COMPUTATION PHYSICS

PHYS 4361
DR. Maher O. El-Ghossain
Associate Professor
SYLLABUS

TEXT BOOKS: MATLAB, An Introduction with applications, AMOS GILAT

Coverage:
1. Introduction To Computing
2. Introduction to FORTRAN
3. And Introduction to MatLaB
4. Creating ARRAYS and Mathematical Operation with ARRAYS
5. PROGRAMING WITH MATLAB
6-APPLICATIONS IN NUMERICAL ANALYSES

7- PLOTS, Polynomials, Curve Fitting, and Interpolation

8- FUNCTIONS

GRADE: M.EXAM 30%, 10% HW and Activities in class

SF. Exam 60%.
CHAPTER 1

INTRODUCTION TO COMPUTING

- Computer: Input – Processing-Output – Storage
- Computer System: Computer – All the Machines connected to the computer, execute program.
- General Processing Unit (CPU): The heart of any computer system convert input to output.
- Data and Programs: Data any information read by a computer, programs are instruction tells the computer how to process information which written in programming language.
- Computer Storage: Primary Storage called Memory, Secondary Storage (separate Device).
The CPU and its Memory: Consists of Arithmetic Logic Unit (ALU) and control unit.

- Hardware and Software: Machines and Programs, users are people using the computer, programmers are people write programs.
- Microcomputer: Small computer system called personal computer PCS.
- Desktop Unit: Desktop computers like IBM PC, Apple, Macintosh.
- Labtop Units: Light weight computer (8 lb)
- Super Computers: High speed, accuracy, high quality computer
• **BINARY System**: 0 or 1, on or off,
  
  **B Byte**: Single Character of Data,
  
  **KB, MB, GB, TB**

**RAM**: Random Access Memory, which work when the computer is on, data lost when computer off.

**ROM**: Read-Only Memory, non-erasable programs.

**The Operating System**: Collective of Program Manage the computer system activities, PC-Dos, Unix, Vax/Vms, Windows system software.
Numbering: Decimal 0, 1, ..., 9. Base 10
7216 = 7 x 10^3 + 2 x 10^2 + 1 x 10^1 + 6 x 10^0

Binary: 0 or 1. Base 2
(11100)_2 = 1 x 2^4 + 1 x 2^3 + 1 x 2^2 + 0 x 2^1 + 0 x 2^0 = (28)_{10}
(1011001)_2 = (89)_{10}

= 1 x 2^0 + 0 x 2^1 + 0 x 2^2 + 1 x 2^3 + 1 x 2^4 + 0 x 2^5 + 1 x 2^6

Decimal To Binary:
\[
\begin{array}{ccccccccc}
89/2 & 44/2 & 22/2 & 11/2 & 5/2 & 2/2 & 2/1 \\
1 & 0 & 0 & 1 & 1 & 0 & 1
\end{array}
\]
• Hexadecimal: Base 16
  - 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F.
  - A = 10, B = 11, C = 12, D = 13, E = 14, F = 15.
  - Hexadecimal number: 4F6A
  - $= 4 \times 16^3 + 15 \times 16^2 + 6 \times 16^1 + 10 \times 16^0$
  - $= 20330$

Some Binary Number:

- 0 0000 1 0001 2 0010
- 3 0011 4 0100 5 0101
- 6 0110 7 0111 8 1000
- 9 1001 10 1010
Computers

- Laptop Unit
- Desktop Unit
Scientific programming languages
What is the best high-level language to use for scientific programming? This, unfortunately, is a highly contentious question. Over the years, literally hundreds of high-level languages have been developed. However, few have stood the test of time. Many languages) e.g. (Algol, Pascal, Haskell) can be dismissed as ephemeral computer science fads. Others) e.g. (Cobol, Lisp, Ada) are too specialized to adapt for scientific use. Let us examine the remaining options
FORTRAN was the first high-level programming language to be developed: in fact, it predates the languages listed below by decades. Before the advent of FORTRAN, all programming was done in assembler code! Moreover, FORTRAN was specifically designed for scientific computing. Indeed, in the early days of computers all computing was scientific in nature--i.e. physicists and mathematicians were the original computer scientists! FORTRAN's main advantages are that it is very straightforward, and it interfaces well with most commonly available, pre-written subroutine libraries (since these libraries generally consist of compiled FORTRAN code). FORTRAN's main disadvantages are all associated with its relative antiquity. For instance, FORTRAN's control statements are fairly rudimentary, whereas its input/output facilities are positively paleolithic.
This language is a major extension to FORTRAN 77 which does away with many of the latter language's objectionable features. In addition, many ``modern'' features, such as dynamic memory allocation, are included in the language for the first time. The major disadvantage of this language is the absence of an inexpensive compiler. There seems little prospect of this situation changing in the near future.
This language was originally developed by computer scientists to write operating systems. Indeed, all UNIX operating systems are written in C. C is, consequently, an extremely flexible and powerful language. Amongst its major advantages are its good control statements and excellent input/output facilities. C's main disadvantage is that, since it was not specifically written to be a scientific language, some important scientific features (e.g. complex arithmetic) are missing. Although C is a high-level language, it incorporates many comparatively low-level features, such as pointers (this is hardly surprisingly, since C was originally designed to write operating systems). The low-level features of C—in particular, the rather primitive implementation of arrays—sometimes make scientific programming more complicated than need be the case, and undoubtedly facilitate programming errors. On the other hand, these features allow scientific programmers to write extremely efficient code. Since efficiency is generally the most important concern in scientific computing, the low-level features of C are, on balance, advantageous.
C++

This language is a major extension of C whose main aim is to facilitate object-orientated programming. Object-orientation is a completely different approach to programming than the more traditional procedural approach: it is particularly well suited to large projects involving many people who are each writing different segments of the same code. However, object-orientation represents a large, and somewhat unnecessary, overhead for the type of straightforward, single person programming tasks considered in this course. Note, however, that C++ incorporates some non-object-orientated extensions to C which are extremely useful.
Other Programs

1. MTHCAD
2. MAPLE
3. MATHEMATICA
4. MATLAB
5. TRUE BASIC
Of the above languages, we can immediately rule out C++, because object-orientation is an unnecessary complication (at least, for our purposes), and FORTRAN 90, because of the absence of an inexpensive compiler. The remaining options are FORTRAN 77 and C. I have chosen to use C (augmented by some of the useful, non-object-orientated features of C++) in this course, simply because I find the archaic features of FORTRAN 77 too embarrassing to teach students in the 21st century.