Centre For High Energy Physics

Faculty of Science
University of the Punjab, Lahore
Course Outline

Programme	BSCP	**Course Code**	CPHY 281	**Credit Hours**	3
Course Title	**Introduction to Scientific Computation**				

Course Introduction

The Computer Programming course offers a thorough investigation of Mathematica and Python-based mathematical computing. The course prepares students for more difficult programming problems by covering math, variables, lists, expressions, patterns, and replacement rules. The construction of functions, data visualization, symbolic and numerical computations, and the solution of linear and nonlinear equations are all covered in the course. The ability to execute accurate numerical computations and simplify algebraic statements will be taught to students. They will also gain knowledge of methods for solving differential equations symbolically as well as how to handle complicated mathematical problems utilizing vectors, matrices, and tensors. Data file reading and writing procedures, output formats, and input and output activities are all covered in the course. By the end, pupils will be adept math and science problem solvers.

Learning Outcomes

The course introduces the subject of scientific computing. Its objectives are as following.

1. Studying the concepts of computer arithmetic and approximations in computing.
2. Getting experience of working with different problem-solving environments.
3. Getting experience of working with different Scientific Libraries.

Course Content

Week 1	*Introduction to Python*
	• **Introduction to Python**
	  o What is Python?
	  o Installing Python and setting up the development environment.
	• **Basic Syntax**
	  o Writing and running Python scripts.
	  o Variables, data types, and basic operations.

	Introduction to Python □ **Control Flow** • Conditional statements (if, elif, else). • Loops (for, while). □ **Practice Problems** • Simple exercises to practice control flow and basic syntax.

Week 2	*Data Structures*
	• **Lists and Tuples**
	  o Creating and manipulating lists and tuples.
	  o List comprehensions.
	• **Dictionaries and Sets**
	  o Creating and using dictionaries and sets.
	  o Dictionary and set comprehensions.

	Data Structures □ **Strings** • String operations and formatting.

	Practice Problems • Exercises to work with different data structures.

Week 3	*Functions and Modules*
	• **Functions**
	  o Defining and calling functions.
	  o Parameters and return values.
	  o Lambda functions.
	• **Modules and Packages**
	  o Importing and using modules.
	  o Standard library overview.
	  o Creating and using packages.

	Error Handling • Try, except, and finally blocks. **Practice Problems** • Exercises on writing functions and using modules.

Week 4	*Data Manipulation and Analysis*
	• **File I/O**
	  o Reading from and writing to files.
	  o Working with CSV and JSON files.

	• Introduction toPandas • Installing Pandas • Data Structures: Series and DataFrame • Basic Operations: Creating, Viewing, and Inspecting Data

Week 5	*Data Manipulation and Analysis*
	Data Visualization
	  What is Matplotlib?
	  Installation and Setup
	  Basic Plotting: Line Plots, Scatter Plots, Bar Charts
	  Customizing Plots: Titles, Labels, Legends, and Annotations
	  Advanced Plotting: Subplots, Histograms, and 3D Plots

	Introduction to NumPy • What is NumPy? • Installation and Setup • NumPy Arrays: Creation, Indexing, and Slicing • Basic Operations: Arithmetic, Aggregation, and Broadcasting • Common Functions: numpy.arange(), numpy.linspace(), numpy.zeros(), numpy.ones(), and numpy.random **Practice Problems** • Basic array operations • Using NumPy for simple statistical calculations
Week 6	**Introduction to SciPy** • What is SciPy? • Installation and Setup • SciPy Modules Overview: scipy.optimize, scipy.integrate, scipy.interpolate, scipy.linalg
	Optimization and Integration • Optimization: Using scipy.optimize.minimize() for finding function minima • Integration: Using scipy.integrate.quad() for numerical integration **Interpolation and Linear Algebra**

	• Interpolation: Using scipy.interpolate.interp1d() for 1D interpolation • Linear Algebra: Using scipy.linalg for matrix operations **Practice Problems** • Solving optimization problems • Performing numerical integration and interpolation
Week 7	**Introduction to SymPy** • What is SymPy? • Installation and Setup • Symbolic Computation Basics: Variables, Expressions, and Simplification • Solving Algebraic Equations and Calculus Operations • Symbolic Integration and Differentiation
	Practice Problems • Simplifying expressions • Solving equations symbolically • Performing symbolic differentiation and integration
Week 8	**Introduction to Matplotlib** • What is Matplotlib? • Installation and Setup • Basic Plotting: Line Plots, Scatter Plots, Bar Charts • Customizing Plots: Titles, Labels, Legends, and Annotations • Advanced Plotting: Subplots, Histograms, and 3D Plots
	Practice Problems • Creating and customizing different types of plots • Visualizing data with advanced plotting techniques
Week 9	Mathematical Computations and Visualization **Numerical Integration and Differentiation** • Numerical Integration: Using scipy.integrate.quad(), scipy.integrate.simps() • Numerical Differentiation: Finite differences and numpy.gradient() **Root Finding and Optimization** • Root Finding: Using scipy.optimize.root(), scipy.optimize.brentq() • Finding the Minimum of a Function: Using scipy.optimize.minimize(), scipy.optimize.minimize_scalar()
	Symbolic Computations • Differentiation and Integration with SymPy: Using sympy.diff(), sympy.integrate() **Practice Problems** • Solving integrals and derivatives numerically and symbolically • Finding roots and minima of functions **Visualization** • Plotting results using Matplotlib: Line plots, scatter plots
Week 10	Linear Algebra and Advanced Numerical Methods **Computation with Vectors and Matrices** • Vectors and Matrices Operations with NumPy: Dot product, matrix multiplication, inverse, and eigenvalues **Tensors** • Introduction to Tensors: Basic operations and manipulations using NumPy **Gradient, Divergence, Curl** • Calculating Gradient, Divergence, and Curl: Using NumPy and SciPy functions

	Practice Problems • Matrix operations and tensor computations • Gradient, divergence, and curl calculations **Visualization** • Visualizing vector fields and tensor operations using Matplotlib
Week 11	Interpolation, Curve Fitting, and Series Approximations **Interpolation Functions and Curve Fitting** • Interpolation: Using scipy.interpolate.interp1d(), scipy.interpolate.CubicSpline() • Curve Fitting: Using scipy.optimize.curve_fit()
	Series Approximations • Series Expansions: Taylor series and Fourier series using SymPy
Week 12	**Practice Problems** • Fitting curves to data and performing interpolations • Approximating functions with series expansions **Visualization** • Plotting fitted curves and interpolation results
	Solving Differential Equations • Ordinary Differential Equations (ODEs): Using scipy.integrate.odeint(), scipy.integrate.solve_ivp() Symbolic Solutions: Using SymPy's dsolve()
Week 13	**Laplace Transforms and Inverse Laplace Transforms** • Laplace Transformations: Using SymPy's laplace_transform() • Inverse Laplace Transformations: Using SymPy's inverse_laplace_transform()
	Practice Problems • Solving ODEs numerically and symbolically • Applying Laplace transforms to solve differential equations
Week 14	**Visualization** • Plotting solutions of differential equations and transformed functions
	Advanced Methods and Applications **Variation of Parameters** • Theory and Application: Solving differential equations using the method of variation of parameters
Week 15	**Shooting Method** • Theory and Example: Using the shooting method to solve boundary value problems
	Numerical Solutions and Boundary Value Problems • Inhomogeneous Boundary Value Problems: Numerical methods for solving these problems **Practice Problems** • Applying the shooting method and variation of parameters • Solving inhomogeneous boundary value problems numerically
Week 16	**Final Project** • A comprehensive project integrating all concepts o Example Project: Solve and visualize a complex system involving differential equations, boundary conditions, and optimization
Textbooks and Reading Material	
1. Scientific Computing: An Introductory Survey, M. Heath, *McGraw-Hill International Edition* (1997). 2. Mathematica for Scientists and Engineers, Thomas B. Bahder, *Addison-Wesley* (1995).	

3. Introduction to Scientific Computing (1st edition), Brigitte Lucquin, *John Wiley & Sons* (1998).
4. Numerical Recipes in C: The Art of Scientific Computing (2nd Edition), W. H. Press, B. P. Teukolsky, W. T. Vetterling, *Cambridge University Press* (1992).

Teaching Learning Strategies

The instructor is required to make use of FORTRAN/C/C++/Mathematica/Python/C# to teach the concepts through visualization/antimutation and symbolic/numerical calculations. The students are required to solve a large portion of related exercises/questions/problems of the main textbooks.

Assignments: Types and Number with Calendar

At least two assignments and two quizzes. A course project may also be assigned.

Assessment

Sr. No.	Elements	Weightage	Details
1.	Midterm Assessment	35%	Written Assessment at the mid-point of the semester.
2.	Formative Assessment	25%	Continuous assessment includes Classroom participation, assignments, presentations, viva voce, attitude and behavior, hands-on-activities, short tests, projects, practical, reflections, readings, quizzes etc.
3.	Final Assessment	40%	Written Examination at the end of the semester. At least fifty percent of the question paper would involve new problems related to the concepts learned in the course. It is mostly in the form of a test, but owing to the nature of the course the teacher may assess their students based on term paper, research proposal development, field work and report writing etc.