# SEMESTER - 5
# Comprehensive Lesson Plan: Computational Physics (SEC-3)

**Overall Learning Objectives:**

- Knowledge:
  - Understand the role of computers in physics.
  - Understand the concepts of algorithms and flowcharts.
  - Understand the fundamentals of the FORTRAN programming language.
  - Understand the basics of LaTeX for scientific document preparation.
  - Understand the basics of data visualization using Gnuplot.
- Skills:
  - Develop algorithms and draw flowcharts for solving physics problems.
  - Write and execute FORTRAN programs to solve physics problems.
  - Use LaTeX to create scientific documents with equations, figures, and tables.
  - Use Gnuplot to visualize and analyze data.
  - Apply computational methods to solve physics problems (e.g., numerical integration, simulation).

Materials and Resources:

- Computers with access to Linux operating system and FORTRAN compiler.
- LaTeX editor and compiler (e.g., TeX Live, Overleaf).
- Gnuplot software.
- Whiteboard or projector.
- Markers or pens.
- Handouts with exercises and examples.
- Textbook or online resources on FORTRAN programming, LaTeX, and Gnuplot.

# Lecture 1-4: Introduction & Algorithms

- Learning Objectives:
  - Understand the role of computers in physics.
  - Understand the concept of algorithms: Definition, properties, development.
  - Understand the concept of flowcharts: Symbols, guidelines, types.
  - Develop algorithms and draw flowcharts for simple physics problems.
- Content:
  - Introduction: Role of computers in physics, computational physics paradigm.
  - Algorithms: Definition, properties (finiteness, definiteness, effectiveness, input, output), algorithm development.
  - Flowcharts: Symbols, guidelines, types. Examples:
    - Algorithm and flowchart for Cartesian to Spherical Polar Coordinates conversion.
    - Algorithm and flowchart for finding roots of a quadratic equation.
- Activities:
  - In-class exercises: Developing algorithms and drawing flowcharts for simple problems.

# Lecture 5-9: Introduction to FORTRAN Programming

- Learning Objectives:
  - Understand basic Linux commands (internal and external).
  - Understand the basic elements of FORTRAN: Character set, constants, variables, data types, keywords.
  - Understand FORTRAN statements: I/O statements, executable and non-executable statements.
  - Understand the structure and layout of a FORTRAN program.
  - Write simple FORTRAN programs.
- Content:
  - Introduction to Linux: Basic commands (ls, cd, mkdir, rm, cp, mv, etc.).

- Introduction to FORTRAN: History, features.
- Basic elements: Character set, constants, variables, data types, keywords.
- Variables: Declaration, initialization.
- Operators: Arithmetic, relational, logical, assignment operators.
- Expressions: Arithmetic, relational, logical, character, assignment expressions.
- FORTRAN statements: I/O statements (read, write, format), executable and non-executable statements.
- Structure of a FORTRAN program: Program statement, end statement, comments.
- Layout of a FORTRAN program: Source code formatting.
- Activities:
- Hands-on exercises: Writing and executing simple FORTRAN programs (e.g., calculating area of a circle, converting units).

## Lecture 10-14: Control Structures & Data Structures in FORTRAN

- Learning Objectives:
- Understand control flow statements in FORTRAN: Branching (if, else if, select case), looping (do, continue, do while).
- Understand the concept of arrays in FORTRAN: Declaration, dimensioning, reading, writing.
- Understand the use of functions and subroutines in FORTRAN: Function subprogram, subroutine subprogram, arguments, return statement, call statement.
- Content:
- Control flow statements: Branching (if, else if, select case), looping (do, continue, do while).
- Arrays: Declaration, dimensioning, reading, writing, array operations.
- Functions and subroutines: Function subprogram, subroutine subprogram, arguments, return statement, call statement.
- Activities:

- Hands-on exercises: Writing FORTRAN programs using control structures, arrays, functions, and subroutines.

## Lecture 15-19: LaTeX and Scientific Document Preparation

- Learning Objectives:
- Understand the basics of LaTeX: Document classes, basic commands, creating a LaTeX file.
- Learn to typeset mathematical equations and formulas using LaTeX.
- Learn to create tables, figures, and other elements in LaTeX.
- Learn to format text, create lists, and change font styles.
- Learn to generate a table of contents, bibliography, and index.
- Content:
- Introduction to LaTeX: TeX/LaTeX word processor, basic commands.
- Creating a LaTeX file: Document classes, preamble, document body.
- Typesetting equations and formulas: Mathematical symbols, Greek letters, superscripts, subscripts, fractions, matrices.
- Formatting text: Fonts, font sizes, colors, spacing.
- Creating tables and figures: Tabular environment, figure environment, captions.
- Generating a table of contents, bibliography, and index.
- Activities:
- Hands-on exercises: Creating simple LaTeX documents.
- Creating a LaTeX document with equations, tables, and figures.

# Lecture 20-24: Data Visualization with Gnuplot

- Learning Objectives:
- Understand the importance of data visualization in physics.
- Learn basic Gnuplot commands: Plotting data from files, plotting functions, customizing plots.

- Understand how to save and export plots from Gnuplot.
- Apply Gnuplot to visualize data from physics simulations.

- Content:
- Introduction to graphical analysis and its limitations.
- Introduction to Gnuplot: Basic commands (plot, set, save, load).
- Plotting data from files: Data formats, reading data from files.
- Plotting functions: Defining functions, plotting functions.
- Customizing plots: Labels, titles, legends, colors, line styles.
- Saving and exporting plots: Saving plots in different formats (e.g., PNG, PDF, EPS).

- Activities:
- Hands-on exercises:
  - Plotting data from files.
  - Plotting functions (e.g., sine wave, exponential function).
  - Creating and customizing plots.

## Lecture 25-30: Applications and Advanced Topics

- Learning Objectives:
- Apply computational techniques to solve physics problems.
- Solve equations numerically (e.g., finding roots of equations).
- Simulate physical systems using numerical methods.

- Content:
- Numerical methods: Root finding methods (e.g., bisection method, Newton-Raphson method).
- Simulation of physical systems:
  - Example 1: Motion of a projectile.
  - Example 2: Motion of a simple harmonic oscillator.
  - Example 3: (Optional) Motion of a particle in a central force field.

- Advanced topics (if time permits):
  - Introduction to object-oriented programming concepts.
  - Brief introduction to other scientific computing languages (e.g., Python, MATLAB).
- Activities:
  - Hands-on exercises: Implementing numerical methods in FORTRAN.
  - Simulating and visualizing physical systems.

Assessment/Evaluation:

- Formative:
  - Class participation and discussions.
  - In-class quizzes on FORTRAN syntax and concepts.
  - Code reviews and feedback on programming assignments.
  - Short quizzes on LaTeX commands and Gnuplot commands.
- Summative:
  - Midterm exam covering algorithms, FORTRAN programming, and basic LaTeX.
  - Final exam covering all topics, including advanced applications and projects.
  - Programming assignments and projects.

Differentiation:

- Advanced learners:
  - Explore more advanced topics in FORTRAN (e.g., pointers, modules).
  - Work on more challenging programming projects.
  - Investigate other scientific computing languages (e.g., Python, MATLAB).
- Struggling learners:
  - Provide additional practice problems and one-on-one assistance.
  - Offer simplified explanations and alternative learning materials.
  - Break down assignments into smaller, more manageable steps.

Closure:

- Summarize the key concepts covered in the course.

- Discuss the importance of computational skills in modern physics research.
- Encourage students to explore further on their own (e.g., by working on personal projects).

Reflection:

- Were the learning objectives met?
- Were the activities engaging and effective?
- Were there any areas where the lesson could be improved?
- What strategies can be used to enhance student understanding in future lessons?