

if ... else ... end

if ... elseif ... else ... end

The simplest form of the if statement is

```
if expression
statements
end
```

Here are some examples based on the familiar quadratic formula.

1.
```
discr = b*b - 4*a*c;
if discr < 0
disp(’Warning: discriminant is negative, roots are
imaginary’);
end
```

2.
```
discr = b*b - 4*a*c;
```

```
if discr < 0
disp('Warning: discriminant is negative, roots are
imaginary');
else
disp('Roots are real, but may be repeated')
end
3. discr = b*b - 4*a*c;
if discr < 0
disp('Warning: discriminant is negative, roots are
imaginary');
elseif discr == 0
disp('Discriminant is zero, roots are repeated')
else
disp('Roots are real')
end
```

It should be noted that:

- elseif has no space between else and if (one word)
- no semicolon (;) is needed at the end of lines containing if, else, end
- indentation of if block is not required, but facilitate the reading.
- the end statement is required

The Nested if Statements

It is always legal in MATLAB to nest if-else statements which means you can use one if or elseif statement inside another if or elseif statement(s).

Syntax

The syntax for a nested if statement is as follows:

```
if <expression 1>
% Executes when the boolean expression 1 is true
if <expression 2>
```

```
% Executes when the boolean expression 2 is true
end
end
```

You can nest elseif...else in the similar way as you have nested if statement.

Example

Create a script file and type the following code in it:

```
a = 100;
b = 200;
% check the boolean condition
if( a == 100 )
% if condition is true then check the following
if( b == 200 )
        % if condition is true then print the following
fprintf('Value of a is 100 and b is 200\n' );
end
end
fprintf('Exact value of a is : %d\n', a );
fprintf('Exact value of b is : %d\n', b );
```

When you run the file, it displays:

```
Value of a is 100 and b is 200
Exact value of a is : 100
Exact value of b is : 200
```

The Switch Statement

- A switch block conditionally executes one set of statements from several choices.
- Each choice is covered by a case statement.
- An evaluated switch_expression is a scalar or string.
- An evaluated case_expression is a scalar, a string or a cell array of scalars or strings.

- The switch block tests each case until one of the cases is true. A case is true when:
- For numbers, eq(case_expression,switch_expression).
- For strings, strcmp(case_expression,switch_expression).
- For objects that support the eq function,eq(case_expression,switch_expression).
- For a cell array case_expression, at least one of the elements of the cell array matches switch_expression, as defined above for numbers, strings and objects.
- When a case is true, MATLAB executes the corresponding statements and then exits the switch block.
- The **otherwise** block is optional and executes only when no case is true.

Syntax

The syntax of switch statement in MATLAB is:

```
switch <switch_expression>
case <case_expression>
        <statements>
case <case_expression>
        <statements>
...
...
otherwise
        <statements>
end
```

Example

Create a script file and type the following code in it:

```
grade = 'B';
switch(grade)
case 'A'
        fprintf('Excellent!\n' );
case 'B'
        fprintf('Well done\n' );
```

```
case 'C'
    fprintf('Well done\n' );
case 'D'
    fprintf('You passed\n' );
case 'F'
   fprintf('Better try again\n' );
otherwise
    fprintf('Invalid grade\n' );
end
```

When you run the file, it displays:

Well done

Your grade is B

The ‘‘for...end’’ **loop**

In the for ... end loop, the execution of a command is repeated at a fixed and predetermined number of times. The syntax is

```
for variable = expression
  statements
end
```

Usually, expression is a vector of the form i:s:j. A simple example of for loop is

```
for ii=1:5
x=ii*ii
end
```

It is a good idea to indent the loops for readability, especially when t hey are nested. Note that MATLAB editor does it automatically.

Multiple for loops can be nested, in which case *indentation* helps to improve the readability. The following statements form the 5-by-5 symmetric matrix A with (*i, j*) element *i/j* for $j \geq i$:

```
n = 5; A = eye(n);
for j=2:n
    for i=1:j-1
        A(i,j)=i/j;
        A(j,i)=i/j;
     end
end
```

The ‘‘while...end’’ loop

This loop is used when the number of *passes* is not specified. The looping continues until a stated condition is satisfied. The while loop has the form:

```
while expression
   statements
end
```

The statements are executed as long as expression is true.

```
x = 1
while x <= 10
x = 3*x
end
```

It is important to note that if the condition inside the looping is not well defined, the looping will continue *indefinitely*. If this happens, we can stop the execution by pressing **Ctrl-C**.

Other Flow Structures

(i) break

Terminate execution of for or while loop

Syntax

```
break
```

Description

break terminates the execution of a for or while loop. Statements in the loop after the break statement do not execute. In nested loops, break exits only from the loop in which it

occurs. Control passes to the statement that follows the end of that loop.

Example

```
limit = 0.8;
s = 0;

while 1
   tmp = rand;
   if tmp > limit
      break
   end
   s = s + tmp;
end
```

(ii) continue

Pass control to next iteration of for or while loopcollapse all in page

Syntax

```
continue
```

Description

continue passes control to the next iteration of a for or while loop. It skips any remaining statements in the body of the loop for the current iteration. The program continues execution from the next iteration.

continue applies only to the body of the loop where it is called. In nested loops, continue skips remaining statements only in the body of the loop in which it occurs.

Example

```
for n = 1:50
   if mod(n,7)
      continue
   end
   disp(['Divisible by 7: ' num2str(n)])
end
```

Output

```
Divisible by 7: 7
Divisible by 7: 14
Divisible by 7: 21
```

Divisible by 7: 28
Divisible by 7: 35
Divisible by 7: 42
Divisible by 7: 49

(iii) end

Terminate block of code, or indicate last array indexcollapse all in page

Syntax

end

Description

end terminates for, while, switch, try, if, and parfor statements. end also marks the termination of a function.

Example

Use end to close a for loop and an if statement.

```
a = [0 0 1 1 0 0 0 1 0];
for k = 1:length(a)
   if a(k) == 0
      a(k) = 2;
   end
end
```

Indexing Expressions

Try this Example

Access the last row of a matrix A using end.

```
A = magic(3)
A =

    8    1    6
    3    5    7
    4    9    2

B = A(end,1:end)
B =
```

(iv) pause

Stop MATLAB execution temporarily

Syntax

```
pause
pause(n)
pause(state)
oldState = pause(state)
```

Description

pause temporarily stops MATLAB execution and waits for the user to press any key. The pause function also temporarily stops the execution of Simulink models, but does not pause their repainting.

Example

Pause Execution

Pause execution for 5 seconds. MATLAB blocks, or hides, the command prompt (>>) while it pauses execution.

```
n = 5;
pause(n)
```

Disable Pause Setting

Disable the pause setting and query the current state.

```
pause('off')
pause('query')
ans =
'off'
```

Pause execution for 100 seconds. Since the pause setting is off, MATLAB ignores the request to pause execution, and immediately returns the command prompt.

```
pause(100)
```

Enable the pause setting.

```
pause('on')
```

Save and Restore Pause State

Store the current pause setting and then disable the ability to pause execution.

```
oldState = pause('off')
oldState =
```

'on'

Query the current pause setting.

```
pause('query')
ans =
'off'
```

Restore the initial pause state.

```
pause(oldState)

pause('query')

ans =
'on'
```

Alternatively, you can store the queried value of the pause state and then disable the ability to pause execution.

```
oldState = pause('query');
pause('off')
```

Restore the initial pause state.

```
pause(oldState)
```

(v) return

Return control to invoking function

Syntax

```
return
```

Description

return forces MATLAB to return control to the invoking function before it reaches the end of the function. The invoking function is the function that calls the script or function containing the call to return. If you call the function or script that contains return directly, there is no invoking function and MATLAB returns control to the command prompt.

Example

```
function idx = findSqrRootIndex(target,arrayToSearch)
idx = NaN;
if target < 0
  return
end
```

```
for idx = 1:length(arrayToSearch)
   if arrayToSearch(idx) == sqrt(target)
      return
   end
end
```

Possible Questions

Part-A (1 mark-Online Exam)

Part –B (2 marks)

1. Write a vote program using if else condition.
2. What is loop?
3. List the types of loops in MATLAB.
4. Write about the types of unconditional statements.

Part – C (6 marks)

1. Discuss about the conditional statements with syntax and example.
2. Discuss about the repetition statements with syntax and example.

Question	Option 1	Option 2	Option 3	Option 4	Answer
When are statements are eexcute done by one it is called as _____ program	sequential	procedural	logical	none	sequential
_____ is the process of starting with a large task and breaking it down into smaller, to perform the desired task alone	top down design	top desgin	bottom up	bottom top	top down design
An _____ is a step by step procedure for finding the solution to a problem	process	processing	algorithm	pseducode	algorithm
The first stage of testing is also called as _____	subtask testing	task testing	unit testing	branch testing	unit testing
During _____ the individual subtasks of the program are tested separately to confirm that they work correctly	subtask testing	task testing	unit testing	branch testing	unit testing
The complete versuion of the program is ususally called the _____	beta release	alpha release	gamma release	none	alpha release
when the most serious bugs have been removed from the program, a new version is called as is prepared	beta release	alpha release	gamma release	none	beta release
the standard forms that is used to describe algorithms are called _____	datas	programs	constructs	procedure	constructs
An algorithm described by using constructs are called _____ algorithm	procedural	structured	object oriented	pseudocode	structured
when the algorithm is implemented in a MATLAB program, the resulting program is called a program	procedural	structured	object oriented	pseudocode	structured
The constructs used to build alogirthm can be described in a special way called _____	functioning	structures	union	pseudocode	pseudocode
_____ is a hybrid mixture of MATLAB and english	Pseudocode	structures	union	none	Pseudocode
The _____ data type is a special type of data that can have one of only two possible values true or false	arithmetic	logical	both a&b	none of the above	logical
Logical values takes _____ bytes	16	64	8	12	8
The _____ function converts numerical data to logical data	real	logical	relation	array	logical
the _____ fucntion converts logical data to numerical data	real	logical	relation	array	real
The _____ operators are operators with two numerical or string operands that yield a logical result	logical	relational	bitwise	arithmetic	relational
The relational operators can compare two strings only if they are of _____ length	equal	different	both equal and different	equ	equal
_____ are MATLAB statements that permit us to select and execute specific sections of code whiel skipping other sections	looping	branches	structures	none	branches
_____ permits a programmer to seelct a perticular code block to execute based on the value of a single integer, character or logical	if	try	switch	if else	switch

The _____ construct is a special form branching construct designed to trap errors	try/catch	switch	if	if else	try/catch
When an error occurs in the try block, it immediately excutes the stements in the block	else	while	catch	none	catch
The statements in the _____ block will always be excuted	catch	else	try	if else	try
The statements in the _____ block will only be executed of an error occurred in try block	catch	else	try	if else	catch
The _____ command is used to display only the subset of the data	axes	axis	plot	plotting	axis
The _____ command sets the axis increments to be equal on both axes	axis normal	axis square	axis on	axis equal	axis equal
The _____ command makes the current axis box square	axis normal	axis square	axis on	axis equal	axis square
the _____ command cancels the effect of axis equal and axis sqaure	axis normal	axis square	axis on	axis equal	axis normal
When a _____ command is used the additional plots will be laid on top of the previously existing plots	hold on	hold off	holded on	none	hold on
A _____ command switches plotting behaviour back to the default situation in which a new plot repalces the previous one	hold on	hold off	holded on	none	hold off
Each figure is identifed by the _____	window number	screen number	figure number	picture number	figure number
the current figure is selected with the _____ fucntion	window	figure	subplot	plotting	figure
The function _____ returns the number of the current figure	acf	gaf	gcf	agf	gcf
A _____ is a special sequence of characters that ells the MATLAB interperter top change its behaviour	stream modifier	modifier	online modifier	file modifier	stream modifier
_____ is a stream modifier which replces the normal font	\rm	\rrf	\rf	\fr	\rm
_____ plots data in polar corodinates	pole	polar	plot	poles	polar
_____ are MATLAB constructs that permit us to execute a sequence of statements more than once	branches	loops	structures	union	loops
A _____ loop is a block of statements that are repeated indefintiely as long as some condition is satisified	do while	while	do	for	while
The _____ loop is a loop that executes a block of statements a specified number of times	do while	while	do	for	while
The _____ of a for loop should not be modified anywhere within the body of the loop	body	loop index	loop expression	none	loop index
In MATLAB, the process of replacing loops by vectorized statements is known as _____	scalarization	vectorization	looping	branching	vectorization

The JIT compiler helps to speed up the execution of _____ loops	do while	while	for	if	for
The _____ statement terminates the execution of a loop and passes control to the next statement after the end of the loop	break	continue	skip	end	break
the _____ statement terminates the current pass through the loop and return control to the top of the loop	break	continue	skip	end	continue
If one loop is completely inside another one, the two loops are called _____ loops	double	grouping	nesting	none	nesting
when MATLAB encounters an _____ statement, it associates that statement with the innermost currently open construct	break	continue	end	skip	end
If _____ loops are nested, they should have independent loop index variables.	do while	while	if	for	for
If a break or continue statement appears inside a set of nested loops, then that statement refers to the of the loops containing	innermost	outermost	top	bottom	innermost
Scalars and arrays of _____ data are created as the output of relational and logic oeprators	vectorizatio n	arithmetic	logical	none	logical
_____ arrays can serve as a mask for arithmetic oeprations	logical	arithmetic	relational	none	logical
A _____ is an array that selects the elements of another array for use in an operation	set	vector	mask	unmask	mask
when the increment value of the index is not mentioned, it is taken as _____ by default	2	1	3	4	1
The _____ block in switch structure is optional	default	otherwise	other	else	otherwise
In _____ structure, case can have multiple values	switch case	if else	while	for	switch case
If the value of switch variable is _____, then it must be entered within single quotes	integer	double	float	character	character
_____ is used to terminate the program due to incorrect input and gives the error message	break	continue	error	none	error

Unit V

Syllabus

Manipulating Text: Writing to a text file, Reading from a text file, Randomising and sorting a list, searching a list. **GUI Interface**: Attaching buttons to actions, Getting Input, Setting Output

Writing to a text file

To save the results of some computation to a file in text format requires the following steps:

a. Open a new file, or overwrite an old file, keeping a 'handle' for the file.
b. Print the values of expressions to the file, using the file handle
c. Close the file, using the file handle

The file handle is a just a variable which identifies the open file in your program. This allows you to have any number of files open at any one time.

```
% open file
fid = fopen('myfile.txt','wt');     % 'wt' means "write text"
if (fid < 0)
   error('could not open file "myfile.txt"');
end;
% write some stuff to file
for i=1:100
   fprintf(fid,'Number = %3d Square = %6d\n',i,i*i);
end;
% close the file
fclose(fid);
```

Reading from a text file

To read some results from a text file is straightforward if you just want to load the whole file into memory. This requires the following steps:

a. Open an existing file, keeping a 'handle' for the file.
b. Read expressions from the file into a single array, using the file handle
c. Close the file, using the file handle

The fscanf() function is the inverse of fprintf(). However it returns the values it reads as values in a matrix. You can control the 'shape' of the output matrix with a third argument.

```
A = fscanf(fid,"%g %g %g\n",[3,inf])      % A has 3 rows and 1 col per line
disp(A(1,1))        % display first value on first line
disp(A(1,2))        % display first value on second line
disp(A(2,1))        % display second value on first line
```

Thus to read back the data we saved above:

```
% open file
fid = fopen('myfile.txt','rt');     % 'rt' means "read text"
if (fid < 0)
   error('could not open file "myfile.txt"');
end;
% read from file into table with 2 rows and 1 column per line
tab = fscanf(fid,'Number = %d Square = %d\n',[2,inf]);
% close the file
fclose(fid);
rtab = tab';          % convert to 2 columns and 1 row per line
```

Reading a table of strings is more complex, since the strings have to be the same length. We can use the fgetl() function to get a line of text as characters, but we'll first need to find out the length of the longest string, then ensure all strings are the same length. Here is a complete function for loading a text file as a table of fixed-length strings:

```
function tab=readtextfile(filename)
% Read a text file into a matrix with one row per input line
% and with a fixed number of columns, set by the longest line.
% Each string is padded with NUL (ASCII 0) characters
%
% open the file for reading
ip = fopen(filename,'rt');          % 'rt' means read text if (ip <
0)
   error('could not open file'); % just abort if error end;
% find length of longest line
max=0;                          % record length of longest string
cnt=0;                           % record number of strings
s = fgetl(ip); while            % get a line
(ischar(s))                           % while not end of file
  cnt = cnt+1;
  if (length(s) > max)             % keep record of longest
     max = length(s);
  end;
   s = fgetl(ip); end;          % get next line
% rewind the file to the beginning frewind(ip);
% create an empty matrix of appropriate size
tab=char(zeros(cnt,max));          % fill with ASCII zeros
% load the strings for real cnt=0;
s = fgetl(ip);
```

```
while (ischar(s))
  cnt = cnt+1;
  tab(cnt,1:length(s)) = s;      % slot into table
   s = fgetl(ip);
end;
% close the file and return
fclose(ip);
return;
```

Here is an example of its use:

```
% write some variable length strings to a file
op = fopen('weekdays.txt','wt');
fprintf(op,'Sunday\nMonday\nTuesday\nWednesday\n');
fprintf(op,'Thursday\nFriday\nSaturday\n');
fclose(op);
% read it into memory
tab = readtextfile('weekdays.txt');
% display it
disp(tab);
```

Randomising and sorting a list

Assuming we have a table of values, how can we randomise the order of the entries? A good way of achieving this is analogous to shuffling a pack of cards. We pick two positions in the pack, then swap over the cards at those two positions. We then just repeat this process enough times that each card is likely to be swapped at least once.

```
function rtab=randomise(tab)
% randomise the order of the rows in tab.
% columns are unaffected
[nrows,ncols]=size(tab);          % get size of input matrix
cnt = 10*nrows;                   % enough times
while (cnt > 0)
   pos1 = 1+fix(nrows*rand);      % get first random row
   pos2 = 1+fix(nrows*rand);      % get second random row
   tmp = tab(pos1,:);             % save first row
   tab(pos1,:) = tab(pos2,:);     % swap second into first
   tab(pos2,:) = tmp;             % move first into second
   cnt=cnt-1;
end;
rtab=tab;                         % return randomised table
return;
```

Sorting a list is easy if you just want some standard alphabetic ordering. But what if you want to

choose some arbitrary ordering function? For example, how could you sort

strings when case was not important? Here we use the ability of MATLAB to evaluate a function by name (feval()) so that we can provide the name of a function for doing the comparisons the way we want. This function should take two rows and return –1 if the first row sorts earlier than the second, 1 if the second row sorts earlier than the first and 0 if there is no preference. Here is a case-independent comparison function:

```
function flag=comparenocase(str1,str2)
% compares two strings without regard to case
% returns –1, 0, 1 if str1 is less than, equal, greater than str2.
len1=length(str1);
len2=length(str2);
for i=1:min(len1,len2)
   c1 = str1(i);
   c2 = str2(i);
   if (('a' <= c1)&(c1 <= 'z'))
      c1 = char(abs(c1)-32);          % convert lower case to upper
   end;
   if (('a' <= c2)&(c2 <= 'z'))
      c2 = char(abs(c2)-32);          % convert lower case to upper
   end;
   if (c1 < c2)
      flag = -1;                  % str1 sorts earlier
      return;
   elseif (c2 < c1)
      flag = 1;                   % str2 sorts earlier
      return;
   end;
end;
% strings match up to length of shorter, so
if (len1 < len2)
   flag = -1;                     % str1 sorts earlier
elseif (len2 < len1)
   flag = 1;                      % str2 sorts earlier
else
   flag = 0;                      % no preference
end;
return;
```

Here is a sort function that might be used with this comparison function.

```
function stab=functionsortrows(tab,funcname)
% sorts the rows of the input table using the supplied
% function name to provide an ordering on pairs of rows
[nrows,ncols]=size(tab);
for i=2:nrows                    % sort each row into place
   j = i;
   tmp = tab(j,:);               % save row
   % compare this row with higher rows to see where it goes
   while ((j > 1)&(feval(funcname,tmp,tab(j-1,:))<0))
      tab(j,:) = tab(j-1,:);     % shift higher rows down
      j = j - 1;
   end;
   tab(j,:) = tmp;               % put in ordered place
end;
stab = tab;                      % return sorted table
return;
```

Searching a list

How might we search a list of items for an item matching a specific value? If the list is unordered, all we can do is run down the list testing each entry in turn. This function finds the index of a row in a table that contains (anywhere) the characters in the supplied match string:

```
function idx=findstring(tab,str)
% find the row index containing a matching string
% returns 0 if the string is not found
[nrows,ncols]=size(tab);
for idx=1:nrows
   matches = findstr(tab(idx,:),str);
   if (length(matches)>0)
      return;
   end;
end;
idx=0;
return;
```

However, the process can be much faster if the listed is sorted and we are searching for an exact match only. A so-called binary search is the fastest possible way of finding an item in a sorted list:

```
function idx=binarysearch(tab,val)
% returns the row index of val in sorted table tab
% returns 0 if val is not found
[nrows,ncols]=size(tab); lo=1;
hi=nrows; while
(lo <= hi)
   idx = fix(lo+hi)/2; if
   (val < tab(idx,:))
       hi = idx - 1;
   elseif (val > tab(idx,:))
       lo = idx +
    1; else
       retur
    n; end;
end;
idx=
0;
```

GUI Interface

Elements of a Graphical User Interface

By a graphical user interface, we mean that we can give a MATLAB program the look and feel of a typical Windows application. The MATLAB GUI design system allows you to create applications consisting of one or more 'dialogs' containing typical 'controls' such as buttons, edit boxes, lists and pictures.

One of the important aspects of a Windows application that is unlike the kind of programs we have considered up to now is that they interact asynchronously with the user. The user can select any function of the program at any time. This means that you need to store the 'state' of your program in a set of variables and be prepared to execute any function based on the current state at any time.

The MATLAB GUI design system helps you in this by associating functions with each element of the dialog. Thus when you press a button, click on a menu, or enter a number in an edit box, you can arrange for a function in your program to be called. Your task is to program the actions related to that function, e.g. opening a file, playing a sound, or displaying the results of a calculation.

The most common controls are:

- Menu options. Selection calls up an operation by name. Push
- buttons. Clicking calls up some operation.
- Edit boxes. User can enter some text or numerical value. List boxes.

User can choose among list of items.
Figures. Program can display graphical results. Text.
Program can display textual result.

You can use the controls themselves to store data or you can create a set of global variables.

How to build a simple dialogue

To start the design program type 'guide' at the MATLAB prompt. You are presented with a blank form upon which you can position controls. Choose a control from the palette and click and size the control on the page to position it. Each control is automatically given a name based on its type.

When the layout is complete, you can save the design to a '.fig' file. This will automatically create a matching '.m' program file which you can use to launch the application and store the code that is operated by the controls. It is not necessary to store all your code in the matching '.m' file; indeed it is a good idea to break up any large sections of code into its own function blocks stored in separate files. You will see that the layout designer builds a 'callback' function prototype in the program file for each control that provides input to the application. This function will be called automatically when that control is activated.

You can edit the properties of the controls on the layout editor by right-clicking on them and choosing 'Property Inspector'. In particular the 'String' property is used to store the default text for buttons, list boxes and edit boxes. The 'Tag' property is the name of the control; and until you are familiar with MATLAB, it is advisable not to change the default name. You can also use the Property Inspector to change the name of the dialog itself.

You can add menu options to your dialog with the 'Menu Editor'. If you leave the callback function entry as "%automatic", then the menu editor adds callback functions to your program for each menu item. Otherwise create your own callback function using existing ones as a model, and associate a call to the function with the menu item manually.

It is important to realise that the '.m' file associated with your application is executed afresh each time there is some event in the dialog. That is you must store the 'current state' of the program in global variables in the workspace, and not in variables local to a function. You can ensure this by using a 'global'statement and initialising them in the part of the file where the figure is initialised.

You can access any property of any control using the 'Tag' property of the control and the MATLAB 'get()' and 'set()' functions.

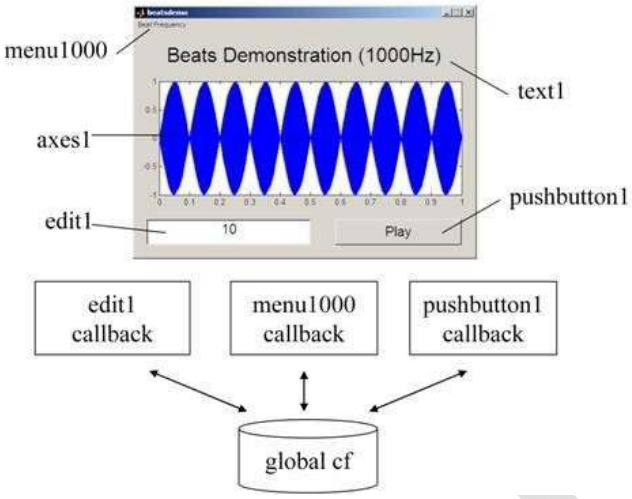
```
value = get(handles.ControlTagName,'PropertyName');
set(handles.ControlTagName,'PropertyName','Value');
```

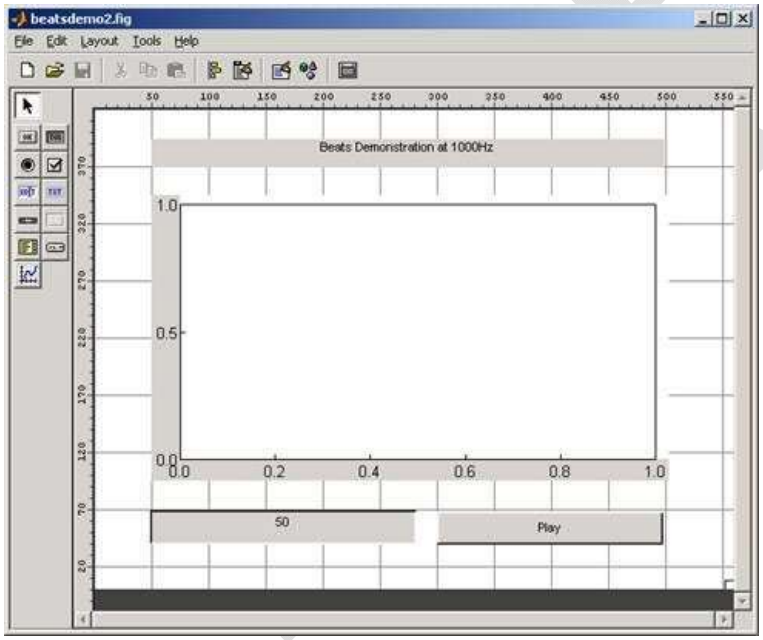
For example:

```
text = get(handles.edit1,'String'); set(handles.edit1,'String','100');
```

Note that most properties have to be get() and set() as strings. Use the num2str() and str2num() functions to help convert between strings and numeric values.

Worked example






```
function varargout =
%     BEATSDEMO, by itself, creates a new BEATSDEMO or raises the
%
%     H = BEATSDEMO returns the handle to a new BEATSDEMO or the
%     handle
```

```
%
%      BEATSDEMO('CALLBACK',hObject,eventData,handles,...) calls the local
%       function named CALLBACK in BEATSDEMO.M with the given input
arguments.
%
%       BEATSDEMO('Property','Value',...) creates a new BEATSDEMO or raises
the
%      existing singleton*. Starting from the left, property value pairs are
%      applied to the GUI before beatsdemo_OpeningFunction gets called. An
%      unrecognized property name or invalid value makes property application
%      stop. All inputs are passed to beatsdemo_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu. Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help beatsdemo

%      Last      Modified      by      GUIDE      v2.5      11-Dec-2005      15:46:27

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton', gui_Singleton, ...
                   'gui_OpeningFcn', @beatsdemo_OpeningFcn, ...
                   'gui_OutputFcn', @beatsdemo_OutputFcn, ...
                   'gui_LayoutFcn', [] , ...
                   'gui_Callback', []);
if nargin & isstr(varargin{1})
   gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
   [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
   gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT

% --- Executes just before beatsdemo is made visible.
function beatsdemo_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
```

```
% eventdata reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin command line arguments to beatsdemo (see VARARGIN)

% Choose default command line output for beatsdemo
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

%%%ADDED%%%
% initialise global variable cf
global cf;
cf=1000;
%%%%%%%%%%

% --- Outputs from this function are returned to the command line.
function varargout = beatsdemo_OutputFcn(hObject, eventdata, handles)
% varargout cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;
% --- Executes during object creation, after setting all properties.
function edit1_CreateFcn(hObject, eventdata, handles)
% hObject handle to edit1 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc
    set(hObject,'BackgroundColor','white');
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end

%%%ADDED%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%% Callback Functions %%%%%%%%%%%%%%%%%%%%%%%%%%%%
function edit1_Callback(hObject, eventdata, handles)
% hObject handle to edit1 (see GCBO)
```

```
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)
%
% just call the pushbutton if text changes ...
pushbutton1_Callback(hObject,eventdata,handles);
return;
% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject handle to pushbutton1 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)
%
% reference global variable
global cf;
% get beat frequency from text box
bf=str2num(get(handles.edit1,'String'));
% make beats by combining two sinewaves
t=0:1/11025:1;
y=0.5*sin(2*pi*(cf-bf/2)*t)-0.5*sin(2*pi*(cf+bf/2)*t);
sound(y,11025);
% set default axes to figure and plot
axes(handles.axes1);
plot(t,y);
return;

% --------------------------------------------------------------------
function menu1000_Callback(hObject, eventdata, handles)
% hObject handle to f1000 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)
global cf;
cf=1000;
set(handles.menu1000,'Checked','on');
set(handles.menu2000,'Checked','off');
set(handles.text1,'String','Beats Demonstration (1000Hz)');
return;
% --------------------------------------------------------------------
function menu2000_Callback(hObject, eventdata, handles)
% hObject handle to f2000 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)
global cf;
cf=2000;
set(handles.menu1000,'Checked','off');
```

```
set(handles.menu2000,'Checked','on');
set(handles.text1,'String','Beats Demonstration (2000Hz)');
return;
```

Possible Questions

Part-A (1 mark-Online Exam)

Part – B (2 marks)

1. What is GUI based application? Give any 2 examples.
2. How to use sort rows in MATLAB.
3. List any 6 tools in GUI.

Part – C (6 marks)

1. Explain about GUI interfaces with examples.
2. Write about randomizing and sorting a list with example.
3. Discuss about GUI and give an example: attaching buttons to action.
4. Define File. How to read and write files give an example.
5. Discuss about step-by-step process to create a GUI based application in MATLAB.

Question	Option 1	Option 2	Option 3	Option 4	Answer
The MATLAB graphics system is based on a hierarchical system of core _____	graphics	system	graphics objects	properties	graphics objects
Each graphics object is known by aunique number called a _____	handle	object	term	component	handle
Each gaphics object has special data known as _____ associated with it	object	properties	term	component	properties
the highest level graphics object in MATLAB is the _____	directory	figures	root	path	root
Each _____ is a separetate window on the computer screen that can display graphical data	figure	plot	handle	object	figure
Each figure can contain _____ types of objects	six	eight	two	figure	figure
_____ are regions within a figure where data isactually plotted	axes	area	data	plane	axes
The _____ is aunique integer or real number that is used by MATLAB to identify the object	handle	object	term	component	handle
Each property has a _____ and an associated value	term name	component name	property name	none of the above	property name
When an _____ is created all of its poperties are automatically initialized to default values	handle	term	object	data	object
The properties of any object can be examined at any time using the _____ function	set	plot	get	end	get
The properties of any object can be modified at any time using the _____ function	set	plot	get	end	set
The _____ function can be used to provide lists of possible property values	set	plot	get	end	set
the function _____ will return all of the possible choices for all the properties of an object	set()	set(hndl)	get()	get(hndl)	set(hndl)
Data values are stored in an object with the _____ function	setapp	set	setappdata	appdata	setappdata
Data values can be retrived from the object using the _____ function	getapp	get	appdata	getappdata	getappdata
_____ returns the handle of the current figure	gcf	gca	gco	findobj	gcf
_____ returns the handle of the current axes in the current figure	gcf	gca	gco	findobj	gca
_____ returns the handle of the current object	gcf	gca	gco	findobj	gco
_____ finds a graphics object with a specified property value	gcf	gca	gco	findobj	findobj
The MATLAB objects have a _____ property which specifies the size and position of the object on the computer screen	position	size	possize	none of the above	position
the position of axes and unicontrol objects is specified by a _____ vector	2 element	3 element	4 element	5 element	4 element
_____ objects have a position property containing only two or three elements	text	value	figure	frame	text
The _____ and units properties specify the location of a figure on the computer screen	position	value	figure	handle	position

The string _____ allows a user to temporarily override a default value and use the original MATLAB default value instead	color	factory	figure	style	factory
A program that response to event sis said to be _____	program driven	event driven	events	none of the above	event driven
Graphical contraols and text boxes are created by the function _____	figure	plot	uicontrol	control	uicontrol
Toolbars are created by the fucntion	utool	uitoolbar	uimenu	uiaxes	uitoolbar
A _____ is a window on the computer screen	figure	container	plot	workspace	container
a most common container is a _____	figure	workspace	plot	area	figure
_____ can contain components or other containers	callbacks	panel	button group	component	panel
The code executed in response to an event is known as a _____	callback	program event	programs	returnback	callback
MATLAB graphical user interfaces are created using a tool called _____	simulink	images	guide	help	guide
The _____ window has a palette of GUI components along the left hand side of the layout area	Layout area	Layout space	Layout Editor	Layout workspace	Layout Editor
The function _____ returns the ahandle of the object that generated the callback	gcbo	gcbf	gcb	gcf	gcbo
the function _____ returns the handle of the figure containing that object	gcbo	gcbf	gcb	gcf	gcbf
A _____ is a graphical object that displays one or more text strings, which are specified in the text field's string property	dynamic text field	text field	static text field	none of the above	static text field
a text field is created by _____	toolbox	uitoolbar	uicontrol	control box	uicontrol
An _____ is a graphical object that allows a user to enter one or more text strings.	static text	tool box	edit boxes	menus	edit boxes
A _____ is a component that a user can click on to trigger a specific action	pushbutton	tool box	static text field	menus	pushbutton
a _____ is a type of button that has two states on and off	pushbutton	tool box	static text box	toggle buttons	toggle buttons
_____ are graphical objects that display many lines of text and allow a user to select one or more of those lines	toolbox	pushbutton	toggle button	list boxes	list boxes
panels are created by the function _____	unipanel	upanel	uipanel	panel	uipanel
A _____ is a special type of figure that is sued to display information or to get input from a user	toolbox	dialog boxes	toggle button	menus	dialog boxes
_____ may be modal or non modal	toolbox	dialog boxes	toggle button	menus	dialog boxes
_____ boxes are typically used for warning and error messages	non modal	modal	text boxes	list boxes	modal
_____ boxes prompt a user to enter one or more values that may be used by a program	output dialog	input dialog	text boxes	list boxes	input dialog
The _____ dialog boxes allows a user to interactively select a directory	uiget	unisetdir	uigetdir	dirname	uigetdir
If the user cancels the dialog box, _____ is set to zero	directoryname	pathname	filename	figurename	directoryname

A _____ allows a user to select actions without additional components appearing on the GUI display	tools	list box	menus	dialog boxes	menus
_____ menus are the pulled down from the menu bar at the top of a figure	context	standard	linear	collinear	standard
_____ menus are pop up over the figure when a user right clicks the mouse over a graphical object	context	standard	linear	collinear	context
Accelerator keys are _____ combinations that cause a menu item to be executed without opening the menu first	CTRL + key	ALT + key	TAB+ key	DEL+Key	CTRL + key
_____ are single letters that can be presses to cause a menu item to execute once the menu is open	Shortcut key	Keyboard mnemonics	Acclerator keys	none of the above	Keyboard mnemonics
_____ create a generic dialog box	arrdialog	create dialog	dialog	errdialog	dialog
_____ function is used to create a standard menu, or amenu item on either a standard menu or a context menu	menus	create menu	uimenu	unicreate	uimenu
_____ is used to create a user defined toolbar	uimenu	unitools	toolbar	unitoolbar	unitoolbar
_____ is used to create a dialog box to ask a question	inputdlg	questdlg	question	dialog boxes	questdlg
_____ is used to print the dialog box	inputdlg	printdlg	questdlg	errordlg	printdlg
_____ and keyboard mnemonics can be used to speed the operations of windows	Shortcut key	Keyboard mnemonics	Acclerator keys	none of the above	Keyboard mnemonics

KARPAGAM ACADEMY OF HIGHER EDUCATION
(Deemed to be University)
(Established Under Section 3 of UGC Act 1956)
COIMBATORE – 641 021

COMPUTER TECHNOLOGY

Third Semester
FIRST INTERNAL EXAMINATION - July 2018

PROGRAMMING IN MATLAB

Class & Section: II B.Sc (CT) **Duration: 2 hours**
Date & Session : **Maximum marks: 50 marks**
Subj.Code: 17CTU304B

PART- A (20 * 1= 20 Marks)
Answer ALL the Questions

1. Software designed to help the user to perform specific tasks is _____
   a) System software c) Operating Systems
   b) Application software d) Utility programs
2. The programs written in machine language are_____
   a) Machine independent c) Machine interconnected
   b) Machine dependent d) Machine interface
3. Assembly language consists of _____
   a) Mnemonics b) Source code c) Object code d) Byte Code
4. Assemblers perform a _____
   a) One to One translation c) Many to Many translation
   b) One to Many translation d) Many to One translation
5. Tool that produces input for compilers_____
   a) Loader c) Preprocessor
   b) Interpreter d) Wordprocessor
6. Computer program that links and merges various object files together in order to make an executable file_____
   a) Dynamic Linker c) Loader
   b) Linker d) Static Linker
7. High Level Languages are belongs to which generation_____
   a) First generation c) Fourth generation
   b) Second generation d) Third generation
8. Functional languages treat procedures like_____
   a) Mathematical Functions c) Symbols
   b) English like words d) Charaters
9. An example of Functional language_____
   a) COBOL b) FORTRAN c) PASCAL d) C++
10. The subprogram may be called by using a Simple_____
    a) Instruction b) Task c) Symbols d) Classes
11. MATLAB stands for _____
    a) Maths Laboratory c) Mathematical Lab
    b) Matrix Laboratory d) Maths Lab
12. MATLAB was developed by _____
    a) MathsWorks b) Intel c) Microsoft d) IBM

13. In MATLAB the matrix is defined as an _____
    a) vector b) scalar c) array d) integer
14. _____ acts as an outstanding tool for visulaizing technical data
    a) C b) C++ c) Java d) MATLAB
15. The fundamental unit of data in any MATLAB program is the _____
    a) array b) vector c) scalar d) none
16. In command window the _____ are entered
    a) data b) values c) commands d) fields
17. _____ window displays plots and graphs
    a) command c) Figure
    b) Edit d) Command history
18. _____ window permits a user to create and modify MATLAB programs
    a) command c) Figure
    b) Editor d) Command history
19. MATLAB programs are saved with the extensions _____
    a) .m b) .mm c) .mf d) .ml
20. The _____ window displays a list of commands that a user has entered in the command window
    a) edit c) debug
    b) figure d) command history

PART- B (3 * 2= 6 Marks)
Answer ALL the Questions

21. List the windows in Matlab environment
22. List the components of computer
23. Write in MATLAB 1. $3^2 + 5$ 2. 3^{2+5}

PART C (3 * 8 = 24 Marks)
Answer ALL the Questions

24. a. Elaborate on Components of computers.

(OR)

    b. Discuss on software hierarchy.

25. a. Explain various Matlab windows with example.

(OR)

    b. Write note on different matrix operators with example in Matlab.

26. a. Explain Variables and assignment in Matlab.

(OR)

    b. Write a program to assign the following expressions to a variable A and then to print out the value of A.
    (i) (3+4)/(5+6) (ii) (0.0000123 + 5.67×10-3) × 0.4567×10-4
    (iii) $2\Pi^2$ (iv) $\sqrt{2}$

KARPAGAM ACADEMY OF HIGHER EDUCATION
(Deemed to be University)
(Established Under Section 3 of UGC Act 1956)
COIMBATORE – 641 021

COMPUTER TECHNOLOGY

Third Semester
FIRST INTERNAL EXAMINATION - July 2018

PROGRAMMING IN MATLAB

Class & Section: II B.Sc (CT) **Duration: 2 hours**
Date & Session : **Maximum marks: 50 marks**
Subj.Code: 17CTU304B

PART- A (20 * 1= 20 Marks)
Answer ALL the Questions

1. Software designed to help the user to perform specific tasks is _____
   a) System software c) Operating Systems
   b) Application software d) Utility programs
2. The programs written in machine language are_____
   a) Machine independent c) Machine interconnected
   b) Machine dependent d) Machine interface
3. Assembly language consists of _____
   a) Mnemonics b) Source code c) Object code d) Byte Code
4. Assemblers perform a _____
   a) One to One translation c) Many to Many translation
   b) One to Many translation d) Many to One translation
5. Tool that produces input for compilers_____
   a) Loader **c) Preprocessor**
   b) Interpreter d) Wordprocessor
6. Computer program that links and merges various object files together in order to make an executable file_____
   a) Dynamic Linker c) Loader
   b) Linker d) Static Linker
7. High Level Languages are belongs to which generation_____
   a) First generation c) Fourth generation
   b) Second generation **d) Third generation**
8. Functional languages treat procedures like_____
   a) Mathematical Functions c) Symbols
   b) English like words d) Charaters
9. An example of Functional language_____
   a) COBOL b) FORTRAN **c) PASCAL** d) C++
10. The subprogram may be called by using a Simple_____
    a) Instruction b) Task c) Symbols d) Classes
11. MATLAB stands for _____
    a) Maths Laboratory c) Mathematical Lab
    b) Matrix Laboratory d) Maths Lab
12. MATLAB was developed by _____
    a) MathsWorks b) Intel c) Microsoft d) IBM

13. In MATLAB the matrix is defined as an _____
    a) vector b) scalar **c) array** d) integer
14. _____ acts as an outstanding tool for visulaizing technical data
    a) C b) C++ c) Java **d) MATLAB**
15. The fundamental unit of data in any MATLAB program is the _____
    a) array b) vector c) scalar d) none
16. In command window the _____ are entered
    a) data b) values **c) commands** d) fields
17. _____ window displays plots and graphs
    a) command **c) Figure**
    b) Edit d) Command history
18. _____ window permits a user to create and modify MATLAB programs
    a) command c) Figure
    b) Editor d) Command history
19. MATLAB programs are saved with the extensions _____
    a) .m b) .mm c) .mf d) .ml
20. The _____ window displays a list of commands that a user has entered in the command window
    a) edit c) debug
    b) figure **d) command history**

PART- B (3 * 2= 6 Marks)
Answer ALL the Questions

21. List the windows in Matlab environment

Table 1-1: MATLAB windows

Window	Purpose
Command Window	Main window, enters variables, runs programs.
Figure Window	Contains output from graphic commands.
Editor Window	Creates and debugs script and function files.
Help Window	Provides help information.
Command History Window	Logs commands entered in the Command Window.
Workspace Window	Provides information about the variables that are used.
Current Folder Window	Shows the files in the current folder.

22. List the components of computer

Functional Units:

a. Input Unit

b. Storage Unit

c. Output Unit

d. Processing

Arithmetic Logic Unit

Control Unit

23. Write in MATLAB 1. $3^2 + 5$ 2. 3^{2+5}

A=3^2+5;

B=3^(2+5);

PART C (3 * 8 = 24 Marks)

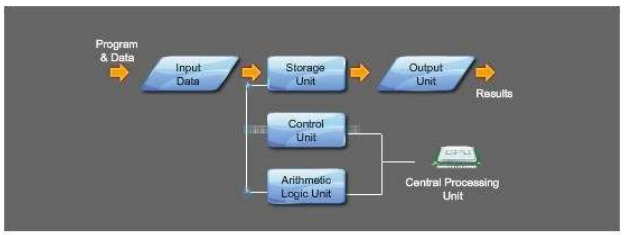
Answer ALL the Questions

24. a. Elaborate on Components of computers.

Components of Computer

A computer system consists of mainly four basic units; namely input unit, storage unit, central processing unit and output unit. Central Processing unit further includes Arithmetic logic unit and control unit, as shown in the figure:. A computer performs five major operations or functions irrespective of its size and make. These are

- it accepts data or instructions as input,
- it stores data and instruction
- it processes data as per the instructions,
- it controls all operations inside a computer,and
- it gives results in the form ofoutput.



Functional Units:

e. Input Unit: This unit is used for entering data and programs into the computer system by the user forprocessing.

f. Storage Unit: The storage unit is used for storing data and instructions before and after processing.

g. Output Unit: The output unit is used for storing the result as output produced by the computer afterprocessing.

h. Processing: The task of performing operations like arithmetic and logical operations is called processing. The Central Processing Unit (CPU) takes data and instructions from the storage unit and makes all sorts of calculations based on the instructions given and the type of data provided. It is then sent back to the storage unit. CPU includes Arithmetic logic unit (ALU) and control unit(CU)

Arithmetic Logic Unit: All calculations and comparisons, based on the instructions provided, are carried out within the ALU. It performs arithmetic functions like addition, subtraction, multiplication, division and also logical operations like greater than, less than and equal to etc.

• Control Unit: Controlling of all operations like input, processing and output are performed by control unit. It takes care of step by step processing of all operations in side the computer.

Memory

Computer's memory can be classified into two types; primary memory and secondary memory

RAM

a. Primary Memory can be further classified as **RAM andROM**.

• RAM or Random Access Memory is the unit in a computer system. It is the place in a computer where the operating system, application programs and the data in current use are kept temporarily so that they can be accessed by the computer's processor. It is said tobe 'volatile' since its contents are accessible only as long as the computer is on. The contents of RAM are no more available once the computer is turnedoff.

ROM or Read Only Memory is a special type of memory which can only be read and contents of which are not lost even when the computer is switched off. It typically contains manufacturer's instructions. Among other things, ROM also stores an initial program called the 'bootstrap loader' whose function is to start the operation of computer system once the power is turned on.

b. SecondaryMemory

RAM is volatile memory having a limited storage capacity. Secondary/auxiliary memory is storage other than the RAM. These include devices that are peripheral and are connected and controlled by the computer to enable permanent storage of programs and data.

• CD ROM

Secondary storage devices are of two types; magnetic and optical. Magnetic devices include hard disks and optical storage devices are CDs, DVDs, Pen drive, Zip drive etc.

• HardDisk

Hard disks are made up of rigid material and are usually a stack of metal disks sealed in a box. The hard disk and the hard disk drive exist together as a unit and is a permanent part of the computer where data and programs are saved. These disks have storage capacities ranging from 1GB to 80 GB and more. Hard disks are rewritable.

• CompactDisk

Compact Disk (CD) is portable disk having data storage capacity between 650-700 MB. It can hold large amount of information such as music, full-motion videos, and text etc. CDs

can be either read only or read writetype.

~~CD Drive~~

Digital VideoDisk

Digital Video Disk (DVD) is similar to a CD but has larger storage capacity and enormous clarity. Depending upon the disk type it can store several Gigabytes of data. DVDs are primarily used to store music or movies and can be played back on your television or the computer too. These are notrewritable.

Hard Disk

Input / Output Devices:

These devices are used to enter information and instructions into a computer for storage or processing and to deliver the processed data to a user. Input/Output devices are required for users to communicate with the computer. In simple terms, input devices bring information INTO the computer and output devices bring information OUT of a computer system. These input/output devices are also known as peripherals since they surround the CPU and memory of a computersystem.

Input Devices

An input device is any device that provides input to a computer. There are many input devices, but the two most common ones are a keyboard and mouse. Every key you press on the keyboard and every movement or click you make with the mouse sends a specific input signal to the computer.

Keyboard

• **Keyboard**: The keyboard is very much like a standard typewriter keyboard with a few additional keys. The basic QWERTY layout of characters is maintained to make it easy to use the system. The additional keys are included to perform certain special functions. These are known as function keys that vary in number from keyboard tokeyboard.

• **Mouse**: A device that controls the movement of the cursor or pointer on a display screen. A mouse is a small object you can roll along a hard and flat surface. Its name is derived from its shape, which looks a bit like a mouse. As you move the mouse, the pointer on the display screen moves in the samedirection.

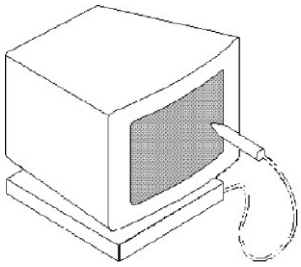
• **Trackball**: A trackball is an input device used to enter motion data into computers or other electronic devices. It serves the same purpose as a mouse, but is designed with a moveable ball on the top, which can be rolled in anydirection.

• **Touchpad**: A touch pad is a device for pointing (controlling input positioning) on a computer display screen. It is an alternative to the mouse. Originally incorporated in laptop computers, touch pads are also being made for use with desktop computers. A touch pad works by sensing the user's finger movement and downward pressure.

• **Touch Screen:** It allows the user to operate/make selections by simply touching the display screen. A display screen that is sensitive to the touch of a finger or stylus. Widely used on ATM

machines, retail point-of-sale terminals, car navigation systems, medical monitors and industrial control panels.

• **Light Pen**: Light pen is an input device that utilizes a light-sensitive detector to select objects on a display screen.



• **Magnetic ink character recognition (MICR)**: MICR can identify character printed with a special ink that contains particles of magnetic material. This device particularly finds applications in bankingindustry.

• **Optical mark recognition (OMR)**: Optical mark recognition, also called mark sense reader is a technology where an OMR device senses the presence or absence of a mark, such as pencil mark. OMR is widely used in tests such as aptitudetest.

• **Bar code reader**: Bar-code readers are photoelectric scanners that read the bar codes or vertical zebra strips marks, printed on product containers. These devices are generally used in super markets, bookshopsetc.

Scanner

Scanner is an input device that can read text or illustration printed on paper and translates the information into a form that the computer can use. A scanner works by digitizing an image.

Output Devices:

Output device receives information from the CPU and presents it to the user in the desired from. The processed data, stored in the memory of the computer is sent to the output unit, which then converts it into a form that can be understood by the user. The output is usually produced in one of the two ways – on the display device, or on paper (hardcopy).

•**Monitor**: is often used synonymously with "computer screen" or "display." Monitor is an output device that resembles the television screen (fig. 1.8). It may use a Cathode Ray Tube (CRT) to display information. The monitor is associated with a keyboard for manual input of characters and displays the information as it is keyed in. It also displays the program or application output. Like the television, monitors are also available in differentsizes.

• **Printer**: Printers are used to produce paper (commonly known as hard copy) output. Based on the technology used, they can be classified as Impact or Non-impactprinters.

Impact printers use the typewriting printing mechanism wherein a hammer strikes the paper through a ribbon in order to produce output. Dot-matrix and Character printers fall under this category.

Non-impact printers do not touch the paper while printing. They use chemical, heat or

electrical signals to etch the symbols on paper. Inkjet, Deskjet, Laser, Thermal printers fall under this category of printers.

• **Plotter**: Plotters are used to print graphical output on paper. It interprets computer commands and makes line drawings on paper using multi colored automated pens. It is capable of producing graphs, drawings, charts, maps etc. • **Facsimile (FAX)**: Facsimile machine, a device that can send or receive pictures and text over a telephone line. Fax machines work by digitizing animage.

• **Sound cards and Speaker(s)**: An expansion board that enables a computer to manipulate and output sounds. Sound cards are necessary for nearly all CD-ROMs and have become commonplace on modern personal computers. Sound cards enable the computer to output sound through speakers connected to the board, to record sound input from a microphone connected to the computer, and manipulate sound stored on adisk.

(OR)

b. Discuss on software hierarchy.

The lowest level description of a computer program is just the sequence of numbers which encode the basic CPU operations. This level is called **machine code**. Machine code is specific to a given CPU manufacturer and often specific to a given model type (for example the Pentium CPU has some codes not used by earlier 8086 CPUs). Machine code is very difficult for a human to read or write,so the lowest level of programming done by humans is in a language in which each basic operation is given a mnemonic code called **assembly language**. Humans can read and write using assembly language which can be converted into machine code using an **assembler**. Assembly language, like machine code is often specific to a particular CPU manufacturer ormodel.

The development of **high-level languages** meant that humans could program using a formalism that was closer to their conceptual models of the data being manipulated: characters, real numbers, lists, tables or database records. Such languages are easier for humans to learn and to use, and furthermore they tend to be available across different computers; with each manufacturer supplying a conversion program between the high-level language and the assembly language for their CPU. Examples of high- level languages are Fortran, Pascal, Basic, C, C++, Java andMATLAB.

Modern computer systems need to deal with complex tasks involving multiple programs interacting simultaneously, and the sharing of access to files on disks, to network resources and displays. To cope with these demands, manufacturers supply **operating systems** (e.g. Windows, Linux), which are themselves programs which help the user operate

the computer and run other **application** programs. Often individual application programs need to work together to achieve an objective: for example a word processing application might call on a drawing package or on a spreadsheet program to do some specific processing within a document. This idea of combining programs is called **scripting**, where the specifications for which programs are to be executed and how they should interact is specified in a **script**.

25. a. Explain various Matlab windows with example.

MATLAB WINDOWS

It is assumed that the software is installed on the computer, and that the user can start the program. Once the program starts, the MATLAB desktop window opens (Figure 1-1). The window contains four smaller windows: the Command Window, the Current Folder Window, the Workspace Window, and the Command History Window. This is the default view that shows four of the various windows of MATLAB. A list of several windows and their purpose is given inTable 1-1. The Start button on the lower left side can be used to access MATLAB tools and features. Four of the windows—the Command Window, the Figure Window, the Editor Window, and the Help Window—are used extensively throughout the book and are briefly described on the followingpages

Command Window: The Command Window is MATLAB's main window and opens when MATLAB is started. It is convenient to have the Command Window as the only visible window, and this can be done by either closing all the other windows (click on the x at the top right-hand side of the window you want to close) or by first selecting the Desktop Layout in the Desktop menu, and then 6 Chapter 1: Starting with MATLAB selecting Command Window Only from the submenu that opens

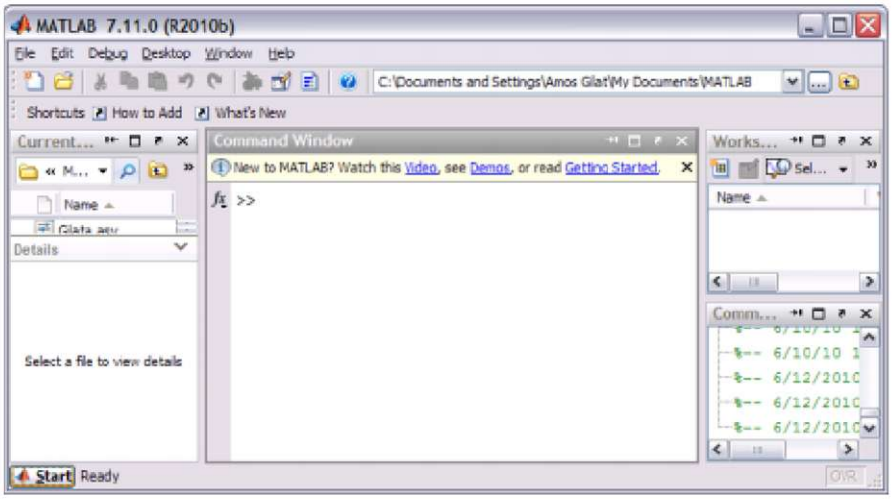


Figure 1-1: The default view of MATLAB desktop.

Table 1-1: MATLAB windows

Window	Purpose
Command Window	Main window, enters variables, runs programs.
Figure Window	Contains output from graphic commands.
Editor Window	Creates and debugs script and function files.
Help Window	Provides help information.
Command History Window	Logs commands entered in the Command Window.
Workspace Window	Provides information about the variables that are used.
Current Folder Window	Shows the files in the current folder.

Figure Window: The Figure Window opens automatically when graphics commands are executed, and contains graphs created by these commands. An example of a Figure Window is shown in Figure 1-2.

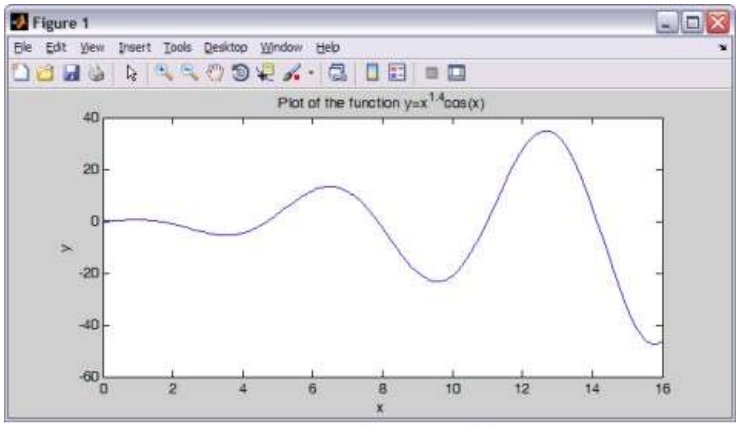


Figure 1-2: Example of a Figure Window.

Editor Window: The Editor Window is used for writing and editing programs. This window is opened from the File menu. An example of an Editor Window is shown in

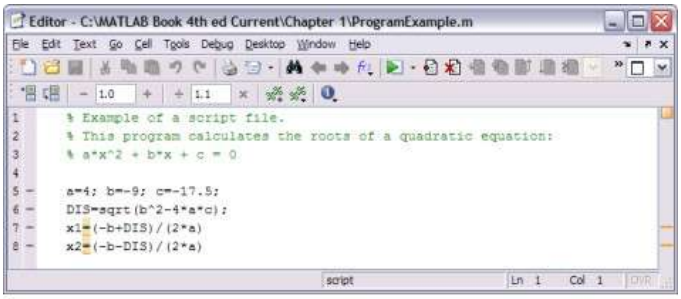


Figure 1-3: Example of an Editor Window.

Figure 1-3.

Help Window: The Help Window contains help information. This window can be opened from the Help menu in the toolbar of any MATLAB window. The Help

Window is interactive and can be used to obtain information on any feature of MATLAB. Figure 1-4 shows an open Help Window.

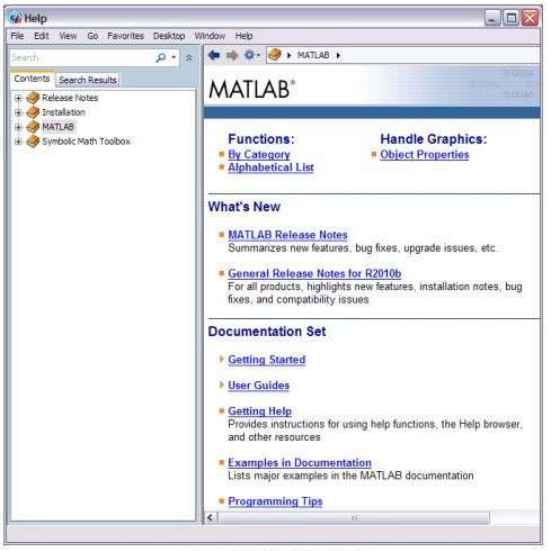


Figure 1-4: The Help Window.

Working In The Command Window The Command Window is MATLAB's main window and can be used for executing commands, opening other windows, running programs written by the user, and managing the software. An example of the Command Window, with several simple commands that will be explained later in

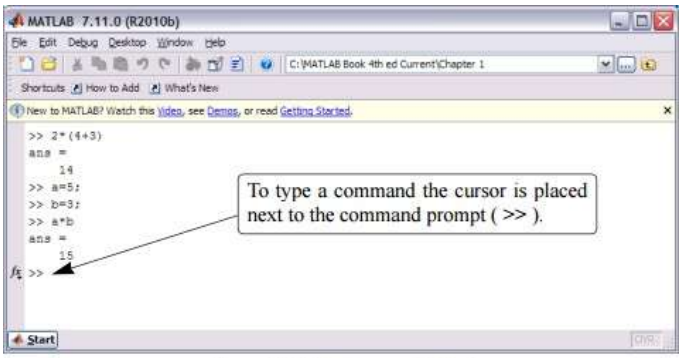


Figure 1-5: The Command Window.

(OR)

b. Write note on different matrix operators with example in Matlab.

Matrix Operators

Expressions use familiar arithmetic operators and precedence rules.

+	Addition
-	Subtraction
*	Multiplication
/	Division
\	Left division
^	Power
'	Complex conjugate transpose
( )	Specify evaluation order

26. a. Explain Variables and assignment in Matlab.

Variables and assignment

Variables are named locations in memory where numbers, strings and other elements of data maybestoredwhiletheprogramisworking.Variablenamesarecombinationsoflettersand digits, but must start with a latter. MATLAB does not require you to declare the names of variables in advance of their use. This is actually a common cause of error, since it allows you to refer accidentally to variables that don't exist. To assign a variable a value, use the **assignment statement**. This takes the form

variable=expression; forexample

```
a=6;
```

or

```
name='Mark';
```

To display the contents of a variable, use

disp(*variable*); Please note that−

- Once a variable is entered into the system, you can refer to itlater.
- Variables must have values before they areused.

- When an expression returns a result that is not assigned to any variable, the system assigns it to a variable named ans, which can be usedlater.

For example,

```
sqrt(78)
```

MATLAB will execute the above statement and return the following result −

```
ans = 8.8318
```

You can use this variable **ans** −

```
sqrt(78);
9876/ans
```

MATLAB will execute the above statement and return the following result −

```
ans = 1118.2
```

Let's look at another example −

```
x = 7 * 8;
y = x * 7.89
```

MATLAB will execute the above statement and return the following result −

```
y = 441.84
```

Multiple Assignments

You can have multiple assignments on the same line. For example,

MATLAB will execute the above statement and return the following result −

```
c = 14
```

(OR)

b. Write a program to assign the following expressions to a variable A and then to print out the value of A.

(i) (3+4)/(5+6) (ii) (0.0000123 + 5.67×10-3) × 0.4567×10-4

(iii) $2\Pi^2$ (iv) $\sqrt{2}$

A=(3+4)/(5+6); disp(A); A=2*3.14^2; disp(A); A=sqrt(2); disp(A);
A=(0.0000123+5.67*10-3)*0.4567*10-4; disp(A);

OUTPUT: 0.6363636 19.7192 1.4142136 241.24796

KARPAGAM ACADEMY OF HIGHER EDUCATION
(Deemed to be University)
(Established Under Section 3 of UGC Act 1956)
COIMBATORE – 641 021

COMPUTER TECHNOLOGY

Third Semester
SECOND INTERNAL EXAMINATION - August 2018

PROGRAMMING IN MATLAB

Class & Section: II B.Sc (CT) **Duration: 2 hours**
Date & Session : 16.8.18 AN **Maximum: 50 marks**
Subj.Code: 17CTU304B

PART- A (20 * 1= 20 Marks)
Answer ALL the Questions

1. The _____ operator swaps the row and columns of any array that is given
   a. transpose b. concatenates c. colon d. semicolon
2. The _____ function can be used to create an all zero array
   a. ones **b. zero** c. eye d. randn
3. The _____ function can be used to generate arrays containing all ones
   a. ones b. zero c. eye d. randn
4. The eye function can be used to generate arrays coantining _____ matrices
   a. square b. null **c. identity** d. none
5. MATLAB always allocated array elements in _____ major order
   a. row **b. column** c. row & column d. none
6. The _____ functions returns the highest value taken on by that subscript
   a. ones b. zero **c. end** d. repalce
7. The _____ command clears the screen
   a. clc b. clr c. cls d. cle
8. The _____ function concatenates a list of arrays along a specified dimension
   a. join **b. cat** c. rand d. joined
9. The _____ function gives the minimum value in row/column vector
   a. min b. least c. max d. minum
10. The _____ function gives the maximum value in row/column vector
    a. min b. least **c. max** d. minum
11. _____ functions concatenates two or more strings ignoring trailing blanks
    a. strrev b. strvcat **c. strcat** d. strcon
12. _____ function determines if two strings are identical
    a. strrev **b. strcmp** c. strncmp d. stricmp
13. _____ function determines if the first n characters of two strings are identical
    a. strncmp b. strcmp c. strcmpi d. stricmp
14. _____ determines if the first n characters of two strings are identical ignoring cases
    a. strncmp b. strcmp **c. strncmpi** d. stricmp
15. _____ function determines if a character is a letter
    a. isalpha **b. isletter** c. ischar d. isstring
16. A _____ plot is a plot in which each data value is represented by a marker and a line connecting the marker vertically to the x axis
    a. stair **b. stem** c. bar d. pie

17. When the _____ function is executed, MATLAB opens the Figure window and displays the plot in that window
    a. edit b. figure c. plotting **d. plot**
18. Functions receive input data from the program that invokes them through a list of variables called an _____ argument list
    a. input b. output c. result d. function
19. _____ are just collections of MATLAB statements that are stored in a file
    a. function files **b. script files** c. legal files d. none
20. A MATLAB function is a special type of _____ that runs in its own independent workspace
    a. G file **b. M file** c. MM file d. MX file

PART- B (3 * 2= 6 Marks)
Answer ALL the Questions

21. Draw a graph that joins the points (0,1), (4,3), (2,0) and (5,-2).
22. What are the color string and line styles used in Plotting?
23. List the Array operators?

PART C (3 * 8 = 24 Marks)
Answer ALL the Questions

24. a. Write any 5 Array functions with example and output

(OR)

    b. How to generate i) Magic Matrix ii) Random numbers iii) identity matrix iv) Transpose a matrix.
25. a. Describe in detail about formatted console input-output statements.

(OR)

    b. Explain in detail about Plotting with Example
26. a. Explain in details about any 10 string handling functions with example?

(OR)

    b. Write a program to generate wave forms, sound and replay

KARPAGAM ACADEMY OF HIGHER EDUCATION
(Deemed to be University)
(Established Under Section 3 of UGC Act 1956)
COIMBATORE – 641 021

COMPUTER TECHNOLOGY

Third Semester
SECOND INTERNAL EXAMINATION - August 2018

PROGRAMMING IN MATLAB

Class & Section: II B.Sc (CT) **Duration: 2 hours**
Date & Session : 16.8.18 AN **Maximum: 50 marks**
Subj.Code: 17CTU304B

PART- A (20 * 1= 20 Marks)
Answer ALL the Questions

1. The _____ operator swaps the row and columns of any array that is given
   a. transpose b. concatenates c. colon d. semicolon
2. The _____ function can be used to create an all zero array
   a. ones **b. zero** c. eye d. randn
3. The _____ function can be used to generate arrays containing all ones
   a. ones b. zero c. eye d. randn
4. The eye function can be used to generate arrays coantining _____ matrices
   a. square b. null **c. identity** d. none
5. MATLAB always allocated array elements in _____ major order
   a. row **b. column** c. row & column d. none
6. The _____ functions returns the highest value taken on by that subscript
   a. ones b. zero **c. end** d. repalce
7. The _____ command clears the screen
   a. clc b. clr c. cls d. cle
8. The _____ function concatenates a list of arrays along a specified dimension
   a. join **b. cat** c. rand d. joined
9. The _____ function gives the minimum value in row/column vector
   a. min b. least c. max d. minum
10. The _____ function gives the maximum value in row/column vector
    a. min b. least **c. max** d. minum
11. _____ functions concatenates two or more strings ignoring trailing blanks
    a. strrev b. strvcat **c. strcat** d. strcon
12. _____ function determines if two strings are identical
    a. strrev **b. strcmp** c. strncmp d. stricmp
13. _____ function determines if the first n characters of two strings are identical
    a. strncmp b. strcmp c. strcmpi d. stricmp
14. _____ determines if the first n characters of two strings are identical ignoring cases
    a. strncmp b. strcmp **c. strncmpi** d. stricmp
15. _____ function determines if a character is a letter
    a. isalpha **b. isletter** c. ischar d. isstring
16. A _____ plot is a plot in which each data value is represented by a marker and a line connecting the marker vertically to the x axis
    a. stair **b. stem** c. bar d. pie

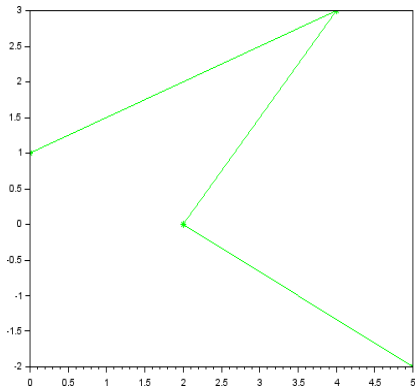
17. When the _____ function is executed, MATLAB opens the Figure window and displays the plot in that window
    a. edit b. figure c. plotting **d. plot**
18. Functions receive input data from the program that invokes them through a list of variables called an _____ argument list
    a. input b. output c. result d. function
19. _____ are just collections of MATLAB statements that are stored in a file
    a. function files **b. script files** c. legal files d. none
20. A MATLAB function is a special type of _____ that runs in its own independent workspace
    a. G file **b. M file** c. MM file d. MX file

PART- B (3 * 2= 6 Marks)
Answer ALL the Questions

21. Draw a graph that joins the points (0,1), (4,3), (2,0) and (5,-2).

Answer

```
X=[0 4 2 5];
Y=[1 3 0 -2];
plot(X,Y,'*-');
```



22. What are the color string and line styles used in Plotting?

Answer

where style is made up from characters as follows:

- Color strings are 'c', 'm', 'y', 'r', 'g', 'b', 'w', and 'k'. These correspond to cyan, magenta, yellow, red, green, blue, white, and black.
- Linestyle strings are '-' for solid, '--' for dashed, ':' for dotted, '-.' for dash-dot, and none for no line.
- The marker types are '+', 'o', '*', and 'x' and the filled marker types 's' for square, 'd' for diamond, '^' for up triangle, 'v' for down triangle, '>' for right triangle, '<' for left triangle, 'p' for pentagram, 'h' for hexagram, and none for no marker.

For example:

```
x = [ 1 2 3 4 ];
y = [ 10 15 20 25 ];
plot(x,y,'g-*');
```

23. List the Array operators?

Arrays

MATLAB is particularly powerful in the way it deals with tables of data, called arrays. An array is simply a variable that can contain a number of values arranged in tabular form. Arrays may be one dimensional (like a list), two dimensional (like a table), or have more dimensions. To set the value of one element of a one dimensional array, use the notation

```
variable(index)=expression;
```

for example

```
table(1)=3;
table(2)=6;
```

PART C (3 * 8 = 24 Marks)
Answer ALL the Questions

24. a. Write any 5 Array functions with example and output

Answer

Arrays

MATLAB is particularly powerful in the way it deals with tables of data, called arrays. An array is simply a variable that can contain a number of values arranged in tabular form. Arrays may be one dimensional (like a list), two dimensional (like a table), or have more dimensions. To set the value of one element of a one dimensional array, use the notation

```
variable(index)=expression;
```

for example

```
table(1)=3;
table(2)=6;
```

Note that indexes must be expressions evaluating to positive integers. The smallest index is 1. To access one element from a one dimensional array, use the notation

```
variable(index)
```

for example

```
a=table(2);
disp(table(2));
```

For two dimensional arrays, use

```
variable(index,index)=expression;
```

to set the value and

```
variable(index,index)
```

to retrieve its value. You can store strings in tables, but each string occupies a row, and all rows must be the same length (think of a two-dimensional array of characters).

You can assign a whole array in one operation using a notation involving square brackets: for example:

```
array = [ v11 v12 v13; v21 v22 v23];
```

where v11 is the value in row 1 col 1; v21 is the value in row 2 col 1; etc. The ';' marks the end of a row.

You can generate arrays containing sequences very easily with the ':' operator. The expression

```
start:stop
```

generates a sequence of integers from start to stop. The expression

```
start:increment:stop
```

generates a sequence from start to stop with the specfied increment. Try

```
disp(1:10);
disp(1:2:10);
```

You can also select sub-parts of the array with the ':' operator. For example,

```
x(3:5)
```

represents the array consisting of the third through fifth elements of x. Also

```
y(2:2:100)
```

represents the array containing the even number elements of y below index 100.

You can also add subtract, multiply and divide arrays of data using the operators we've mentioned previously. However MATLAB makes a difference between operations that work on a cell-by-cell basis (so-called "dot" operations) as opposed to operations that work on the arrays as a whole. For example, if you want to multiply two arrays of equal size to give a third array in which each cell contains the product of the corresponding cells in the input, then you need to use the "dot-multiply" operator .* for example

```
C = A.*B;
```

Finally you can transpose rows and columns of a matrix with the ' operator, for example

```
disp(A')
```

(OR)

b. How to generate i) Magic Matrix ii) Random numbers iii) identity matrix iv) Transpose a matrix.

Answer

i) Magic Matrix

An n-by-n magic square is an array containing the integers from 1 to n 2arranged so that each of the rows, each of the columns, and the two principal diagonals have the same sum. For each n > 2, there are many different magic squares of order n, but the Matlab function magic(n) generates a particular one. Matlab can generate Lo Shu with

A = magic(3)

which produces

A =

8 1 6

3 5 7

4 9 2

25. a. Describe in detail about formatted console input-output statements.

Answer

Formatted console input-output

You can control the exact way in which values are printed to the screen with the 'fprintf()' function (fprintf= "file print formatted"). This function takes one argument

repesenting the formatting instructions, followed by a list of values to be printed. Embedded within the format string are 'percent commands' which control where and how the values are to be written. Here are some examples:

```
fprintf('The answer is %g seconds.\n',nsec);
fprintf('Day of the week = %s\n',dayofweek([7 12 1941]));
fprintf('Mean=%.3f ± %.4f\n',mean,stddev);
```

The command %g represents a general real number, %f means a fixed point number, %d a decimal integer, and %s a string. You can put numeric values between the '%' and the letter to control the field width and the number of digits after the decimal point. For example (□=space):

fprintf('%5g',10)	□□□10
fprintf('%10.4f',123.456)	□□123.4560
fprintf('%10s', 'fred')	□□□□□□fred

You can input a value or a string from the command line with the 'input()' function. This has two forms depending on whether you want to input a number or a string:

```
yval=input('Enter a number: ');
name=input('Enter your name: ', 's');
```

Input and Output Commands

MATLAB provides the following input and output related commands −

Command	Purpose
disp	Displays contents of an array or string.
fscanf	Read formatted data from a file.
format	Controls screen-display format.
fprintf	Performs formatted writes to screen or file.
input	Displays prompts and waits for input.
;	Suppresses screen printing.

The **fscanf** and **fprintf** commands behave like C scanf and printf functions. They support the following format codes

(OR)

b. Explain in detail about Plotting with Example

Answer

Basic Plotting

To create XY graphs, it is easiest to form your data into two row vectors, one for the x co-ordinates, and one for the y co-ordinates. The command

```
plot(x,y)
```

will then create a figure with points at each y value for each matching x value. You can control the style of any line drawn through the points by a third string argument to the plot command:

```
plot(x,y,style);
```

where style is made up from characters as follows:

- Color strings are 'c', 'm', 'y', 'r', 'g', 'b', 'w', and 'k'. These correspond to cyan, magenta, yellow, red, green, blue, white, and black.
- Linestyle strings are '-' for solid, '--' for dashed, ':' for dotted, '-.' for dash-dot, and none for no line.
- The marker types are '+', 'o', '*', and 'x' and the filled marker types 's' for square, 'd' for diamond, '^' for up triangle, 'v' for down triangle, '>' for right triangle, '<' for left triangle, 'p' for pentagram, 'h' for hexagram, and none for no marker.

For example:

```
x = [ 1 2 3 4 ];
y = [ 10 15 20 25 ];
plot(x,y,'g-*');
```

You can plot multiple lines by repeating the arguments:

```
plot(x1,y1,x2,y2,…);
```

or

```
plot(x1,y1,style1,x2,y2,style2,…);
```

You can give the graph a title with the

```
title(label);
```

command, where label is a character string. Likewise you can add labels to the X and Y axes with

```
xlabel(label);
```

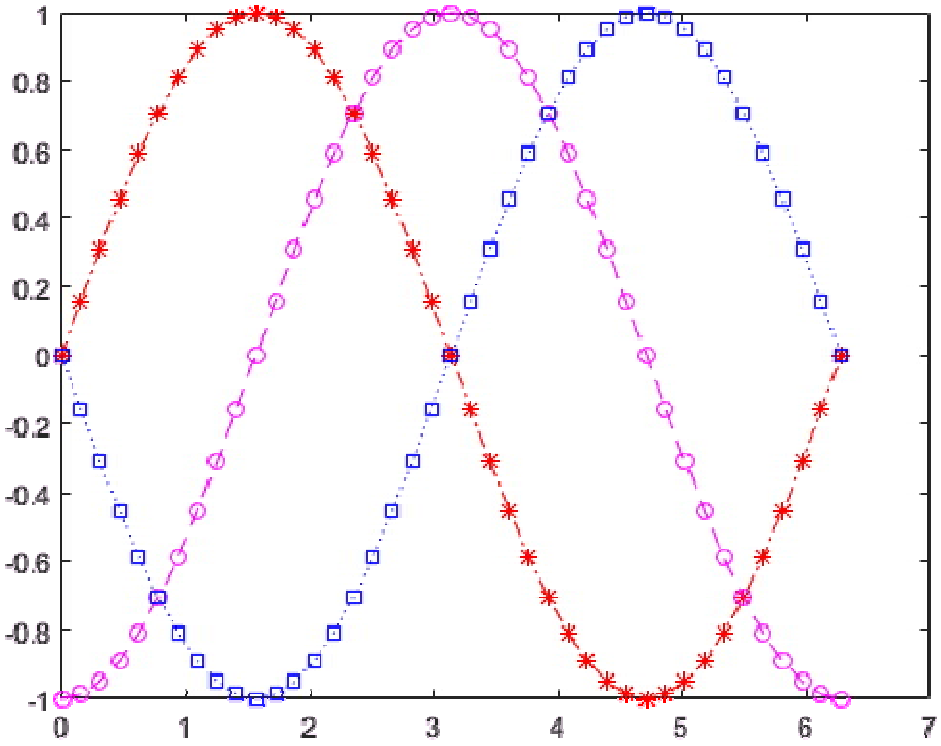
and

```
ylabel(label);
```

You can add a legend with

```
legend(label1,label2,label3,…);
```

```
figure
t = 0:pi/20:2*pi;
plot(t,sin(t),'-.r*')
hold on
plot(t,sin(t-pi/2),'--mo')
plot(t,sin(t-pi),':bs')
hold off
```



26. a. Explain in details about any 10 string handling functions with example?

Answer

String handling

Simple strings are stored as tables with one row and a number of columns: one column per character. You can concatenate any table or strings simply by making the contents part of one table. For example:

```
str1='Hello';
str2='Mark ';
str=[str1 ' ' str2];
```

You can convert numbers to strings using the 'sprintf()' function, which operates analogously to the fprintf() function but outputs to a string rather than to the screen.

```
str=sprintf('%10.4f',123.45);
```

The 'abs()' function can be used to find the standard character codes for a string:

```
disp(abs('Mark'));
   77    97   114   107
```

The 'char()' function can be used to convert character codes back to a string:

```
disp(char([77 97 114 107]));
Mark
```

The 'eval()' function can be used to evaluate an expression stored in a string. This allows you to execute expressions typed in by the user:

```
expr=input('Enter an expression (e.g. "2+3*4") : ', 's');
disp(eval(expr));
```

Function	Purpose
Functions for storing text in character arrays, combine character arrays, etc.	
blanks	Create string of blank characters
cellstr	Create cell array of strings from character array
char	Convert to character array (string)
iscellstr	Determine whether input is cell array of strings
ischar	Determine whether item is character array
sprintf	Format data into string
strcat	Concatenate strings horizontally
strjoin	Join strings in cell array into single string
Functions for identifying parts of strings, find and replace substrings	
ischar	Determine whether item is character array
isletter	Array elements that are alphabetic letters
isspace	Array elements that are space characters
isstrprop	Determine whether string is of specified category
sscanf	Read formatted data from string

strfind	Find one string within another
strrep	Find and replace substring

(OR)

b. Write a program to generate wave forms, sound and replay

Generating waveforms

Waveforms are just long vectors with one number per amplitude sample. Usually they are best kept scaled so that each amplitude is between –1 and 1. To generate a sinewave, first generate a time sequence t representing the times of each sampling instant; for example:

```
t = 0:0.0001:2;
```

would generate a two second sequence with a sampling interval of 0.1ms (i.e. 10,000Hz). You can then generate a sinewave at frequency F with the expression

```
y = sin(2*pi*F*t);
```

You can create a pulse by creating a vector of zeros and setting a single element to one. A pulse train has a series of elements set to one. If these occurred every 100 elements, you might use the expression

```
y(1:100:10000)=1;
```

To create a simple sawtooth, you can use the remainder function, for example

```
y = rem(1:10000,100)/100;
```

To create a noise waveform, you can use the 'rand(*nrows*,*ncols*)' function, for example

```
y = rand(1,10000);
```

Sound Replay, Load and Save

To replay a waveform, you can use

```
sound(wave,samplerate);
```

To ensure that the waveform is scaled to the range –1 .. +1 before replay, use

```
soundsc(wave,samplerate);
```

instead.

To save a waveform to a file, use

```
save filename variable;
```

To load a waveform from a file, use

```
load filename variable;
```

To save a waveform in a Windows compatible audio file format, use

```
wavwrite(waveform,samplerate,filename);
```

To load a Windows compatible audio file, use

```
[waveform,samplerate,nbits]=wavread(filename);
```

KARPAGAM ACADEMY OF HIGHER EDUCATION
(Deemed to be University)
(Established Under Section 3 of UGC Act 1956)
COIMBATORE – 641 021

COMPUTER TECHNOLOGY

Third Semester
THIRD INTERNAL EXAMINATION - October 2018

PROGRAMMING IN MATLAB

Class & Section: II B.Sc (CT) **Duration: 2 hours**
Date & Session : **Maximum: 50 marks**
Subj.Code: 17CTU304B

PART- A (20 * 1= 20 Marks)
Answer ALL the Questions

1. The _____ function converts logical data to numerical data
   a. real b. logical c. relation d. array
2. The _____ operators are operators with two numerical or string operands that yield a logical result
   a. logical c. bitwise
   b. relational d. arithmetic
3. The relational operators can compare two strings only if they are of _____ length
   a. equal c. both equal and different
   b. different d. equ
4. _____ are MATLAB statements that permit us to select and execute specific sections of code whiel skipping other sections of code
   a. looping **b. branches** c. structures d. none
5. The properties of any object can be examined at any time using the _____ function
   a. set b. plot **c. get** d. end
6. The properties of any object can be modified at any time using the _____ function
   a. set b. plot c. get d. end
7. The _____ function can be used to provide lists of possible property values
   a. set b. plot c. get d. end
8. the function _____ will return all of the possible choices for all the properties of an object
   a. set() b. set(hndl) c. get() **d. get(hndl)**
9. The _____ command is used to display only the subset of the data
   a. axes **b. axis** c. plot d. plotting
10. The _____ command sets the axis increments to be equal on both axes
    a. axis normal b. axis square c. axis on **d. axis equal**
11. The _____ command makes the current axis box square
    a. axis normal **b. axis square** c. axis on d. axis equal
12. The _____ command cancels the effect of axis equal and axis sqaure
    a. axis normal b. axis square c. axis on d. axis equal
13. Data values can be retrieved from the object using the _____ function
    a. getapp b. get c. appdata d. getappdata
14. _____ returns the handle of the current figure
    a. gcf b. gca c. gco d. findobj

15. _____ returns the handle of the current axes in the current figure
    a. gcf **b. gca** c. gco d. findobj
16. _____ returns the handle of the current object
    a. gcf b. gca **c. gco** d. findobj
17. Each figure is identifed by the _____
    a. window number **c. figure number**
    b. screen number d. picture number
18. The position of axes and unicontrol objects is specified by a _____ vector
    a. 2 element b. 3 element **c. 4 element** d. 5 element
19. _____ objects have a position property containing only two or three elements
    a. text b. value c. figure d. frame
20. The _____ and units properties specify the location of a figure on the computer screen
    a. position b. value c. figure d. handle

PART- B (3 * 2= 6 Marks)
Answer ALL the Questions

21. Define loop. List its types.
Answer

Loop Type	Description
while loop	Repeats a statement or group of statements while a given condition is true. It tests the condition before executing the loop body.
for loop	Executes a sequence of statements multiple times and abbreviates the code that manages the loop variable.
nested loops	You can use one or more loops inside any another loop.

22. How to use sort rows in MATALB?
Answer

```
A = 6×7

    95    45    92    41    13     1    84
    95     7    73    89    20    74    52
    95     7    73     5    19    44    20
    95     7    40    35    60    93    67
    76    61    93    81    27    46    83
    76    79    91     0    19    41     1

B = sortrows(A)
B = 6×7

    76    61    93    81    27    46    83
```

```
 76    79    91     0    19    41     1
 95     7    40    35    60    93    67
 95     7    73     5    19    44    20
 95     7    73    89    20    74    52
 95    45    92    41    13     1    84
```

Sort the rows of A based on the values in the second column. When the specified column has repeated elements, the corresponding rows maintain their original order.

23. Write a vote program using if else condition in MATALB.

Answer

```
Age=21;
If (Age>=18)
disp('Eligible');
Else
disp(Not eligible);
End
```

Output

Eligible

PART C (3 * 8 = 24 Marks)
Answer ALL the Questions

24. a. Explain Switch and nested switch statements with example?

Answer

Switch Statements

A switch block conditionally executes one set of statements from several choices. Each choice is covered by a case statement.

An evaluated switch_expression is a scalar or string.

An evaluated case_expression is a scalar, a string or a cell array of scalars or strings.

The switch block tests each case until one of the cases is true. A case is true when −

- For numbers, **eq(case_expression,switch_expression)**.
- For strings, **strcmp(case_expression,switch_expression)**.
- For objects that support the **eq(case_expression,switch_expression)**.
- For a cell array case_expression, at least one of the elements of the cell array matches switch_expression, as defined above for numbers, strings and objects.

Syntax

The syntax of switch statement in MATLAB is −

```
switch<switch_expression>
case<case_expression>
<statements>
case<case_expression>
<statements>
...
...
otherwise
<statements>
end
```

Example

Create a script file and type the following code in it −

```
grade='B';
switch(grade)
case'A'
fprintf('Excellent!\n');
case'B'
fprintf('Well done\n');
case'C'
fprintf('Well done\n');
case'D'
fprintf('You passed\n');

case'F'
fprintf('Better try again\n');

otherwise
fprintf('Invalid grade\n');
```

```
end
```

When you run the file, it displays −

```
Well done
```

Nested Switch statements

It is possible to have a switch as part of the statement sequence of an outer switch. Even if the case constants of the inner and outer switch contain common values, no conflicts will arise.

Syntax

The syntax for a nested switch statement is as follows −

```
switch(ch1)
case'A'
fprintf('This A is part of outer switch');
switch(ch2)
case'A'
fprintf('This A is part of inner switch');

case'B'
fprintf('This B is part of inner switch');
end
case'B'
fprintf('This B is part of outer switch');
end
```

Example

Create a script file and type the following code in it −

```
a =100;
b =200;
switch(a)
case100
fprintf('This is part of outer switch %d\n', a );
```

```
switch(b)
case200
fprintf('This is part of inner switch %d\n', a );
end
end
fprintf('Exact value of a is : %d\n', a );
fprintf('Exact value of b is : %d\n', b );
```

When you run the file, it displays −

```
This is part of outer switch 100
This is part of inner switch 100
Exact value of a is : 100
Exact value of b is : 200
```

(OR)

b. Explain If Else-if and nested if Statements with example?

Answer

If elseifelseif else end statements

An **if** statement can be followed by one (or more) optional **elseif...** and an **else** statement, which is very useful to test various conditions.

When using if... elseif...else statements, there are few points to keep in mind:

- An if can have zero or one else's and it must come after any elseif's.
- An if can have zero to many elseif's and they must come before the else.
- Once an else if succeeds, none of the remaining elseif's or else's will be tested.

Syntax

```
if<expression 1>
%Executeswhen the expression 1istrue
<statement(s)>

elseif<expression 2>
%Executeswhen the boolean expression 2istrue
```

```
<statement(s)>

Elseif<expression 3>
%Executeswhen the boolean expression 3istrue
<statement(s)>

else
%  executeswhen the none of the above condition istrue
<statement(s)>
end
```

Example

Create a script file and type the following code in it −

```
a =100;
%check the boolean condition
if a ==10
%if condition istruethenprint the following
fprintf('Value of a is 10\n');
elseif( a ==20)
%ifelseif condition istrue
fprintf('Value of a is 20\n');
elseif a ==30
%ifelseif condition istrue
fprintf('Value of a is 30\n');
else
%if none of the conditions istrue'
fprintf('None of the values are matching\n');
fprintf('Exact value of a is:%d\n', a );
end
```

When the above code is compiled and executed, it produces the following result −

```
None of the values are matching
Exact value of a is: 100
```

Nested If Statements

It is always legal in MATLAB to nest if-else statements which means you can use one if or elseif statement inside another if or elseif statement(s).

Syntax

The syntax for a nested if statement is as follows −

```
if<expression 1>
%Executeswhen the boolean expression 1istrue
if<expression 2>
%Executeswhen the boolean expression 2istrue
end
end
```

You can nest elseif...else in the similar way as you have nested if statement.

Example

Create a script file and type the following code in it −

```
a =100;
b =200;
% check the boolean condition
if( a ==100)

%if condition istruethen check the following
if( b ==200)

%if condition istruethenprint the following
fprintf('Value of a is 100 and b is 200\n');
end

end
```

```
fprintf('Exact value of a is : %d\n', a );
fprintf('Exact value of b is : %d\n', b );
```

When you run the file, it displays −

```
Value of a is 100 and b is 200
Exact value of a is : 100
Exact value of b is : 200
```

25. a. Write matlab program to find factorial of a given number

Answer

A Simple Example: factorial

• Some definitions:

```
function y = factorial( n )
%FACTORIAL Calculate the factorial of n
% Function FACTORIAL calculates the factorial of n:
% n!=1*2*...*(n-  1)*n
%
% Written by: Qi Ying (Jan 2011)
% CVEN 302
y=1;
for ii=1:n
y=y*ii;
end
end
```

1. The M- file is called factorial.m.
2. n is called the input argument
3. y is called the output argument
4. The first comment line is called the H1 comment line, which is searchable by the 'lookfor' command
5. The remaining comment lines until the first blank line or executable line is displayed by the 'help' command
6. Loop variable ii is a 'local' variable, only visible inside the function.

(OR)

b. Explain in detail about Files. How to read and write in files?

Answer

Writing to a text file

To save the results of some computation to a file in text format reqires the following steps:

a. Open a new file, or overwrite an old file, keeping a 'handle' for the file.
b. Print the values of expressions to the file, using the file handle
c. Close the file, using the file handle

The file handle is a just a variable which identifies the open file in your program. This allows you to have any number of files open at any one time.

```
% open file
fid = fopen('myfile.txt','wt');     % 'wt' means "write text"
if (fid < 0)
   error('could not open file "myfile.txt"');
end;
% write some stuff to file
for i=1:100
   fprintf(fid,'Number = %3d Square = %6d\n',i,i*i);
end;
% close the file
fclose(fid);
```

2. Reading from a text file

To read some results from a text file is straightforward if you just want to load the whole file into memory. This requires the following steps:

a. Open an existing file, keeping a 'handle' for the file.
b. Read expressions from the file into a single array, using the file handle
c. Close the file, using the file handle

The fscanf() function is the inverse of fprintf(). However it returns the values it reads as values in a matrix. You can control the 'shape' of the output matrix with a third argument.

```
A = fscanf(fid,"%g %g %g\n",[3,inf])       % A has 3 rows and 1 col per line
disp(A(1,1))         % display first value on first line
disp(A(1,2))         % display first value on second line
disp(A(2,1))         % display second value on first line
```

Thus to read back the data we saved above:

```
% open file
fid = fopen('myfile.txt','rt');     % 'rt' means "read text"
if (fid < 0)
   error('could not open file "myfile.txt"');
end;
% read from file into table with 2 rows and 1 column per line
tab = fscanf(fid,'Number = %d Square = %d\n',[2,inf]);
% close the file
fclose(fid);
rtab = tab';          % convert to 2 columns and 1 row per line
```

26. a. Explain in detail about GUI Interfaces with example?

Answer

What Is a UI?

A user interface (UI) is a graphical display in one or more windows containing controls, called components, that enable a user to perform interactive tasks. The user does not have to create a script or type commands at the command line to accomplish the tasks. Unlike coding programs to accomplish tasks, the user does not need to understand the

details of how the tasks are performed.

a. UI components can include menus, toolbars, push buttons, radio buttons, list boxes, and
b. sliders—just to name a few. UIs created using MATLAB®
c. tools can also perform any typeof computation, read and write data files, communicate with other UIs, and display dataas tables or as plots.

The following figure illustrates a simple UI that you can easily build yourself.
The UI contains these components:
• An axes component
• A pop-up menu listing three data sets that correspond to MATLAB functions: peaks, membrane, and sinc
What Is a UI?
1-3
• A static text component to label the pop-up menu
• Three buttons that provide different kinds of plots: surface, mesh, and contour
When you click a push button, the axes component displays the selected data set using the specified type of 3-D plot.
Goal Description of Approach Create a dialog box Call a function that creates predefined dialog box. For more information, see "Predefined Dialog Boxes". Create a UI containing a few components It is often simpler to create UIs that contain only a few components

(OR)

b. Explain in detail about Randomizing and sorting a list with example
Answer

Randomising and sorting a list

Assuming we have a table of values, how can we randomise the order of the entries? A good way of achieving this is analogous to shuffling a pack of cards. We pick two positions in the pack, then swap over the cards at those two positions. We then just repeat this process enough times that each card is likely to be swapped at least once.

```
functionrtab=randomise(tab)
% randomise the order of the rows in tab.
% columns are unaffected
[nrows,ncols]=size(tab);          % get size of input matrix
cnt = 10*nrows;                   % enough times
while (cnt> 0)
   pos1 = 1+fix(nrows*rand);      % get first random row
   pos2 = 1+fix(nrows*rand);      % get second random row
   tmp = tab(pos1,:);             % save first row
   tab(pos1,:) = tab(pos2,:);     % swap second into first
   tab(pos2,:) = tmp;             % move first into second
   cnt=cnt-1;
end;
rtab=tab;                         % return randomised table
return;
```

Here is a sort function that might be used with this comparison function.

```
function stab=functionsortrows(tab,funcname)
% sorts the rows of the input table using the supplied
% function name to provide an ordering on pairs of rows
[nrows,ncols]=size(tab);
for i=2:nrows                          % sort each row into place
   j = i;
   tmp = tab(j,:);                     % save row
   % compare this row with higher rows to see where it goes
   while ((j > 1)&(feval(funcname,tmp,tab(j-1,:))<0))
      tab(j,:) = tab(j-1,:);           % shift higher rows down
      j = j - 1;
   end;
   tab(j,:) = tmp;                     % put in ordered place
end;
stab = tab;                            % return sorted table
return;
```

KARPAGAM ACADEMY OF HIGHER EDUCATION
(Deemed to be University)
(Established Under Section 3 of UGC Act 1956)
COIMBATORE – 641 021

COMPUTER TECHNOLOGY

Third Semester
THIRD INTERNAL EXAMINATION - October 2018

PROGRAMMING IN MATLAB

Class & Section: II B.Sc (CT) **Duration: 2 hours**
Date & Session : **Maximum: 50 marks**
Subj.Code: 17CTU304B

PART- A (20 * 1= 20 Marks)
Answer ALL the Questions

1. The _____ function converts logical data to numerical data
   a. real b. logical c. relation d. array
2. The _____ operators are operators with two numerical or string operands that yield a logical result
   a. logical c. bitwise
   b. relational d. arithmetic
3. The relational operators can compare two strings only if they are of _____ length
   a. equal c. both equal and different
   b. different d. equ
4. _____ are MATLAB statements that permit us to select and execute specific sections of code whiel skipping other sections of code
   a. looping **b. branches** c. structures d. none
5. The properties of any object can be examined at any time using the _____ function
   a. set b. plot **c. get** d. end
6. The properties of any object can be modified at any time using the _____ function
   a. set b. plot c. get d. end
7. The _____ function can be used to provide lists of possible property values
   a. set b. plot c. get d. end
8. the function _____ will return all of the possible choices for all the properties of an object
   a. set() b. set(hndl) c. get() **d. get(hndl)**
9. The _____ command is used to display only the subset of the data
   a. axes **b. axis** c. plot d. plotting
10. The _____ command sets the axis increments to be equal on both axes
    a. axis normal b. axis square c. axis on **d. axis equal**
11. The _____ command makes the current axis box square
    a. axis normal **b. axis square** c. axis on d. axis equal
12. The _____ command cancels the effect of axis equal and axis sqaure
    a. axis normal b. axis square c. axis on d. axis equal
13. Data values can be retrieved from the object using the _____ function
    a. getapp b. get c. appdata d. getappdata
14. _____ returns the handle of the current figure
    a. gcf b. gca c. gco d. findobj

15. _____ returns the handle of the current axes in the current figure
    a. gcf **b. gca** c. gco d. findobj
16. _____ returns the handle of the current object
    a. gcf b. gca **c. gco** d. findobj
17. Each figure is identifed by the _____
    a. window number **c. figure number**
    b. screen number d. picture number
18. The position of axes and unicontrol objects is specified by a _____ vector
    a. 2 element b. 3 element **c. 4 element** d. 5 element
19. _____ objects have a position property containing only two or three elements
    a. text b. value c. figure d. frame
20. The _____ and units properties specify the location of a figure on the computer screen
    a. position b. value c. figure d. handle

PART- B (3 * 2= 6 Marks)
Answer ALL the Questions

21. Define loop. List its types.
22. How to use sort rows in MATALB?
23. Write a vote program using if else condition in MATALB.

PART C (3 * 8 = 24 Marks)
Answer ALL the Questions

24. a. Explain Switch and nested switch statements with example?

(OR)

    b. Explain If Else-if and nested if Statements with example?
25. a. Write matlab program to find factorial of a given number

(OR)

    b. Explain in detail about Files. How to read and write in files?
26. a. Explain in detail about GUI Interfaces with example?

(OR)

    b. Explain in detail about Randomizing and sorting a list with example