YOGI VEMANA UNIVERSITY::KADAPA

Syllabus for 4-Year UG Honours in B.Sc. (Quantum Technologies) as Major in consonance with Curriculum framework w.e.f. AY 2025-26 COURSE STRUCTURE (for Semester I to VI)

Year	Semester	Course	Title of the Course	No.of Hrs /Week	No. of Credits
I	I	1	Computer Fundamentals and Office Automation	3	3
			Computer Fundamentals and Office Automation-Practical	2	1
		2	Mathematical Physical and Computing foundations of Quantum Computing	3	3
			Mathematical Physical and Computing foundations of Quantum Computing -practical	2	1
	II	3	Problem Solving using C	3	3
			Problem Solving using C - Practical	2	1
		4	Numerical Methods for Quantum Computing	3	3
			Numerical Methods for Quantum Computing - Practical	2	1
II	III	5	Data Structures using Python	3	3
			Data Structures using Python-Practical	2	1
		6	Operating Systems	3	3
			Operating Systems-Practical	2	1
		7	Foundations of Quantum Technologies	3	3
			Foundations of Quantum Technologies-Practical	2	1
	IV	8	Python for Quantum Computing	3	3
			Python for Quantum Computing-Practical	2	1
		9	Quantum Computing using Qiskit	3	3
			Quantum Computing using Qiskit -Practical	2	1
		10	Quantum Computing Algorithms	3	3
			Quantum Computing Algorithms -Practical	2	1

III	**V**	**11**	Quantum Computing Applications	**3**	**3**
			Quantum Computing Applications -Practical	**2**	**1**
Year	**Semester**	**Course**	**Title of the Course**	**No.of Hrs /Week**	**No. of Credits**
		12 A	Computer Networks	**3**	**3**
			Computer Networks-Practical	**2**	**1**
		OR			
		12 B	Mathematics for Machine Learning	**3**	**3**
			Mathematics for Machine Learning-Practical	**2**	**1**
		13 A	Classical Cryptography & Network Security	**3**	**3**
			Classical Cryptography & Network Security-Practical	**2**	**1**
		OR			
		13 B	Python for Machine Learning	**3**	**3**
			Python for Machine Learning-Practical	**2**	**1**
		14 A	Quantum Cryptography Protocols	**3**	**3**
			Quantum Cryptography Protocols-Practical	**2**	**1**
		OR			
		14 B	Deep Learning Fundamentals	**3**	**3**
			Deep Learning Fundamentals-Practical	**2**	**1**
	VI	**15 A**	Quantum Simulation in Cyber Security	**3**	**3**
			Quantum Simulation in Cyber Security-Practical	**2**	**1**
		OR			
		15 B	Quantum Machine Learning	**3**	**3**
			Quantum Machine Learning-Practical	**2**	**1**

Note: In the III Year (during the V and VI Semesters), students are required to select a pair of electives from one of the **TWO** specified domains. **For example: if set 'A' is chosen, courses 12 to 15 to be chosen as 12A, 13A, 14A and 15A.** To ensure in-depth understanding and skill development in the chosen domain, students must continue with the same domain electives in both the V and VI Semesters.

SEMESTER-I

COURSE 1: COMPUTER FUNDAMENTALS AND OFFICE AUTOMATION

Theory **Credits: 3** **3 hrs/week**

Course Objectives

1. **Understand foundational computing concepts**, including number systems, the evolution of computers, block diagrams, and generational progress.
2. **Develop knowledge of computer architecture**, focusing on system organization and networking fundamentals.
3. **Acquire practical skills in document creation**, formatting, and digital presentations using word processing tools.
4. **Gain proficiency in spreadsheet operations**, such as data entry, formulas, functions, and charting techniques.
5. **Introduce data visualization and basic modelling principles**, fostering analytical thinking in structuring and interpreting data sets.

Course Outcomes

1. At the End of the Course, The Students will be able to **explain different number systems**, the historical evolution of computers, and identify key components in a block diagram.
2. Learners will demonstrate **basic blocks of a computer and fundamental networking knowledge.**
3. Learners will create professional-level documents and **design visually appealing presentations** using word processing software and presentation software.
4. Learners will manipulate data within spreadsheets, apply formulas, and **generate accurate summaries and visualizations.**
5. Learners will apply data modelling techniques to **analyze, organize, and represent data effectively** in various scenarios.

Syllabus:

Unit 1. Number Systems, Evolution, Block Diagram and Generations:

Number Systems: Binary, Decimal, Octal, Hexadecimal; conversions between number systems.

Evolution of Computers: History from early mechanical devices to modern-day systems.

Block Diagram of a Computer: Components like Input Unit, Output Unit, Memory, CPU (ALU + CU).

Generations of Computers: First to Fifth Generation with technologies, characteristics, examples.

Unit 2. Basic organization and N/W fundamentals:

Computer Organization: Functional components –Storage types, Memory hierarchy.

Types of Computers: Micro, Mini, Mainframe, and Supercomputers.

Networking Fundamentals: Definition, need for networks, types (LAN, WAN, MAN), topology (Star, Ring, Bus).

Internet Basics: IP Address, Domain Name, Web Browser, Email, WWW.

Unit 3. Word Processing and presentations:

Word Processing Basics: Using MS Word/Google Docs – formatting, styles, tables, mail merge. Applications - Creating resumes, reports. Keyboard Shortcuts- File Operations, Editing Operations, Formatting Shortcuts, Navigation and Selection.

Presentation Tools: Using PowerPoint/Google Slides – slide design, animations, transitions. Applications - Brochures, and presentations.

Unit 4. Spreadsheet Basics:

Spreadsheet Concepts: Understanding rows, columns, cells in tools like MS Excel/Google Sheets, cell referencing.

Functions and Formulae: SUM, AVERAGE, IF, COUNT.

Charts and Graphs: Creating visual representations, Types of Charts.

Data Handling: Sorting, filtering, conditional formatting.

Text Functions: LEFT, RIGHT, MID, LEN, TRIM, CONCAT, TEXTJOIN

Advanced Functions: Logical: IF, AND, OR, IFERROR, Lookup

Unit 5. Data Analysis and Visualization:

Conditional Formatting: Custom rules, Color scales, Icon sets, Data bars

Data Analysis Tools: Pivot Tables and Pivot Charts, Data Validation (Drop-downs, Input Messages, Error Alerts).

What-If Analysis: Goal Seek, Scenario Manager, Data Tables

Charts and Dashboards: Creating Interactive Dashboards, Using slicers with Pivot Tables, Combo Charts and Sparklines

Textbooks:

1. **Fundamentals of Computers,** Reema Thareja, Oxford University Press, Second Edition
2. **Fundamentals of Computers**, V. Rajaraman – PHI Learning
3. **Introduction to Computers** by Peter Norton – McGraw Hill
4. **Microsoft Office 365 In Practice** by Randy Nordell – McGraw Hill Education

References:

1. **Excel 2021 Bible** by Michael Alexander, Richard Kusleika – Wiley
2. **Networking All-in-One For Dummies** by Doug Lowe – Wiley
3. **Microsoft Official Docs and Training**: https://learn.microsoft.com
4. **Google Workspace Learning Center**: https://support.google.com/a/users/

Activities:

Outcome: At the End of the Course, The Students will be able to **explain different number systems**, the historical evolution of computers, and identify key components in a block diagram.

Activity: Create a digital poster or info graphic comparing number systems (binary, decimal, octal, hexadecimal) and illustrating the timeline of computer generations with key innovations.

Evaluation Method: Rubric-based assessment of the poster presentation on a 10-point scale focusing on:

- Accuracy of number system conversions
- Correct identification of block diagram components
- Visual organization and creativity

Outcome: Learners will demonstrate **basic blocks of a computer and fundamental networking knowledge.**

Activity: Design a concept map showing the internal architecture of a computer and types of networks (LAN, WAN, MAN), including devices and topologies.

Evaluation Method: Checklist-based peer review and instructor validation:

- Completeness of the map

- Correctness of networking concepts
- Use of appropriate terminology
- Logical flow and structure of the map

Outcome: Learners will create professional-level documents and **design visually appealing presentations** using word processing software and presentation software.

Activity: Prepare a formal report (e.g., project proposal) in a word processor and present it using a slide deck with transitions, embedded media, and design elements.

Evaluation Method: Performance-based evaluation using a 10-point scoring scale:

- Formatting and structure of the document
- Presentation aesthetics and clarity
- Communication skills during presentation

Outcome: Learners will manipulate data within spreadsheets, apply formulas, and **generate accurate summaries and visualizations.**

Activity: Analyze a dataset (e.g., student scores or sales data) using spreadsheet software. Apply formulas (SUM, AVERAGE, IF, VLOOKUP) and create relevant charts.

Evaluation Method: Practical test with a rubric:

- Correct use of formulas
- Accuracy of data summaries

Outcome: Learners will apply data modeling techniques to **analyze, organize, and represent data effectively** in various scenarios.

Activity: Prepare an interactive dashboard for a given data set using EXCEL.

Evaluation Method: Evaluation of the dashboard on a 10-point scoring scale:

- Presentation aesthetics and clarity
- Inter activeness
- Communication skills during presentation

SEMESTER-I

COURSE 1: COMPUTER FUNDAMENTALS AND OFFICE AUTOMATION

Practical **Credits: 1** **2 hrs/week**

List of Experiments:

1. Demonstration of Assembling and Dissembling of Computer Systems.
2. Identify and prepare notes on the type of Network topology of your institution.
3. Prepare your resume in Word.
4. Using Word, write a letter to your higher official seeking 10-days leave.
5. Prepare a presentation that contains text, audio and video.
6. Using a spreadsheet, prepare your class Time Table.
7. Using a Spreadsheet, calculate the Gross and Net salary of employees(Min 5) considering all the allowances.
8. Generate the class-wise and subject-wise results for a class of 20 students. Also generate the highest and lowest marks in each subject.
9. Using IF, AND, OR, and IFERROR to Automate Grade Evaluation.
    a. Create a table of student scores in different subjects.
    b. Use IF to assign grades (A/B/C/Fail).
    c. Use IFERROR to handle missing scores or invalid data.
10. Employee Database Search Using VLOOKUP, HLOOKUP, XLOOKUP, INDEX, and MATCH
    a. Create a database of employees (Name, ID, Department, Salary).
    b. Implement VLOOKUP to search by employee ID.
    c. Use HLOOKUP to extract department heads by role.
    d. Apply XLOOKUP for more flexible searches.
    e. Use INDEX + MATCH as an alternative to VLOOKUP.
11. Sales Report Analysis Using Pivot Tables and Charts
    a. Use a dataset of product sales (Product, Region, Date, Quantity, Revenue).
    b. Create Pivot Tables to summarize data by region/product.
    c. Insert Pivot Charts for visual analysis (e.g., bar, line).
    d. Add slicers to make the dashboard interactive.
12. Designing a Data Entry Form with Drop-downs and Input Rules
    a. Create a student registration form.
    b. Add drop-down lists for course selection using Data Validation.
    c. Add input messages to guide users.
    d. Add error alerts for wrong entries.
13. Monthly Budget Planning using Goal Seek and Scenario Manager
    a. Create a simple personal budget (income, expenses, savings).
    b. Use Goal Seek to determine income needed to save a desired amount.
    c. Use Scenario Manager to compare different budgeting scenarios (best/ worst/ realistic case).
    d. Create a one-variable Data Table to analyze how different expenses affect savings.
14. Dashboard Creation Using Combo Charts, Sparklines & Slicers
    a. Use existing sales or attendance data.
    b. Insert combo charts (e.g., column + line).
    c. Add spark lines to show trends.
    d. Use slicers with Pivot Tables to control dashboard elements.
    e. Finalize and format for interactivity.

SEMESTER-I

COURSE 2: Mathematical Physical and Computing foundations of Quantum Computing

Theory **Credits: 3** **3 hrs/week**

Course Objectives:

By the end of the course, students will be able to:

1. Understand foundational mathematical concepts such as vectors, matrices, and linear transformations in quantum computing.
2. Explore the mathematical representation of quantum gates and qubits using linear algebra.
3. Apply complex numbers, eigenvalues, and eigenvectors to quantum computational models.
4. Relate abstract mathematical ideas to practical quantum computing operations.
5. Develop problem-solving skills for quantum mechanics and quantum circuit analysis.
6. To introduce the evolution of computer science and its mathematical basis.

Learning Outcomes:

Students will be able to:

1. Apply complex number operations in quantum computations.
2. Compute eigenvalues and eigenvectors for quantum state matrices.
3. Interpret the Bloch sphere for single-qubit representation.
4. Identify Hermitian and unitary operators as foundational elements in quantum mechanics.
5. Understand the evolution and fundamental experiments leading to quantum mechanics.
6. Explain the mathematical and physical foundations behind superposition and entanglement.
7. Explain classical computational models and their historical development.

Unit I: Foundations of Quantum Mathematics (9 Periods)

Vectors and Linear Combinations: Euclidean vectors, magnitude and direction, scalar multiplication, vector addition.

Superposition and Measurement: Understanding superposition states and measurement in quantum systems.

Introduction to Matrices: Matrix definition, notation, simple operations (addition, transpose, scalar multiplication).

Matrix Multiplication and Properties: Matrix-vector and matrix-matrix multiplication.

Quantum Gates and Circuit Models: Logic gates vs quantum gates, matrix representation of quantum gates (NOT, AND, OR, Pauli-X).

Unit II: Elementary Linear Algebra for Quantum Systems (9 Periods)

Sets, Functions, and Vector Spaces: Sets, Cartesian product, functions, fields, and vector spaces.

Subspaces, Basis, and Dimension: Linear independence, span, and basis of a vector space.

Linear Transformations: Definition, properties, and matrix representation.

Transformations Inspired by Euclid: Translation, rotation, and projection.

Linear Operators and Functionals: Matrix dependence on basis, commutator operations.

Unit III: Complex Numbers and Eigen Concepts (9 Periods)

Complex Number System: Cartesian, polar, and exponential forms; conjugates and modulus.

Complex Numbers in Quantum Mechanics: Bloch sphere representation and Euler's formula.

Eigenvalues and Eigenvectors: Definitions, computation, and physical significance.

Determinants and Invertibility: Matrix inverse and the invertible matrix theorem.

Applications in Quantum State Analysis: Diagonalization, Hermitian and unitary operators.

Unit 4 – Quantum Mechanics and Its Principles

History of Quantum Mechanics: Classical Physics foundations (Newton, Maxwell, Galileo),Failures of classical theory and emergence of quantum concepts,Key contributors: Planck, Einstein, Bohr, Heisenberg, Schrödinger, Dirac

Atoms and Thermodynamics: Atomic theory, kinetic theory, and laws of thermodynamics,**Statistical Mechanics:** Maxwell–Boltzmann distribution and entropy (Boltzmann's constant),**Photoelectric Effect:** Einstein's explanation of quantized energy, **Wave–**

Particle Duality: de Broglie's matter waves and Bohr's model, **Quantum Models:** Schrödinger's wave equation, Heisenberg's matrix mechanics, **Copenhagen Interpretation:** Bohr's complementarity and probabilistic measurement

Key Principles for Quantum Computing: Linear Algebra in Quantum Systems,Superposition and Interference, Dirac (Bra–Ket) Notation ,Quantum Uncertainty (Heisenberg Principle), Entanglement and Non-locality, Postulates of Quantum Mechanics

Unit-5: Introduction to Computer Science, Historical Foundations, Turing Machines, Logic Circuits and Gates, Computational Complexity and Efficiency,Energy, Reversibility, and Computation

Text Books:

1. ***Essential Mathematics for Quantum Computing*** **by Leonard S. Woody III (Packt, 2022)**.
2. ***Quantum Computing from the Ground Up by Riley Tipton Perry, by World Scientific Publishing Co. Pte. Ltd.***

SEMESTER-I

COURSE 2: Mathematical Physical and Computing foundations of Quantum Computing

Practical **Credits: 1** **2 hrs/week**

List of Experiments:

Tools: GeoGebra, Desmos, Python (NumPy), or any simple matrix calculator.

No.	Experiment Title	Concepts Covered	Tool/Method
1	**Vector Addition and Superposition**	Represent two 2D vectors and visualize their superposition (quantum state analogy).	GeoGebra / Desmos
2	**Matrix Multiplication Basics**	Multiply 2×2 matrices and visualize the transformation of a 2D point.	Python NumPy / GeoGebra
3	**Quantum NOT Gate Simulation**	Implement NOT gate matrix (X = \begin{bmatrix}0 & 1 \ 1 & 0\end{bmatrix}) and apply to	0⟩ and
4	**Matrix Transpose and Symmetry Check**	Create any 3×3 matrix and verify (A = A^T) (symmetric) or (A \neq A^T).	Simple math tool
5	**Vector Space Verification**	Verify if given vectors form a subspace under addition and scalar multiplication.	GeoGebra / manual
6	**Linear Transformation Visualization**	Use a 2×2 matrix to rotate or project points in 2D space.	GeoGebra
7	**Complex Number Plane Plotting**	Plot complex numbers in Cartesian and Polar forms; verify Euler's relation (e^{iθ} = \cosθ + i\sinθ).	Desmos / Python
8	**Bloch Sphere Projection (Intro)**	Represent	0⟩ and

Tools: Logisim or Logisim Evolution

No.	Experiment Title	Concept	Tool/Method
1	**Binary to Decimal Converter**	Implement conversion circuit using XOR, AND, and OR gates.	Logisim
2	**Half Adder and Full Adder Circuits**	Design classical addition logic for two bits, then extend to full adder.	Logisim
3	**NOT, AND, OR, XOR Gates Demonstration**	Build and verify truth tables for basic logic gates.	Logisim
4	**Turing Machine Emulator (Basic)**	Simulate tape read/write using a simple state transition circuit.	Logisim
5	**Binary Pattern Recognizer**	Design finite automata circuit that outputs high for a specific binary pattern (like "101").	Logisim
6	**Logic-to-Quantum Transition Concept**	Compare Boolean logic gates to unitary matrix equivalents (e.g., NOT ↔ X gate).	Logisim + explanation

SEMESTER-II

COURSE3: PROBLEM SOLVING USING C

Theory **Credits:3** **3 hrs/week**

Course Objectives:

1. Understand the fundamentals of computer programming, Apply structured problem-solving approaches using algorithms, flowcharts, and C programming constructs.
2. Develop efficient logic using decision-making, loop, and jump control statements.
3. Utilize derived data types like arrays and strings for modular program design.
4. Design and implement modular solutions using functions, recursive logic, pointer operations, and dynamic memory management.
5. Handle complex data structures including structures, unions, and text file operations.

Course Outcomes:

At the end of the course, students will be able to:

1. Understand basic computing concepts, programming paradigms and write structured C programs.
2. Apply control flow statements to solve logical and repetitive tasks in C.
3. Implement arrays and string operations to manage and manipulate data efficiently.
4. Design modular code using functions, recursion, and appropriate parameter passing.
5. Utilize pointers and memory operations for effective data handling. Demonstrate competence in dynamic memory allocation and text file processing.

Unit 1. Introduction to computer programming:

Introduction, Types of software, Compiler and interpreter, Concepts of Machine level, Assembly level and high-level programming, Flowcharts and Algorithms, Fundamentals of C: History of C, Features of C, C Tokens-variables and keywords and identifiers, constants and Data types, Rules for constructing variable names, Operators, Structure of C program, Input /output statements in C-Formatted and Unformatted I/O

Unit 2. Control statements:

Decision making statements: if, if else, else if ladder, switch statements. Loop control statements: while loop, for loop and do-while loop. Jump Control statements: break,continue and goto.

Unit 3. Derived datatypes in C:

Arrays: One Dimensional arrays - Declaration, Initialization and Memory representation; Two Dimensional arrays - Declaration, Initialization and Memory representation. Strings: Declaring & Initializing string variables; String handling functions, Character handling functions

Unit 4. Functions:

Pointers: Pointer data type, Pointer declaration, initialization, accessing values using pointers. Pointer arithmetic, Pointers and arrays.

Function Prototype, definition and calling. Return statement. Nesting of functions. Categories of functions. Recursion (Basic Concept only). Parameter Passing by address & by value. Local and Global variables. Storage classes: automatic, external, static and register.

Unit 5. Dynamic Memory Management:

Introduction, Functions - malloc, calloc, realloc, free Structures: Basics of structure, structure members, accessing structure members, nested structures, array of structures, structure and functions, structures and pointers. Unions - Union definition; difference between Structures and Unions. Working with text files - modes: opening, reading, writing and closing text files.

Text Books:

1. Programming in ANSI C, E. Balagurusamy, Tata McGraw Hill, 6th Edn
2. Computer fundamentals and programming in C, Reema Theraja, Oxford University Press

Reference Books:

1. Let us C, Y. Kanetkar, BPB publications
2. Head First C: A Brain-Friendly Guide, David Griffiths, Dawn Griffiths

Activities:

Outcome: Understand basic computing concepts, programming paradigms and write structured C programs.

Activity: Create a concept map of computing fundamentals and programming paradigms (procedural, structured, object-oriented). Then, they write a structured C program (e.g., a calculator or student grade system) using proper syntax, indentation, and modular design.

Evaluation Method: Rubric-based Code Review & Viva to check the ○ The correctness of the concept map
○ Correct use of structure (main + functions)

○ Identification of paradigm used
○ Code readability and documentation

Outcome: Apply control flow statements to solve logical and repetitive tasks in C.

Activity: Implement a program that solves a logic puzzle (e.g., number guessing game, pattern

generation, or prime number finder) using if, switch, for, while, and do-while.
Evaluation Method: Automated Test Cases + Peer Review to check the
○ Correct use of control statements
○ Logical correctness of output
○ Efficiency and edge case handling
○ Peer feedback on clarity and logic
Outcome: Implement arrays and string operations to manage and manipulate data efficiently.
Activity: Build a program that stores and arranges student marks in ascending and descending order using arrays and performs string operations like concatenation, comparing, and formatting names.
Evaluation Method: Functional Demonstration + Code Walkthrough to check the
○ Correct array and string usage
○ Memory efficiency
○ Handling of invalid inputs
○ Explanation of sorting/searching logic
Activity:
● Recursive Problem Solver
Students write a modular program to solve a recursive problem (e.g., factorial, Fibonacci, or Tower of Hanoi) using functions with parameters and return values.
Evaluation Method:
● Code Trace + Written Quiz
○ Correct function decomposition
○ Proper parameter passing (by value/reference)
○ Recursion depth and base case handling
○ Quiz on tracing recursive calls

Outcome: Utilize pointers and memory operations for effective data handling. Demonstrate competence in dynamic memory allocation and text file processing.
Activity: Create a program that dynamically stores user input (e.g., survey responses) using pointers and writes/reads the data to/from a text file.
Evaluation Method: Memory Debugging + File I/O Assessment to check the ○ Proper use of malloc, calloc, realloc, and free
○ Pointer arithmetic and dereferencing
○ File creation, reading, writing, and error handling
○ Use of tools like Valgrind or manual memory trace (Optional for Unix flavours)

SEMESTER-II

COURSE3:PROBLEM SOLVING USING C

Practical **Credits:1** **2 hrs/week**

List of Experiments

1. Write a program to check whether the given number is Armstrong or not.
2. Write a program to find the sum of individual digits of a positive integer.
3. Write a program to generate the first *n* terms of the Fibonacci sequence.
4. Write a program to find both the largest and smallest number in a list of integer values.
5. Write a program to demonstrate change in parameter values while swapping two integer variables using Call by Value & Call by Address.
6. Write a program to perform various string operations.
7. Write a program to search an element in a given list of values.
8. Write a program that uses functions to add two matrices.
9. Write a program to calculate factorial of a given integer value using recursive functions.
10. Write a program for multiplication of two N × N matrices.
11. Write a program to sort a given list of integers in ascending order.
12. Write a program to calculate the salaries of all employees using the Employee (ID, Name, Designation, Basic Pay, DA, HRA, Gross Salary, Deduction, Net Salary) structure.
    a. DA is 30% of Basic Pay
    b. HRA is 15% of Basic Pay
    c. Deduction is 10% of (Basic Pay + DA)
    d. Gross Salary = Basic Pay + DA + HRA
    e. Net Salary = Gross Salary – Deduction
13. Write a program to read/write the data from/to a file.
14. Write a program to reverse the contents of a file and store in another file.
15. Write a program to create Book (ISBN, Title, Author, Price, Pages, Publisher) structure and store book details in a file and perform the following operations:
    a. Add book details
    b. Search a book details for a given ISBN and display book details, if available
    c. Update a book details using ISBN
    d. Delete book details for a given ISBN and display list of remaining books.

SEMESTER-II

COURSE 4: Numerical Methods for Quantum Computing

Theory **Credits: 3** **3 hrs/week**

Course Objectives:

1. To understand the foundations of numerical computation and approximation.
2. To analyze different types and sources of numerical errors.
3. To apply numerical methods for solving algebraic and linear systems.
4. To compute numerical differentiation and integration using appropriate algorithms.
5. To solve ordinary differential equations using iterative and Runge-Kutta methods.

Course Learning Outcomes

After completing this course, students will be able to:

1. Explain the significance and limitations of numerical methods in solving engineering problems.
2. Identify and minimize different sources of numerical errors in computation.
3. Apply root-finding and linear system solving methods to practical problems.
4. Perform numerical differentiation and integration accurately.
5. Implement numerical algorithms to solve ordinary differential equations with controlled error.

Unit–I: Introduction to Numerical Methods and Errors (9 Periods)

Aim of Numerical Methods, Concept of Numerical Approximation,Measuring Errors: Absolute, Relative, and Percentage Errors, Important Definitions and Theorems Related to Numerical Methods, Round-off and Truncation Errors, Error Propagation and Machine Epsilon, Trade-off between Round-off and Truncation Errors

Unit–II: Solution of Algebraic and Transcendental Equations (9 Periods)

Classification of Equations, Bracketing Methods: Bisection and False Position Methods, Open Methods: Fixed-Point Iteration, Newton-Raphson Method, Convergence Analysis and Order of Convergence, Estimation of Errors and Convergence Rate, Backward and Forward Error Analysis

Unit–III: Systems of Linear Equations (9 Periods)

Matrix Representation of Linear Systems, Gaussian Elimination and Gauss-Jordan Methods, LU Decomposition and Pivoting, Iterative Methods: Jacobi and Gauss-Seidel Methods, Convergence Criteria and Error Estimation

Unit–IV: Numerical Differentiation and Integration (9 Periods)

Numerical Differentiation using Finite Differences, Polynomial Interpolation and Lagrange Interpolation, Newton-Cotes Quadrature Formulas (Trapezoidal, Simpson's Rules), Romberg Integration and Richardson Extrapolation, Gauss Quadrature Methods and Error Estimation

Unit–V: Numerical Solutions of Differential Equations (9 Periods)

Initial Value Problems (IVPs), Euler's Method – Stability and Convergence, Runge-Kutta Methods (Second and Fourth Order), Adaptive Methods for IVPs, Systems and Higher-Order Differential Equations, Global and Local Truncation Errors

Text Books:

1. Numerical Methods for Engineering and Data Science (Wuthrich & El Ayoubi, CRC Press, 2025
2. Numerical Methods Fundamentals by R.V.Dukkipati

SEMESTER-II

COURSE 4: Numerical Methods for Quantum Computing

Practical **Credits: 1** **2 hrs/week**

List of Experiments:

Exp. No.	Title of Experiment	Objective	Major Tasks / Activities	Software / Tools
1	Study of Round-off and Truncation Errors	To understand the effect of finite precision and floating-point arithmetic on numerical results.	Perform addition/subtraction of small & large numbers, observe round-off errors, compute machine epsilon (`eps`).	Octave / Python (`sys.float_info.epsilon`)
2	Error Propagation and Significant Digits	To study how input data errors propagate through calculations.	Evaluate propagation of input errors using addition/subtraction; demonstrate loss of significance.	Octave / Python
3	Bisection Method for Nonlinear Equations	To find the root of a nonlinear equation using a bracketing method.	Implement Bisection Method for (f(x)=x^3 - 4x + 1); plot error reduction per iteration.	Octave / Python
4	Newton-Raphson and Fixed Point Iteration	To solve nonlinear equations and compare convergence speed.	Apply Newton-Raphson and Fixed-Point Iteration to (f(x)=e^{-x}-x); compare iteration count and error.	Octave / Python
5	Solving Linear Systems using Gaussian Elimination	To solve a linear system using the direct elimination approach.	Input 3×3 system; perform forward elimination and back substitution; verify with built-in solver.	Octave / Python (`numpy.linalg.solve()`)
6	LU Decomposition Method	To solve linear systems efficiently using LU factorization.	Decompose matrix (A = LU); solve for (X) given (B); compare with direct methods.	Octave (`lu()`), Python (`scipy.linalg.lu()`)
7	Polynomial and Lagrange Interpolation	To approximate a function from discrete data points.	Input (x, y) values; generate interpolation polynomial; plot interpolated curve and compare with true function.	Octave / Python (`numpy.polyfit`, `matplotlib`)

Exp. No.	Title of Experiment	Objective	Major Tasks / Activities	Software / Tools
8	Numerical Integration using Trapezoidal and Simpson's Rules	To compute definite integrals numerically and compare accuracy.	Implement both rules for (f(x)=\sin(x)) on [0, π]; compare with analytical result.	Octave / Python (`scipy.integrate`)
9	Euler's Method for Solving First-Order ODE	To solve an initial value problem numerically using Euler's method.	Solve (\frac{dy}{dx}=y-x^2+1); compare results for step sizes h=0.1, 0.05; plot error vs. iteration.	Octave / Python
10	Runge-Kutta (RK4) Method for IVPs	To apply the 4th order Runge-Kutta method and compare with Euler's.	Solve (\frac{dy}{dx}=x+y, y(0)=1); compute RK4 solution; compare accuracy & convergence with Euler's method.	Octave / Python

SEMESTER-III

COURSE 5: Data Structures using Python

Theory **Credits: 3** **3 hrs/week**

Course Objectives:

1. To introduce the concept and classification of data structures and their role in efficient programming.
2. To understand algorithm design, analysis, and implementation using Python.
3. To study and apply fundamental data structures such as arrays, linked lists, stacks, queues, trees, and graphs.
4. To analyze searching, sorting, and hashing techniques for problem-solving efficiency.
5. To implement data structures using Python and explore their practical applications in computing.

Course Learning Outcomes:

After successful completion of this course, students will be able to:

1. Explain the fundamental concepts of data structures and their operations.
2. Design and implement algorithms for solving computational problems efficiently.
3. Apply and manipulate linear data structures such as arrays, stacks, queues, and linked lists.
4. Build and traverse hierarchical data structures like trees and graphs using Python.
5. Analyze the time and space complexity of algorithms and implement hashing for efficient data access.

Unit 1: Introduction to Data Structures (9 Periods)

Definition, importance, and applications of data structures, Types of data structures: Linear and Non-Linear, Static and Dynamic, Homogeneous and Non-Homogeneous, Primitive vs Non-Primitive data types, Operations on data structures (Creation, Insertion, Deletion, Traversal, Sorting, etc.),Algorithm design and analysis: Time & Space complexity, Big-O notation, Time-Space trade-off,Abstract Data Types (ADT)

Unit 2: Arrays and Lists (9 Periods)

Definition and representation of arrays/lists, Array initialization and address calculation, Operations on arrays/lists (traversing, insertion, deletion, searching, merging), Multidimensional arrays and sparse matrices, Applications of arrays/lists

Unit 3: Linked Lists (9 Periods)

Concept and memory representation of linked lists, Types of linked lists: singly, circular, and doubly linked lists, Operations: creation, insertion, deletion, traversal, searching, Header linked lists and applications. Polynomial representation using linked lists

Unit 4: Stacks and Queues (9 Periods)

Stack definition, operations (push, pop, peek), and implementation using arrays/lists and linked lists. Applications: expression conversion, postfix/prefix evaluation, recursion, parenthesis balancing, Queue definition and operations (enqueue, dequeue), Types of queues: Circular Queue, Priority Queue, Double-Ended Queue (Deque),Applications of queues

Unit 5: Trees, Graphs, and Hashing (9 Periods)

Trees: terminology, binary tree, binary search tree, AVL tree – representation and traversal methods, Graphs: definitions, types, representation (adjacency matrix/list), BFS and DFS traversal, Searching and Sorting overview: linear, binary, and interpolation search; insertion and quick sort basics, Hashing: hash functions, hash tables, collision and resolution techniques, Applications in file structures and indexing

Text Books:

1. *Data Structures and Program Design Using Python"* by Dheeraj Malhotra & Neha Malhotra.
2. *Data Structures and Algorithms in Java/C++"*) by Robert Lafore

SEMESTER-III

COURSE 5: Data Structures using Python

Practical **Credits: 1** **2 hrs/week**

Experiment No.	Name of the Experiment	Concepts / Topics Covered
1	Write a Python program to create, insert, delete, and traverse elements of different primitive data structures.	Basic data operations, ADT
2	Develop and test simple algorithms (e.g., find max/min, average, factorial) and calculate their time complexity using Big-O notation.	Algorithm design, complexity analysis
3	Implement a Python program to perform insertion, deletion, and traversal in a list/array.	Array operations and manipulation
4	Develop a program to perform matrix addition, subtraction, and multiplication using 2-D arrays/lists.	Multidimensional arrays and matrices
5	Implement sparse matrix representation and display its triplet form.	Sparse matrices representation
6	Create and traverse a singly linked list using Python classes and objects.	Node creation and traversal
7	Implement insertion and deletion operations in singly and doubly linked lists.	Dynamic memory and pointer manipulation
8	Write a program to represent and add two polynomials using linked lists.	Application of linked lists
9	Implement a stack using Python lists and perform push, pop, and peek operations.	Stack ADT and operations
10	Convert an infix expression to postfix and evaluate the postfix expression using a stack.	Expression conversion and evaluation
11	Implement queue operations (enqueue, dequeue) using lists and linked lists. Demonstrate Circular Queue and Priority Queue.	Queue types and applications
12	Implement creation and traversal (inorder, preorder, postorder) of a Binary Search Tree.	Tree representation and traversal
13	Write a program to perform insertion and rotation in an AVL tree.	Height-balanced trees
14	Represent a graph using adjacency matrix and adjacency list; implement BFS and DFS.	Graph representation and traversal algorithms
15	Implement hashing with linear probing and quadratic probing for collision resolution.	Hash tables, hash functions, collision resolution

SEMESTER-III

COURSE 6: Operating Systems

Theory **Credits: 3** **3 hrs/week**

Course Objectives:

1. To understand the fundamental principles, architecture, and functionality of operating systems.
2. To explore process, thread, and CPU scheduling mechanisms for multitasking systems.
3. To study memory management, paging, segmentation, and virtual memory algorithms.
4. To understand file system architecture, I/O subsystems, and device management.
5. To gain hands-on experience with OS concepts using Linux system programming and shell utilities.

Course Learning Outcomes:

After completing this course, students will be able to:

1. Explain the fundamental components, structure, and types of operating systems.
2. Analyze and simulate process management, scheduling, and synchronization techniques.
3. Apply memory management and virtual memory concepts to system design.
4. Demonstrate understanding of file systems, I/O handling, and system call mechanisms.
5. Evaluate operating system designs and compare real-world systems such as Linux, Android, and Windows.

Unit I – Introduction to Operating Systems (9 Periods)

Definition, Objectives, and Functions of Operating Systems, OS as Extended Machine and Resource Manager, Evolution: Batch, Multiprogramming, Time Sharing, Personal and Mobile OS, Types: Mainframe, Server, Embedded, Real-Time, IoT, Smart Card OS, System Calls, Operating System Structures (Monolithic, Microkernel, Client-Server, VM, Exokernel), OS Research and Modern Trends

Unit II – Process and Thread Management (9 Periods)

Process Concept, States, PCB, Process Creation & Termination, Process Scheduling: Preemptive and Non-Preemptive Scheduling Algorithms, Threads: User vs Kernel Threads, Multithreading, POSIX Threads, Synchronization: Semaphores, Monitors, Mutexes, Classical Problems: Producer-Consumer, Readers-Writers, Dining Philosophers

Unit III – Memory Management (9 Periods)

Contiguous and Non-contiguous Allocation, Paging, Segmentation, and Virtual Memory Concepts, Page Table Structures, TLB, and Page Replacement Algorithms (FIFO, LRU, Optimal, WSClock), Thrashing and Working Set Model, Linux Memory Management Overview

Unit IV – File Systems and I/O Management (9 Periods)

File Concept: Structure, Access Methods, and Attributes, Directory Structure and File-System Implementation, Disk-Space Management, Disk Scheduling (FCFS, SSTF, SCAN, C-SCAN) I/O System: Device Controllers, Drivers, DMA, Interrupts, Case Study: UNIX File System, Linux I/O Handling

Unit V – Deadlocks, Virtualization, and Security (9 Periods)

Deadlocks: Conditions, Detection, Prevention, and Avoidance (Banker's Algorithm),Multiprocessor and Distributed Operating Systems, Virtualization: Hypervisors, Containers, Cloud OS Concepts, Security Principles: Authentication, Access Control, Vulnerabilities, OS Hardening, Case Studies: Linux, Android, and Windows 11 Internals

Text Books:

1. Andrew S. Tanenbaum & Herbert Bos, *Modern Operating Systems*, 5th Ed., Pearson, 2023.
2. Abraham Silberschatz, Peter Baer Galvin, Greg Gagne, *Operating System Concepts*, 10th Ed., Wiley.
3. William Stallings, *Operating Systems: Internals and Design Principles*, 9th Ed., Pearson.
4. Daniel P. Bovet & Marco Cesati, *Understanding the Linux Kernel*, O'Reilly.

SEMESTER-III
COURSE 6: Operating Systems

Practical **Credits: 1** **2 hrs/week**

List of Experiments:

Exp. No.	Title of Experiment	Objective / Description
1	Introduction to Linux Shell and System Calls	Explore file operations, user management, and system calls using `open()`, `read()`, `write()`, `fork()`.
2	Process Creation using `fork()`	Demonstrate parent-child process relationship and process IDs.
3	Implementation of Scheduling Algorithms	Simulate FCFS, SJF, Priority, and Round Robin scheduling in C.
4	Inter-Process Communication	Implement message passing and shared memory communication.
5	Producer-Consumer Problem	Implement synchronization using semaphores or mutex locks.
6	Reader-Writer Problem	Demonstrate readers–writers synchronization using semaphores.
7	Memory Management Simulation	Simulate paging and page replacement algorithms (FIFO, LRU).
8	File System Commands and Manipulation	Implement file creation, deletion, and directory traversal using shell commands and C programs.
9	Disk Scheduling Algorithms	Implement disk scheduling algorithms (FCFS, SSTF, SCAN, C-SCAN).
10	Deadlock Detection and Avoidance	Simulate resource allocation and Banker's algorithm in C.
11	Virtualization Demonstration	Demonstrate VirtualBox or Docker container setup and basic OS virtualization concepts.
12	Mini Project	Combine process, memory, or file management concepts into a single integrated OS simulation.

SEMESTER-III

COURSE 7: Foundations of Quantum Technologies

Theory **Credits: 3** **3 hrs/week**

Course Objectives:

1. To introduce the foundational principles of quantum mechanics relevant to computing.
2. To understand various architectures and technologies used in quantum computers.
3. To explore core quantum phenomena such as superposition, entanglement, and quantum gates.
4. To gain practical exposure to Qiskit and IBM Q for simulating and implementing quantum algorithms.
5. To analyze quantum algorithms, communication, and applications across industries.

Course Learning Outcomes (CLOs):

After successful completion, learners will be able to:

1. Explain the fundamental quantum mechanical principles applied in computing.
2. Compare and contrast classical and quantum computing paradigms.
3. Design and implement quantum gates and circuits using Qiskit.
4. Apply quantum communication and error correction techniques for secure systems.
5. Evaluate the potential of quantum algorithms and their industrial applications.

Unit I – Principles of Quantum Computing (9 Periods)

Fundamentals of quantum behavior, Subatomic particles and quantum ensemble, Wave-particle duality and uncertainty principle, Quantum entanglement and synchronization Quantum information, teleportation, and technologies, Quantum field theory, Schrödinger's wave mechanics, and interpretations (Copenhagen, Many-Worlds)

Unit II – Quantum Computers and Quantum States (9 Periods)

Types of quantum computers: D-Wave, IonQ, Honeywell, Microsoft, Xanadu, Rigetti, Superconducting qubits and cryogenic cooling, Superposition and entanglement concepts, Mathematical representation of quantum states, Decoherence, fault tolerance, and scalability issues

Unit III – Quantum Gates, Circuits, and Qiskit (9 Periods)

Unitary matrices and Bloch sphere, Basic quantum gates: X, Y, Z, H, S, T, CNOT, SWAP Building and simulating circuits using Qiskit and IBM Q, Quantum Composer and Quantum Lab, Grover's search and Oracle in Qiskit

Unit IV – Quantum Communication and Error Correction (9 Periods)

Quantum superdense coding (Message 00–11), Quantum teleportation protocols, Quantum key distribution (QKD) and post-quantum cryptography, Quantum error types and mitigation, Shor's Code and 3-qubit bit-flip/phase-flip correction

Unit V – Quantum Algorithms and Industrial Applications (9 Periods)

Quantum algorithms: Phase Kickback, Grover's, Shor's, Deutsch-Jozsa, Bernstein–Vazirani, Quantum computing in cryptography, finance, AI, pharmaceuticals, and materials science, Challenges in post-quantum cryptography and standardization, Industry case studies: IBM, Google, Microsoft, and Atom Computing

Text Books:

1. **Sudeep Satheesan & Sri Mounica Kalidasu (2025).** *Quantum Computing: Concepts, Fundamentals, Circuits, and Code.* BPB Publications, India.

SEMESTER-III

COURSE 7: Foundations of Quantum Technologies

Practical **Credits: 1** **2 hrs/week**

List of Experiments:

Expt. No.	Title of the Experiment	Learning Objective	Expected Outcome
1	Introduction to IBM Quantum Lab and Qiskit Installation	Set up Python environment and install Qiskit library	Successfully configure Qiskit and connect to IBM Quantum backend
2	Representation of Qubits and Bloch Sphere Visualization	Understand superposition and visualize single-qubit states	Visualize
3	Implementation of Basic Quantum Gates (X, Y, Z, H, S, T)	Study and apply single-qubit unitary gates	Create and simulate gates in Qiskit to observe state transformations
4	Construction of Two-Qubit Gates (CNOT, SWAP)	Demonstrate entanglement and controlled operations	Build simple two-qubit circuits and verify entangled output
5	Simulation of Superposition and Entanglement States	Explore concepts of quantum parallelism	Observe and measure Bell pair entanglement using Qiskit
6	Quantum Circuit Design using IBM Quantum Composer	Create, visualize, and simulate circuits graphically	Design quantum circuits for gate sequences and analyze output
7	Grover's Search Algorithm using Qiskit	Implement Grover's algorithm for searching unsorted data	Demonstrate quantum speedup over classical search
8	Deutsch–Jozsa Algorithm Implementation	Distinguish between constant and balanced functions	Implement circuit and verify results through simulation
9	Shor's Algorithm Demonstration (Simulation)	Understand integer factorization using quantum circuits	Simulate small-number factorization using modular arithmetic
10	Quantum Teleportation Protocol	Implement quantum state transfer using entanglement	Successfully teleport a qubit state across quantum nodes
11	Quantum Key Distribution (BB84 Protocol)	Explore secure communication using quantum mechanics	Implement and test a basic QKD communication setup
12	Quantum Error Detection and Correction (3-Qubit Code)	Apply bit-flip and phase-flip correction codes	Demonstrate error detection and correction on qubits
13	Design of Classical Logic Gates using Quantum Gates	Construct AND, OR, XOR using quantum equivalents	Simulate logic gates and verify classical truth tables
14	Implementation of Bernstein–Vazirani Algorithm	Learn to find hidden binary strings efficiently	Build circuit and verify output using Qiskit backend
15	Real Quantum Device Execution	Execute any quantum circuit on IBM's real hardware	Observe difference between simulator and actual device output

SEMESTER-IV

COURSE 8: Python for Quantum Computing

Theory **Credits: 3** **3 hrs/week**

Course Objectives:

1. To introduce students to Python-based quantum computing frameworks for algorithm simulation and experimentation.
2. To enable learners to use **Cirq** for quantum circuit creation, execution, and analysis.
3. To develop proficiency in **LazyQML** for hybrid quantum–classical machine learning model design.
4. To explore **quantum cryptographic techniques** and **quantum key distribution (QKD)** through Python libraries.
5. To equip students with the ability to build, test, and analyze secure quantum algorithms for real-world applications.

Course Outcomes:

After completing this course, students will be able to:

1. Implement quantum circuits using **Cirq** and visualize qubit operations.
2. Design and execute **quantum machine learning models** using LazyQML.
3. Demonstrate understanding of **quantum-safe cryptographic algorithms**.
4. Integrate classical and quantum computation techniques in Python.
5. Apply quantum libraries to solve practical problems in **quantum cyber security** and **secure communication**.

Unit I – Foundations of Quantum Programming in Python

Overview of quantum computing and its Python ecosystem, Installation and setup of quantum libraries (Cirq, LazyQML, Qiskit, QuCrypt, etc.), Qubit representation, gates, and measurement, Superposition, entanglement, and unitary operations in Python

Unit II – Quantum Circuit Design and Simulation with Cirq

Introduction to Cirq and Google's quantum framework, Creating and visualizing quantum circuits, Measurement outcomes and probability distributions, Quantum teleportation and Grover's algorithm in Cirq, Integration with TensorFlow Quantum for hybrid models

Unit III – Quantum Machine Learning using LazyQML

Overview of hybrid quantum-classical ML models, Installation and structure of LazyQML, Quantum feature maps, kernels, and parameterized circuits, Training quantum neural networks and classifiers, Comparison: LazyQML vs PennyLane and Qiskit Machine Learning

Unit IV – Python Libraries for Quantum Cyber Security

Introduction to Quantum Cryptography and Quantum Key Distribution (QKD), Quantum-safe algorithms and post-quantum cryptography overview, Libraries: QuCrypt, PyQKD, and SimulaQron, Implementing BB84 and E91 protocols in Python, Quantum random number generation (QRNG) using Python

Unit V – Integrated Applications and Research Trends

Combining Cirq and LazyQML for quantum data protection, Quantum error correction and noise simulation libraries, Case studies: Quantum-enhanced cybersecurity systems, Current research in quantum-safe blockchain and authentication, Future trends in Python-based quantum development

Suggested Textbooks / References

1. **Google Cirq Documentation** – https://quantumai.google/cirq
2. **LazyQML Official Docs and Tutorials** – https://lazyqml.readthedocs.io
3. **Qiskit Textbook (IBM Quantum)** – “Learn Quantum Computation Using Qiskit” (2023 Edition)
4. K. Bharti et al., *"Quantum Machine Learning: Theory and Implementations"*, Springer, 2022.
5. M. Mosca, *"Quantum Cryptography and Cybersecurity"*, Cambridge University Press, 2023.

SEMESTER-IV

COURSE 8: Python for Quantum Computing

Practical **Credits: 1** **2 hrs/week**

List of Experiments:

Experiment No.	Title	Tools/Libraries
1	Implementing Basic Quantum Gates in Cirq	Cirq
2	Bell State Creation and Measurement	Cirq
3	Quantum Teleportation Protocol	Cirq
4	Quantum Classifier using LazyQML	LazyQML
5	Quantum Key Distribution (BB84) Simulation	PyQKD
6	Quantum Random Number Generator	QuCrypt
7	Quantum Noise Simulation and Error Correction	Cirq
8	Hybrid Quantum–Classical Neural Network	LazyQML + TensorFlow Quantum
9	Quantum Secure Chat using Python QKD	PyQKD
10	Capstone: Quantum Cybersecurity Demo	All Libraries

SEMESTER-IV

COURSE 9: Quantum Computing using QISKIT

Theory **Credits: 3** **3 hrs/week**

Course Objectives:

1. To introduce fundamental concepts of quantum computation and information theory.
2. To provide practical skills for quantum programming using Python and Qiskit.
3. To develop an understanding of quantum states, gates, and circuits.
4. To explore mathematical foundations such as vectors, matrices, and Hilbert spaces used in quantum systems.
5. To equip learners to simulate and execute basic quantum algorithms on quantum simulators or IBM Quantum devices.

Course Outcomes:

After successful completion, students will be able to:

1. Describe the structure of quantum computing systems and their difference from classical computers.
2. Write and execute quantum programs using Qiskit and Python.
3. Apply linear algebra concepts to understand qubit transformations and superposition.
4. Design and simulate quantum circuits for computation and measurement tasks.
5. Demonstrate practical implementations of simple quantum algorithms and interpret simulation results.

Unit I – Foundations of Quantum Computing (9 Periods)

Introduction to Quantum Computing and Qiskit textbook overview, Classical bits vs Qubits: representation and manipulation, The concept of superposition and entanglement, Understanding quantum states and measurement, Basic Python and Jupyter Notebooks for Quantum Programming, Installing and configuring Qiskit

Unit II – Programming with Qiskit (9 Periods)

QuantumCircuit, QuantumRegister, ClassicalRegister, Building and visualizing quantum circuits using Qiskit, Single-qubit and multi-qubit gates (X, Y, Z, H, CX, etc.), Measurement operations and simulation on QASM and Statevector simulators, Using Aer and IBMQ providers for quantum backend access

Unit III – Mathematical Foundations for Quantum Computing (9 Periods)

Basics of Linear Algebra: vectors, matrices, and tensor products, Vector spaces, linear dependence, basis, and dimension, Hermitian and Unitary matrices in quantum computation
Inner product, normalization, and orthogonality in Hilbert spaces, Eigenvalues and eigenvectors of quantum operators

Unit IV – Quantum Logic and Circuit Design (9 Periods)

Quantum gates as matrix operations, Quantum state transformations and unitary evolution, Circuit composition and custom gate creation in Qiskit, Measurement and classical control, Accessing and executing jobs on IBM Quantum hardware

Unit V – Quantum Algorithms and Applications (9 Periods)

Overview of basic quantum algorithms: Deutsch–Jozsa, Grover's, and Bernstein–Vazirani, Quantum teleportation and superdense coding, Quantum error and noise introduction, Introduction to Variational Quantum Circuits (VQC) and hybrid quantum-classical approaches
Future directions: Quantum machine learning and optimization

Textbook:

- *Learn Quantum Computing Using Qiskit* (Qiskit Textbook, IBM Quantum Community)

References:

1. Nielsen, M. A., & Chuang, I. L. (2010). *Quantum Computation and Quantum Information*.
2. Asfaw, A. et al. *Qiskit Textbook Online Resource*.
3. Benenti, G., Casati, G., & Strini, G. (2019). *Principles of Quantum Computation and Information*.
4. Qiskit Documentation – https://qiskit.org/documentation

SEMESTER-IV

COURSE 9: Quantum Computing using QISKIT

Practical **Credits: 1** **2 hrs/week**

List of Experiments:

Exp. No.	Title of Experiment	Learning Focus / Tools
1	**Setting up the Qiskit Environment** – Installing Anaconda, Python, and Qiskit; exploring Jupyter Notebook	Qiskit, Jupyter Notebook
2	**Introduction to Qiskit Circuits** – Create simple quantum circuits and visualize them using `qc.draw()`	QuantumCircuit, draw()
3	**Single-Qubit Gates Implementation** – Apply X, Y, Z, and H gates on qubits and analyze results on `statevector_simulator`	Quantum gates, statevector
4	**Two-Qubit Entanglement (Bell State)** – Implement H + CX to create an entangled pair and measure results	Entanglement, plot_histogram()
5	**Measurement and Probabilistic Results** – Measure multiple qubits and interpret output distributions	Measurement, QASM simulator
6	**Custom Quantum Gate Creation** – Build and append user-defined gates; observe their effect on quantum states	to_instruction(), append()
7	**Bloch Sphere Visualization** – Represent qubit states and transformations using Bloch vector plots	qiskit.visualization, matplotlib
8	**Linear Algebra Simulation** – Demonstrate vector and matrix operations using NumPy and apply to qubit states	NumPy, Matrix algebra
9	**Quantum Teleportation Protocol** – Implement teleportation using 3-qubit circuit	CNOT, CX, Measure, Conditional Gates
10	**Deutsch–Jozsa Algorithm** – Identify balanced or constant function using quantum speedup	Quantum Algorithms
11	**Grover's Search Algorithm** – Implement 2-qubit search problem using Oracle and Diffusion operators	Oracle design, Quantum search
12	**Running on Real Quantum Hardware** – Execute selected circuits on `ibmq_qasm_simulator` and IBM Quantum backend	IBMQ provider, real-device execution

SEMESTER-IV

COURSE 10: Quantum Computing Algorithms

Theory **Credits: 3** **3 hrs/week**

Course Objectives:

1. To introduce the principles and mathematical foundations underlying quantum algorithms.
2. To explore the design and implementation of core quantum algorithms such as Shor's and Grover's.
3. To apply quantum optimization and machine learning techniques for solving complex computational problems.
4. To provide practical exposure through simulation frameworks like Qiskit, Cirq, and PennyLane.

Course Learning Outcomes:

Upon successful completion of this course, students will be able to:

1. Explain the theoretical principles of major quantum algorithms.
2. Implement and simulate quantum algorithms using Python-based libraries.
3. Differentiate between classical and quantum optimization strategies.
4. Analyze the computational complexity and efficiency of quantum algorithms.
5. Apply quantum computing concepts to real-world domains such as cryptography, optimization, and AI.

Unit-I: Introduction to Quantum Computing Algorithms

Quantum computation fundamentals: qubits, superposition, and entanglement, Quantum operators and circuits overview, Quantum algorithm structure and complexity classes (BQP, BQNP), Classical vs Quantum approaches in data processing, Overview of Quantum Information Processing Framework

Unit-II: Foundational Quantum Algorithms

Deutsch–Jozsa algorithm: balanced vs constant functions, Simon's algorithm: period-finding and oracle problem, Shor's algorithm: integer factorization and quantum Fourier transform, Grover's search algorithm: unstructured search and amplitude amplification

Unit-III:Quantum Optimization Algorithms

Introduction to quantum optimization and combinatorial problems, Approximate optimization algorithms (QAOA, SDP), Quantum semidefinite programming, Quantum NP problems and quantum annealing

Unit-IV: Advanced Quantum Algorithms and Eigen Solvers

Quantum least squares fitting, Quantum eigenvalue and eigenvector solvers, Quantum semidefinite programming methods, Variational Quantum Eigensolver (VQE) concept

Unit-V: **Quantum AI and Hybrid Algorithms**

Quantum neural network and associative memory, Quantum deep learning and Boltzmann machine, Quantum search and pattern recognition, Hybrid quantum-classical models for machine learning

Text Books:

1. *Bhagvan Kommadi, "Quantum Computing Solutions: Real-World Algorithms," Apress, 2020.*
2. E. Rieffel and W. Polak, — **"Programming Quantum Computers: Essential Algorithms and Code Samples,"** O'Reilly Media, Sebastopol, CA, USA, 2020.
3. D. Wang and H. Wang, — **"Quantum Computing Algorithms,"** Packt Publishing, Birmingham, U.K., 2022.

Recommended Tools:

- IBM **Qiskit**, **Google Cirq**, **Microsoft Q#**, **PennyLane**, and **TensorFlow Quantum**

SEMESTER-IV

COURSE 10: Quantum Computing Algorithms

Practical **Credits: 1** **2 hrs/week**

List of Experiments:

Exp. No.	Title of Experiment	Description / Learning Focus	Tools / Framework
1	**Introduction to Quantum Circuits and Gates**	Construct basic circuits using Hadamard, Pauli-X/Y/Z, and CNOT gates. Observe superposition and entanglement.	Qiskit / Cirq
2	**Simulation of Deutsch–Jozsa Algorithm**	Implement the algorithm to distinguish between constant and balanced functions using quantum oracles.	Qiskit
3	**Implementation of Simon's Algorithm**	Find the hidden bit string using quantum parallelism. Demonstrate quantum advantage.	Qiskit / PennyLane
4	**Shor's Algorithm for Integer Factorization**	Perform modular exponentiation and use Quantum Fourier Transform (QFT) for factoring numbers.	Qiskit
5	**Grover's Search Algorithm**	Implement Grover's algorithm for unstructured database search. Compare with classical search complexity.	Cirq / Qiskit
6	**Quantum Approximate Optimization Algorithm (QAOA)**	Solve simple combinatorial optimization problems such as MaxCut or Traveling Salesman Problem.	PennyLane / Qiskit
7	**Variational Quantum Eigensolver (VQE)**	Use VQE to estimate the ground-state energy of a molecule or matrix Hamiltonian.	PennyLane / TensorFlow Quantum
8	**Quantum Neural Network (QNN)**	Construct a basic QNN with parameterized quantum circuits. Evaluate for simple pattern recognition tasks.	TensorFlow Quantum
9	**Quantum Support Vector Machine (QSVM)**	Implement a quantum kernel-based classifier for binary classification. Compare with classical SVM.	PennyLane / Qiskit
10	**Quantum Cryptography and Security Experiment**	Simulate Quantum Key Distribution (QKD) using BB84 protocol and analyze post-quantum security.	Qiskit / Microsoft Q#

SEMESTER-V

COURSE 11: Quantum Computing Applications

Theory **Credits: 3** **3 hrs/week**

Course Objectives:

1. To introduce the foundational principles and integration of quantum computing with emerging technologies.
2. To explore quantum computing applications across domains such as AI, IoT, cybersecurity, and smart infrastructure.
3. To analyze how quantum algorithms revolutionize optimization, cryptography, and data analytics.
4. To understand the role of quantum computing in healthcare, finance, and sustainable systems.
5. To develop problem-solving skills using quantum-inspired and hybrid computing approaches.

Course Outcomes:

After completing the course, learners will be able to:

1. Explain the key principles and frameworks that enable quantum computing applications.
2. Demonstrate understanding of how quantum computing enhances AI, IoT, and machine learning systems.
3. Apply quantum cryptography and optimization concepts in secure and intelligent systems.
4. Analyze the benefits and challenges of quantum computing in industrial, healthcare, and financial sectors.
5. Design application-oriented case studies integrating quantum computing with real-world technologies.

Unit I – Fundamentals and Integration Applications *(9 Periods)*

Introduction to quantum computing and quantum mechanics primer, Quantum gates, qubits, superposition, entanglement, and teleportation, Quantum computing architectures, processors, and software tools (Qiskit, Cirq), Integration applications: role of quantum computing in Industry 4.0 and Education 5.0 ecosystems.

Unit II – Quantum Computing in Artificial Intelligence *(9 Periods)*

Role of quantum computing in AI and deep learning, Quantum neural networks and quantum machine learning models, Quantum annealing and variational quantum circuits, Explainable AI and hybrid quantum-classical ML approaches.

Unit III – Quantum Computing for Cybersecurity and Cryptography *(9 Periods)*

Quantum cryptography and quantum key distribution (QKD), Quantum-safe and post-quantum encryption, Blockchain and distributed ledger integration with quantum systems, Quantum resilience for data privacy and digital trust.

Unit IV – Quantum Computing in FinTech and Healthcare *(9 Periods)*

FinTech applications: quantum-enhanced optimization and risk modeling, Quantum data analytics in financial forecasting, Quantum applications in precision medicine and drug discovery, AI-aided data analytics and IoT-driven healthcare ecosystems.

Unit V – Quantum Computing for Smart Infrastructure and Sustainability *(9 Periods)*

Smart grids, IoT-enabled energy systems, and renewable energy optimization, Smart city management: waste reduction, transportation, and disaster response using quantum models, Sustainable computing and green quantum technologies, Case studies: Quantum computing applications in smart logistics and climate modeling.

Reference Book:

1. Khang, A. (2023). *Applications and Principles of Quantum Computing.* IGI Global Press. DOI:10.4018/979-8-3693-1168-4

SEMESTER-V

COURSE 11: Quantum Computing Applications

Practical **Credits: 1** **2 hrs/week**

List of Experiments:

Expt. No.	Title of the Experiment	Objective / Description	Expected Learning Outcome
1	*Introduction to Qiskit and Quantum Circuit Design*	Install Qiskit; create and simulate basic quantum circuits (Hadamard, Pauli, CNOT gates).	Understand quantum gates, qubits, and Bloch sphere representation.
2	*Quantum Superposition and Entanglement Simulation*	Create two- and three-qubit systems using superposition and entanglement operators.	Visualize entanglement and verify measurement correlations.
3	*Implementation of Grover's Search Algorithm*	Simulate a quantum search algorithm and compare complexity with classical search.	Learn how quantum speedup works for unstructured data search.
4	*Quantum Machine Learning using Cirq/Qiskit ML*	Build a simple quantum classifier using Qiskit Machine Learning or TensorFlow Quantum.	Implement and analyze a hybrid quantum-classical machine learning model.
5	*Quantum Cryptography – QKD Protocol Simulation*	Demonstrate the BB84 protocol for secure quantum key distribution using Python simulator.	Learn principles of quantum key exchange and post-quantum cryptography.
6	*Quantum Optimization for Financial Data (QAOA)*	Use Quantum Approximate Optimization Algorithm for portfolio or risk optimization.	Apply quantum optimization to financial applications (FinTech).
7	*Quantum IoT Data Encryption and Communication*	Simulate a small IoT network where quantum cryptography enhances data security.	Explore how quantum technologies improve IoT security and performance.
8	*Quantum Smart City Simulation (Energy or Traffic Optimization)*	Model a quantum-assisted smart grid or traffic system optimization using hybrid models.	Understand how quantum computing supports sustainability and infrastructure optimization.

SEMESTER-V

COURSE 12A: Computer Networks

Theory **Credits: 3** **3 hrs/week**

Course Objectives:

1. To introduce the fundamental concepts and architecture of computer networks.
2. To understand various network models, protocols, and communication layers.
3. To study data transmission, error detection, and flow control mechanisms.
4. To familiarize students with network addressing, routing, and security concepts.
5. To provide hands-on knowledge of emerging networking technologies and applications.

Course Learning Outcomes:

After successful completion of this course, the students will be able to:

1. Explain the basic concepts, components, and architectures of computer networks.
2. Describe various network models (OSI and TCP/IP) and their layer functionalities.
3. Demonstrate knowledge of data transmission techniques, switching, and routing algorithms.
4. Analyze network performance and apply error detection and flow control methods.
5. Identify network security issues and understand modern technologies like IoT and 5G.

Unit I: Introduction to Computer Networks (9 Periods)

Definition and evolution of Computer Networks, Network Goals, Applications, and Topologies, Network Hardware and Software components, Types of Networks: LAN, MAN, WAN, PAN, Network Models: OSI Reference Model and TCP/IP Model comparison, Concept of Protocols, Interfaces, and Standards

Unit II: Data Transmission and Physical Layer (9 Periods)

Data and Signals: Analog and Digital transmission, Transmission Media: Guided (Twisted Pair, Coaxial, Fiber Optic) and Unguided Media, Switching Techniques: Circuit, Packet, and Message Switching, Multiplexing: FDM, TDM, WDM, Error Detection and Correction: Parity, CRC, Hamming Code, Performance Metrics: Bandwidth, Throughput, Latency

Unit III: Data Link Layer and MAC Sub-layer (9 Periods)

Framing and Flow Control: Stop and Wait, Sliding Window Protocols, Error Control Mechanisms: ARQ, Media Access Control: CSMA/CD, CSMA/CA, LAN Technologies: Ethernet, Token Ring, Wireless LAN (IEEE 802.11), VLANs and Concept of Switching (Layer 2 Switches)

Unit IV: Network and Transport Layers (9 Periods)

Logical Addressing: IPv4, IPv6 structure and addressing, Routing Concepts and Algorithms: Distance Vector, Link State, Internetworking Devices: Routers, Gateways, Firewalls, Transport Layer Services: TCP and UDP protocols, Congestion Control and Quality of Service (QoS)

Unit V: Application Layer and Network Security (9 Periods)

Application Layer Protocols: HTTP, FTP, SMTP, DNS, DHCP, SNMP, Socket Programming Basics, Network Security Principles: Cryptography, Authentication, Firewalls, VPN, Introduction to Cloud Networking, IoT Networking, and 5G Communication, Emerging Trends: SDN, Network Virtualization, and Edge Computing

Text Books:

1. Andrew S. Tanenbaum, *Computer Networks*, 5th Edition, Pearson Education.
2. Behrouz A. Forouzan, *Data Communications and Networking*, 5th Edition, McGraw Hill.
3. William Stallings, *Data and Computer Communications*, 10th Edition, Pearson.

Reference Books:

1. James F. Kurose & Keith W. Ross, *Computer Networking: A Top-Down Approach*, 7th Edition, Pearson.
2. Larry L. Peterson & Bruce S. Davie, *Computer Networks: A Systems Approach*, Morgan Kaufmann.
3. Natalia Olifer & Victor Olifer, *Computer Networks: Principles, Technologies and Protocols for Network Design*, Wiley.

SEMESTER-V
COURSE 12A: Computer Networks

Practical **Credits: 1** **2 hrs/week**

Expt. No.	Title of the Experiment	Tools / Software Used	Learning Outcome
1	Study of basic network components and topologies (bus, star, ring, mesh)	Cisco Packet Tracer / GNS3	Understand physical network setup and topological arrangements.
2	Simulation of data transmission using guided and unguided media	Packet Tracer / ns-3	Demonstrate basic data communication and media differences.
3	Configuration of IP addressing and subnetting in a LAN	Packet Tracer / Mininet	Learn IPv4 and IPv6 address configuration and subnet calculation.
4	Study and analysis of OSI and TCP/IP layer interactions using Wireshark	Wireshark	Identify headers and fields in Ethernet, IP, TCP, and UDP packets.
5	Implementation of error detection techniques: Parity check, CRC, Hamming code	Python / C program	Develop algorithms for error detection and correction.
6	Simulation of flow control protocols – Stop-and-Wait and Sliding Window	Python	Understand flow control mechanisms in Data Link Layer.
7	Study and configuration of MAC layer protocols (CSMA/CD and CSMA/CA)	ns-3 / Mininet	Analyze contention-based medium access control methods.
8	Configuration of a small LAN with multiple switches and routers	Cisco Packet Tracer	Understand LAN design, VLAN configuration, and inter-VLAN routing.
9	Implementation of static and dynamic routing protocols (RIP, OSPF)	Packet Tracer / GNS3	Study packet forwarding and routing algorithm behavior.
10	Socket Programming in Python – Client-Server chat application	Python 3	Apply TCP/UDP concepts to create communication applications.
11	Simulation of congestion control (TCP slow start, AIMD)	ns-3	Analyze congestion avoidance mechanisms in transport layer.
12	Packet capture and analysis of HTTP, FTP, DNS, and SMTP protocols	Wireshark	Understand application layer protocol headers and operations.
13	Demonstration of basic firewall and port filtering configuration	Ubuntu iptables / pfSense	Implement basic network security measures.

SEMESTER-V

COURSE 12B: Mathematics for Machine Learning

Theory **Credits: 3** **3 hrs/week**

Course Objectives:

By the end of the course, students will be able to:

1. Understand key mathematical foundations that underpin modern machine learning algorithms.
2. Apply concepts of linear algebra and vector calculus to model and analyze data transformations.
3. Utilize probability and statistics to handle uncertainty and make predictions in ML models.
4. Explore optimization techniques essential for training machine learning models.
5. Integrate mathematical reasoning to implement and evaluate machine learning algorithms.

Course Learning Outcomes:

After successful completion of this course, students will be able to:

1. Demonstrate an understanding of vector spaces, eigenvalues, and matrix operations in ML.
2. Apply differential calculus and gradient-based optimization methods in model training.
3. Use probability distributions and statistical inference for data-driven decision-making.
4. Formulate and solve optimization problems related to regression and classification tasks.
5. Mathematically interpret the internal working of machine learning algorithms such as PCA, SVM, and neural networks.

Unit I: Linear Algebra for Machine Learning (10 Periods)

Vectors, matrices, and tensor notation, Linear transformations and matrix operations, Determinants, rank, inverse, and orthogonality, Eigenvalues and eigenvectors, Applications: Dimensionality reduction (PCA), covariance matrix.

Unit II: Multivariable Calculus and Optimization (9 Periods)

Functions of several variables, limits, and continuity, Partial derivatives, gradient, Jacobian, Hessian, Taylor series expansion and approximation, Gradient descent, stochastic gradient descent, Convex functions and constrained optimization

Unit III: Probability and Random Variables (9 Periods)

Basics of probability theory: conditional probability, Bayes theorem, Random variables, PMF, PDF, CDF, Expectation, variance, covariance, correlation, Common distributions: Bernoulli, Binomial, Gaussian, Exponential, Law of large numbers, central limit theorem

Unit IV: Statistics and Data Analysis (9 Periods)

Sampling, estimation, and hypothesis testing, Confidence intervals, p-values, chi-square test Correlation and regression analysis, Maximum likelihood estimation (MLE), Feature scaling and normalization techniques

Unit V: Advanced Mathematical Tools for Machine Learning (8 Periods)

Vector spaces and inner product spaces, Singular Value Decomposition (SVD) and Matrix Factorization, Information theory: Entropy, cross-entropy, KL divergence, Fourier and Laplace transforms in ML signal processing. Numerical methods for large-scale ML computations

Recommended Textbooks:

1. Deisenroth, M. P., Faisal, A. A., & Ong, C. S. (2020). *Mathematics for Machine Learning.* Cambridge University Press.
2. Bishop, C. M. (2006). *Pattern Recognition and Machine Learning.* Springer.
3. Strang, G. (2019). *Linear Algebra and Learning from Data.* Wellesley-Cambridge Press.
4. Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning.* MIT Press.
5. Ross, S. M. (2014). *Introduction to Probability and Statistics for Engineers and Scientists.* Academic Press.

SEMESTER-V

COURSE 12B: Mathematics for Machine Learning

Practical **Credits: 1** **2 hrs/week**

List of Practicals

1. Matrix operations using NumPy and visualization of transformations
2. Implementation of PCA for dimensionality reduction
3. Visualization of gradient descent optimization
4. Probability distribution fitting using real-world datasets
5. Naïve Bayes classifier on categorical data
6. Linear regression using least squares and gradient descent
7. Logistic regression with sigmoid function and cost minimization
8. Implementation of MLE on sample datasets
9. Correlation and covariance analysis
10. Entropy and information gain computation for decision trees
11. SVD and matrix factorization on recommendation data
12. Visualization of loss functions and convergence curves
13. Comparison of optimization algorithms (SGD, Adam, RMSProp)
14. Monte Carlo simulation for probabilistic estimation
15. Implementation of basic neural network mathematics manually

SEMESTER-V

COURSE 13A: Classical Cryptography & Network Security

Theory **Credits: 3** **3 hrs/week**

Course Objectives:

1. To introduce the evolution, concepts, and principles of classical and modern cryptography.
2. To explain symmetric and asymmetric cryptographic techniques for securing data.
3. To analyze various network models, protocols, and cryptographic mechanisms in network security.
4. To study web and email security protocols such as TLS, SSL, IPsec, and PGP.
5. To develop the ability to design secure communication systems resistant to common attacks.

Course Outcomes:

After completion of the course, students will be able to:

1. Demonstrate understanding of classical ciphers and their cryptanalytic techniques.
2. Compare and contrast symmetric and asymmetric encryption algorithms.
3. Explain the design and function of network security architectures and protocols.
4. Implement cryptographic algorithms to achieve confidentiality, integrity, and authentication.
5. Apply security mechanisms in web, email, and VPN communications effectively.

Unit I – Fundamentals of Cryptography and Classical Systems (9 Periods)

Introduction to Cryptography and Security Goals: Confidentiality, Integrity, Authenticity, Non-repudiation, Cryptographic Terminologies: Plain text, Cipher text, Encryption, Decryption, Key, Keystream, **Classical Cryptography**: Substitution and Transposition Ciphers – Caesar, Vigenère, Playfair, Hill Cipher.Enigma Machine and Zimmerman Telegram case study. **Cryptanalysis**: Frequency analysis, brute-force attacks, limitations of classical ciphers.

Unit II – Symmetric Cryptography (9 Periods)

Concept of Symmetric Key Cryptography, Stream Ciphers vs Block Ciphers, Algorithms: DES, Triple DES, IDEA, AES – Architecture and Operation, Modes of Operation: ECB, CBC, CFB, OFB, CTR, Key Management and Distribution Issues.

Unit III – Asymmetric Cryptography and Hash Functions (9 Periods)

Principles of Public Key Cryptography, RSA Algorithm: Key generation, encryption, and decryption, Diffie-Hellman Key Exchange, Elliptic Curve Cryptography (ECC) basics, Message Authentication Codes (MAC), Digital Signatures, and Hash Functions (SHA-1, SHA-2, SHA-3).

Unit IV – Network Security Models and Protocols (9 Periods)

OSI & TCP/IP Models – Structure, layers, and encapsulation/decapsulation process, Security services at different layers, Network Security Components: Firewalls, IDS/IPS, VPN, NAT, and Proxy Servers, Web Security Protocols: SSL, TLS (v1.2 & v1.3), HTTPS – Operation and Handshake Process.

Unit V – Applied Security: VPN, Email, and Web Applications (9 Periods)

IPsec and its modes of operation – AH, ESP, and IPComp, Virtual Private Networks (VPN): Remote access and site-to-site VPNs, Email Security: PGP, S/MIME, MIME, Web Application Security and OWASP Overview.

Textbook:

Sandip Dholakia (2025). *Modern Cryptography: Securing Data.* Rheinwerk Publishing Inc., Boston (MA).

Suggested Reference Books:

1. William Stallings – *Cryptography and Network Security: Principles and Practice,* Pearson.
2. Behrouz A. Forouzan – *Cryptography and Network Security,* McGraw Hill.
3. Christof Paar & Jan Pelzl – *Understanding Cryptography: From Established Symmetric and Asymmetric Ciphers to Post-Quantum Algorithms,* Springer, 2024.

SEMESTER-V

COURSE 13A: Classical Cryptography & Network Security

Practical **Credits: 1** **2 hrs/week**

Exp. No.	Title of Experiment	Objective / Description	Tools / Software Used	Expected Outcome
1	Implementation of Substitution Ciphers	Implement **Caesar** and **Vigenère** cipher algorithms for encryption and decryption.	Python / CrypTool 2	Understand substitution encryption and key dependency.
2	Implementation of Transposition Ciphers	Implement **Rail Fence** and **Columnar Transposition** ciphers.	Python / CrypTool 2	Differentiate substitution and transposition techniques.
3	Cryptanalysis of Classical Ciphers	Perform **frequency analysis** to break Caesar and Monoalphabetic ciphers.	Python / CrypTool 2	Learn basic code-breaking and cryptanalysis concepts.
4	Simulation of DES Algorithm	Encrypt and decrypt data using the **DES algorithm**.	Python (PyCryptoDome) / CrypTool	Understand Feistel rounds and key scheduling.
5	Implementation of AES Algorithm	Perform **AES-128 and AES-256** encryption on text/files.	Python / OpenSSL	Study modern symmetric encryption structures.
6	Demonstration of Block Cipher Modes	Compare **ECB, CBC, CFB, OFB, CTR** modes of operation.	Python / CrypTool 2	Analyze diffusion and error propagation effects.
7	Implementation of RSA Algorithm	Generate keys, encrypt and decrypt data using **RSA**.	Python / OpenSSL	Learn public-key encryption mechanism.
8	Diffie–Hellman Key Exchange	Simulate secure key exchange between two users.	Python / Manual Calculation	Understand shared secret generation in insecure channels.
9	Hash Functions and Message Authentication	Compute **MD5, SHA-1, SHA-256, SHA-3** hashes; verify integrity with **HMAC**.	Python / HashCalc	Learn hashing and authentication in data protection.
10	Analyze HTTPS using Wireshark	Capture HTTPS packets; analyze **TLS/SSL handshake** and cipher suites.	Wireshark	Understand secure data transmission protocols.

SEMESTER-V

COURSE 13B: Python for Machine Learning

Theory **Credits: 3** **3 hrs/week**

Course Objectives:

1. To introduce the principles and workflow of machine learning using Python and Scikit-Learn.
2. To understand supervised and unsupervised learning algorithms and their mathematical foundations.
3. To apply model selection, feature engineering, and evaluation techniques for predictive analytics.
4. To develop the ability to preprocess and visualize datasets for real-world problem-solving.
5. To implement and optimize ML models using Scikit-Learn for practical and research applications.

Course Outcomes:

After successful completion of the course, students will be able to:

1. Demonstrate understanding of machine learning fundamentals and workflows using Scikit-Learn.
2. Apply classification, regression, and clustering algorithms to real datasets.
3. Evaluate model performance using appropriate metrics and validation techniques.
4. Design preprocessing and feature engineering pipelines for ML projects.
5. Build, tune, and deploy predictive models for real-world applications.

Unit I:

Introduction to Machine Learning and Scikit-Learn

Definition of ML – Applications – ML Pipeline – Overview of Scikit-Learn architecture – Installing and using Scikit-Learn – Data loading, preprocessing, and splitting – Feature scaling – Data visualization using Matplotlib and Seaborn.

Unit II:

Supervised Learning Algorithms

Linear Regression – Polynomial Regression – Logistic Regression – Decision Trees – Random Forests – K-Nearest Neighbors – Support Vector Machines – Performance metrics (accuracy, precision, recall, F1-score).

Unit III:

Unsupervised Learning and Dimensionality Reduction

K-Means Clustering – Hierarchical Clustering – DBSCAN – PCA – t-SNE – Evaluation of clustering results – Applications of unsupervised learning in anomaly detection and customer segmentation.

Unit IV:

Model Selection and Optimization

Train-test split – Cross-validation – Bias-variance tradeoff – Hyperparameter tuning (GridSearchCV, RandomizedSearchCV) – Regularization (L1, L2) – Ensemble methods (Bagging, Boosting, AdaBoost, Gradient Boosting).

Unit V:

Practical Applications and Deployment

Building complete ML projects – Feature engineering pipelines – Saving and loading models (joblib, pickle) – Case studies (house price prediction, image classification, text sentiment analysis) – Model deployment basics (Flask, Streamlit).

Text Books:

1. *Machine Learning with Scikit-Learn* by Parag Saxena, 2024, ISBN: 978-8197223945
2. *Python Data Science Cookbook (2025)* by Taryn Voska

SEMESTER-V

COURSE 13B: Python for Machine Learning

Practical **Credits: 1** **2 hrs/week**

Exp. No.	Title of Experiment	Objective / Learning Outcome
1	Installation and setup of Python, Scikit-Learn, Pandas, NumPy, and Matplotlib	To familiarize students with the ML programming environment and verify successful installation.
2	Loading and preprocessing datasets using Scikit-Learn	To learn dataset loading, cleaning, encoding categorical data, and handling missing values.
3	Data visualization and feature correlation analysis using Seaborn	To visualize dataset distributions and relationships between variables.
4	Implementation of Linear Regression using Scikit-Learn	To build and evaluate a simple regression model and interpret model coefficients.
5	Polynomial Regression for non-linear data fitting	To demonstrate curve fitting using polynomial regression.
6	Logistic Regression for binary classification	To classify data into binary categories and evaluate accuracy and confusion matrix.
7	Decision Tree Classifier on sample dataset	To understand tree-based learning and interpret decision boundaries.
8	Random Forest and feature importance analysis	To implement ensemble learning and interpret feature importance rankings.
9	K-Nearest Neighbors (KNN) Classification	To explore distance-based classification and parameter tuning (K-value).
10	K-Means Clustering and visualization	To perform unsupervised learning and visualize clusters using PCA plots.
11	Principal Component Analysis (PCA) for dimensionality reduction	To reduce high-dimensional data and visualize principal components.
12	Hyperparameter tuning using GridSearchCV	To optimize model performance through systematic parameter search.
13	Model evaluation using confusion matrix, precision, recall, F1-score, and ROC curve	To compare model metrics for better understanding of classification performance.
14	Saving and loading trained models using joblib/pickle	To demonstrate model persistence and reusability.
15	Mini Project: Build and deploy an ML model using Streamlit	To design, train, evaluate, and deploy a real-world predictive model as a mini project.

SEMESTER-V

COURSE 14A: Quantum Cryptography Protocols

Theory **Credits: 3** **3 hrs/week**

Course Objectives:

1. To introduce the fundamental principles of quantum mechanics as applied to information security.
2. To understand quantum key distribution (QKD) and its role in secure communication.
3. To explore various quantum cryptographic protocols and their implementation mechanisms.
4. To analyze the security advantages and challenges of quantum protocols over classical systems.
5. To study post-quantum cryptographic approaches and the integration of quantum and classical systems.

Course Outcomes:

After successful completion of this course, students will be able to:

1. Explain the principles of quantum mechanics relevant to cryptography.
2. Describe and compare different quantum key distribution (QKD) protocols such as BB84, B92, and E91.
3. Analyze the security and performance of quantum cryptographic systems.
4. Implement basic simulations of quantum protocols using open-source quantum computing frameworks.
5. Evaluate hybrid and post-quantum cryptographic schemes for secure communication.

Unit I:

Introduction to Quantum Cryptography

Overview of Classical vs Quantum Cryptography, Basic Postulates of Quantum Mechanics, Quantum Bits and Superposition, Quantum Entanglement and Measurement Principles, Quantum No-Cloning Theorem and Its Implications

Unit II:

Quantum Key Distribution (QKD)

Concept and Need for QKD, BB84 Protocol – Theory and Steps, B92 Protocol – Simplified QKD, E91 Entanglement-Based QKD Protocol, Security Analysis and Attack Models (Intercept-Resend, Photon Number Splitting, etc.)

Unit III:

Advanced Quantum Cryptographic Protocols

Quantum Secret Sharing (QSS), Quantum Digital Signatures (QDS), Quantum Bit Commitment and Oblivious Transfer, Quantum Authentication and Quantum Teleportation in Security, Device-Independent Quantum Cryptography

Unit IV:

Implementation & Simulation of Quantum Protocols

Quantum Cryptography Hardware Components (Photon Sources, Detectors, Polarizers), Simulation Tools: Qiskit, QuTiP, and IBM Quantum Experience, Hands-on: Simulating BB84 and E91 Protocols Error Correction and Privacy Amplification Techniques

Unit V:

Future Directions and Post-Quantum Security

Post-Quantum Cryptography vs Quantum Cryptography, Lattice-based, Code-based, and Multivariate Cryptosystems, Quantum Network Security and Quantum Internet, Standardization Efforts and NIST PQC Initiatives, Research Trends and Ethical Considerations in Quantum Security

Textbooks:

1. Nielsen, M.A. & Chuang, I.L. – *Quantum Computation and Quantum Information*, Cambridge University Press.
2. Imre, S. & Balazs, F. – *Quantum Computing and Communications: An Engineering Approach*, Wiley.

3. Gisin, N. et al. – *Quantum Cryptography*, Reviews of Modern Physics.
4. Menezes, A. – *Handbook of Applied Cryptography*, CRC Press.

Online Tools & Platforms:

- IBM Quantum Experience (Qiskit)
- Microsoft Quantum Development Kit (Q#)
- QuTiP (Quantum Toolbox in Python)

SEMESTER-V

COURSE 14A: Quantum Cryptography Protocols

Practical **Credits: 1** **2 hrs/week**

Exp. No.	Title of Experiment	Objectives / Learning Outcomes	Tools / Platform
1	Introduction to Quantum Computing Environment	To set up Qiskit / QuTiP environment and understand quantum circuit basics.	IBM Quantum Experience, Qiskit
2	Implementation of Qubits and Quantum Gates	To implement quantum states, apply gates (X, H, Z), and observe superposition effects.	Qiskit Notebook
3	Simulation of Qubit Measurement and No-Cloning Theorem	To understand measurement effects and verify the no-cloning principle.	Qiskit / QuTiP
4	Implementation of BB84 Quantum Key Distribution Protocol	To simulate secure key generation using polarization states in BB84.	Qiskit / Python
5	Analysis of Intercept-Resend Attack on BB84	To simulate an eavesdropper and measure quantum bit error rate (QBER).	Qiskit / Python
6	Implementation of B92 QKD Protocol	To perform simplified two-state QKD and measure key efficiency.	Qiskit / QuTiP
7	Simulation of E91 Entanglement-Based QKD	To implement E91 protocol and study entanglement correlations.	Qiskit / IBM Quantum Experience
8	Visualization of Quantum Entanglement and Bell's Inequality	To verify quantum non-locality using Bell test simulation.	Qiskit / Python
9	Quantum Secret Sharing Protocol	To distribute information securely among multiple parties using GHZ states.	Qiskit / Python
10	Quantum Bit Commitment and Quantum Digital Signature	To understand and simulate quantum bit commitment scheme and signature verification.	Qiskit / QuTiP
11	Quantum Teleportation and Authentication Protocol	To implement quantum teleportation and analyze its security aspects.	Qiskit / Python

SEMESTER-V

COURSE 14B: Deep Learning Fundamentals

Theory **Credits: 3** **3 hrs/week**

Course Objectives:

1. To understand the theoretical foundations and mathematics underlying deep learning algorithms.
2. To explore neural network architectures and their optimization techniques.
3. To apply convolutional and recurrent neural networks to real-world data.
4. To understand the role of autoencoders, generative models, and transfer learning.
5. To develop practical deep learning models using open-source frameworks such as TensorFlow or PyTorch.

Course Outcomes:

After successful completion of this course, students will be able to:

1. Explain the fundamental principles of neural networks and deep learning.
2. Implement and train deep neural networks for various data modalities.
3. Analyze and optimize network performance using gradient-based methods.
4. Apply advanced architectures such as CNNs, RNNs, and GANs to solve complex problems.
5. Design, evaluate, and deploy end-to-end deep learning systems for practical applications.

Unit I: Introduction to Deep Learning and Neural Networks

Introduction to AI and Machine Learning – Motivation for Deep Learning – Biological Neurons and Artificial Neurons – Perceptrons and Multilayer Perceptrons – Activation Functions – Loss Functions – Backpropagation and Gradient Descent

Unit II: Optimization and Training Deep Networks

Challenges in Training Deep Networks – Vanishing and Exploding Gradients – Regularization (Dropout, Batch Normalization) – Weight Initialization – Optimizers (SGD, Adam, RMSProp) – Hyperparameter Tuning and Model Evaluation

Unit III: Convolutional Neural Networks (CNNs)

Convolution and Pooling Operations – CNN Architectures (LeNet, AlexNet, VGG, ResNet, Inception) – Transfer Learning – Visualization and Interpretation – Applications in Image Recognition and Computer Vision

Unit IV: Sequential Models and Recurrent Neural Networks (RNNs)

Sequence Modeling – RNNs and LSTMs – GRUs – Attention Mechanisms – Transformers Overview – Applications in Natural Language Processing (Text, Speech)

Unit V: Generative and Advanced Deep Learning Models

Autoencoders – Variational Autoencoders (VAEs) – Generative Adversarial Networks (GANs) – Reinforcement Learning Basics – Ethical and Societal Implications of Deep Learning

Text Books:

1. *Fundamentals of Deep Learning (2nd Edition)* by Nikhil Buduma, Nithin Buduma & Joe Papa, O'Reilly Media
2. Programming Neural Networks with Python" by Joachim Steinwender and Roland Schwaiger (Rheinwerk Computing).

SEMESTER-V

COURSE 14B: Deep Learning Fundamentals

Practical **Credits: 1** **2 hrs/week**

No.	Practical Title	Objective	Key Concepts / Tools
1	Implement a Perceptron from Scratch	Understand the basic neuron model and activation functions	NumPy, dot product, step function
2	Train a Simple Neural Network for AND/OR Gates	Learn supervised learning and error correction	Feedforward, weight updates, loss function
3	Handwritten Digit Recognition using MNIST	Implement your first deep neural network	Keras/TensorFlow, softmax, dense layers
4	Visualize Activation Functions	Compare sigmoid, tanh, and ReLU behaviors	Matplotlib visualization, activation study
5	Build a Multi-Layer Perceptron (MLP) for Classification	Introduce hidden layers and backpropagation	Dense networks, batch training
6	Implement Gradient Descent from Scratch	Learn the optimization concept manually	Mean squared error, learning rate tuning
7	Image Classification with Convolutional Neural Networks	Learn CNN architecture and filters	Conv2D, MaxPooling, Flatten layers
8	Train a Neural Network with Dropout Regularization	Prevent overfitting and improve generalization	Dropout layers, training/testing accuracy
9	Predict House Prices Using a Regression Neural Network	Apply deep learning to regression problems	Normalization, linear output layer
10	Implement a Basic Reinforcement Learning Agent	Understand neural networks in decision-making	Q-learning, reward function, exploration

SEMESTER-V

COURSE 15A: Quantum Simulation in Cyber Security

Theory **Credits: 3** **3 hrs/week**

Course Objectives:

1. To explore practical quantum simulation frameworks in cybersecurity contexts.
2. To develop skills for simulating cyber-attacks, network defenses, and cryptographic systems using quantum tools.
3. To understand and simulate hybrid quantum–classical architectures for security.
4. To implement quantum machine learning (QML) and blockchain-based secure systems.
5. To examine real-world applications, standards, and future research in quantum-secure infrastructures.

Course Outcomes:

After completing this course, learners will be able to:

- **CO1:** Apply quantum simulators for modeling cyber-attack and defense scenarios.
- **CO2:** Design quantum-safe and hybrid cybersecurity architectures.
- **CO3:** Implement simulation-based approaches for secure communication and blockchain systems.
- **CO4:** Utilize quantum machine learning for intrusion and anomaly detection.
- **CO5:** Critically evaluate current research, ethical issues, and future trends in quantum cybersecurity.

Unit I: Quantum Simulation Frameworks for Cybersecurity

Architecture of quantum simulators for cybersecurity research, Quantum circuit emulation, noise modeling, and gate fidelity in security simulation, Frameworks and platforms: IBM Qiskit Aer, Cirq, QuTiP, Xanadu PennyLane, Workflow for simulating quantum algorithms in cybersecurity

Unit II: Simulation of Cyber Attacks and Quantum Defense Mechanisms

Simulation of brute-force and Grover-based quantum attacks, Modeling RSA and AES vulnerabilities under quantum threats, Quantum-resistant protocols and defensive simulations, Quantum replay and Man-in-the-Middle (MITM) simulation in communication networks, Performance metrics: simulation accuracy, execution time, and qubit resources.

Unit III: Quantum-Safe Network and Cloud Security Simulation |

Designing quantum-resilient network architectures, Quantum-safe cloud computing and access control simulation, Secure key exchange and data integrity validation using quantum protocols, Simulation of IoT and industrial control systems security (SCADA, IIoT), Hybrid classical–quantum security system simulation and benchmarking.

Unit IV: Quantum Machine Learning and Blockchain for Cybersecurity

Quantum Machine Learning (QML) fundamentals and types (QNN, QSVM, QGAN), Simulation of QML models for intrusion detection and anomaly analysis,Quantum Blockchain: principles, consensus mechanisms, and cryptographic integration,Simulation of secure quantum-ledger transactions and decentralized systems,Comparative study: classical vs quantum blockchain performance.

Unit V: Emerging Research Trends, Standards, and Ethics in Quantum Cybersecurity

Quantum Internet and Quantum Cloud Security initiatives, Standardization efforts: NIST PQC standards, ETSI quantum-safe working group, Future research: quantum-secure infrastructure, post-quantum transition planning, Case studies on government and industry quantum cybersecurity projects (e.g., IBM, Google, ISRO), Ethical, legal, and societal impacts of quantum cybersecurity research.

Textbooks & References:

1. **S. B. Goyal et al.**, *Quantum Computing, Cyber Security and Cryptography: Issues, Technologies, Algorithms, Programming and Strategies*, 2025.
2. **A. Joshi**, *Quantum Computing Models for Cybersecurity and Communications*, O'Reilly, 2025.

3. **IBM Quantum Team**, *Qiskit Textbook (2025 Update): Simulation of Quantum Systems for Cybersecurity*.
4. **Narayanan, A.**, *Post-Quantum Cryptography and Secure Simulation Frameworks*, Springer, 2025.
5. Recent papers (2023–2025) from IEEE, Springer, and Elsevier on **Quantum Simulation and Cybersecurity Applications**.

SEMESTER-V

COURSE 15A: Quantum Simulation in Cyber Security

Practical **Credits: 1** **2 hrs/week**

Exp. No.	Title of Experiment	Objective / Description	Tools / Software Used	Expected Outcome	Mapped COs
1	**Setting up the Quantum Simulation Environment & Basic Qubit Operations**	Install Qiskit/PennyLane, create and measure qubit states, visualize Bloch sphere.	Qiskit, Python, Jupyter Notebook	Students can prepare and measure qubit states, interpret Bloch sphere and measurement probabilities.	CO1, CO2
2	**Quantum Gates & Noise Simulation**	Implement multiple gates and simulate noise models (depolarizing, amplitude damping) to observe fidelity degradation.	Qiskit Aer, Cirq	Understanding of gate operations, noise impact, and error simulation in quantum systems.	CO1, CO2
3	**Simulation of Shor's Algorithm for Integer Factorization**	Simulate simplified Shor's algorithm to factor small integers and analyze implications on RSA security.	Qiskit Algorithms, Python	Demonstrate factorization and understand resource requirements and quantum limits.	CO2, CO3
4	**Grover's Search Simulation – Quantum Brute-Force Attack**	Implement Grover's algorithm to simulate a quantum brute-force attack and compare with classical search.	Qiskit / Cirq	Visualize quadratic speedup and evaluate attack feasibility on classical encryption.	CO2, CO3
5	**Quantum Key Distribution (BB84) Protocol Simulation**	Simulate QKD between Alice and Bob; introduce an eavesdropper and detect via QBER.	Qiskit, Python	Simulate secure key exchange, compute QBER, and detect intrusion.	CO2, CO3
6	**Quantum Random Number Generation (QRNG)**	Generate random keys using quantum measurements and evaluate statistical randomness.	Qiskit, Python (NIST randomness tests)	Demonstrate QRNG and validate randomness for cryptographic use.	CO2, CO4
7	**Simulation of Quantum-Safe Hybrid Network (PQC + QKD)**	Implement hybrid key exchange (post-quantum + quantum) for secure network handshake simulation.	Python, Qiskit	Understand hybrid encryption and quantum-resilient handshake process.	CO3, CO4

SEMESTER-V

COURSE 15B: Quantum Machine Learning

Theory **Credits: 3** **3 hrs/week**

Course Objectives:

1. To understand the quantum representation of classical data and feature spaces.
2. To study quantum versions of supervised and unsupervised learning algorithms.
3. To apply hybrid quantum-classical approaches in machine learning.
4. To explore quantum reinforcement learning and optimization techniques.
5. To evaluate and apply quantum ML models in real-world case studies.

Course Outcomes:

1. Explain quantum data encoding and feature mapping methods.
2. Implement quantum supervised and unsupervised algorithms such as QSVM and quantum clustering.
3. Design and simulate hybrid quantum-classical models.
4. Apply quantum reinforcement learning and optimization algorithms.
5. Develop and present quantum ML solutions for domain-specific applications.

Unit I:

Quantum Data Representation & Feature Spaces

Quantum data encoding – amplitude, basis, and angle encoding methods, Quantum feature maps and kernels, Quantum distance measures and inner product evaluation, Quantum Principal Component Analysis (qPCA) for dimensionality reduction.

Unit II:

Quantum Supervised Learning Algorithms

Quantum Support Vector Machine (QSVM), Quantum Perceptron and Quantum Linear Regression, Quantum Decision Trees and Ensemble methods, Performance metrics and complexity analysis.

Unit III:

Quantum Unsupervised and Generative Models

Quantum k-Means and clustering approaches, Quantum Self-Organizing Maps, Quantum Autoencoders for feature extraction, Quantum Generative Adversarial Networks (QGANs) for data generation.

Unit IV:

Hybrid Quantum–Classical and Reinforcement Learning

Variational Quantum Circuits (VQC) and hybrid architectures, Quantum Approximate Optimization Algorithm (QAOA), Quantum Reinforcement Learning (QRL) and Quantum Policy Gradient, TensorFlow Quantum and PennyLane-based implementations.

Unit V:

Applications, Case Studies & Future Directions

Applications in finance, drug discovery, healthcare, and image recognition, Performance benchmarking and error mitigation in NISQ devices, Emerging quantum ML research (2025 trends), Ethical considerations and societal impact.

Textbooks and References:

1. **Karthikeyan, P., Akila, M., Sumathi, S., Poongodi, M. (2025).** *Quantum Machine Learning: A Modern Approach.* Routledge, CRC Press.
2. **Packt Publishing (2025).** *A Practical Guide to Quantum Machine Learning and Quantum Optimization.*
3. **Cambridge University Press (2025).** *Machine Learning in Quantum Sciences.*
4. **Nielsen, M.A. & Chuang, I.L. (2025).** *Quantum Computation and Quantum Information – Updated Edition.*
5. **IBM Qiskit Documentation (2025)** – https://qiskit.org
6. **TensorFlow Quantum Documentation (2025)** – https://www.tensorflow.org/quantum

SEMESTER-V

COURSE 15B: Quantum Machine Learning

Practical **Credits: 1** **2 hrs/week**

Exp. No.	Experiment Title	Tools / SDKs
1	Quantum Data Encoding and Visualization	IBM Qiskit
2	Implement Quantum Feature Mapping	Qiskit / PennyLane
3	Quantum Principal Component Analysis (qPCA)	Qiskit
4	Quantum Support Vector Machine (QSVM) for Classification	Qiskit ML
5	Quantum k-Means Clustering	PennyLane
6	Variational Quantum Circuit Optimization	TensorFlow Quantum
7	Hybrid Quantum Neural Network	TensorFlow Quantum
8	Quantum Reinforcement Learning Simulation	Cirq
9	Quantum Generative Adversarial Network (QGAN)	PennyLane
10	Mini Project – Quantum ML for Real Dataset	Any SDK (IBM Cloud or Local)

Yogi Vemana University::Kadapa

B.Sc. Honours(Quantum Technologies)

w.e.f. 2025-26 admitted batch

Recommended Format of Question Paper for all Courses

Time: 3 Hours *Max. Marks : 70*

Section-A

Answer any FIVE of the following questions. **5X4=20**

1. From unit-I
2. From Unit-I
3. From Unit-II
4. From Unit-II
5. From Unit-III
6. From Unit-III
7. From Unit-IV
8. From Unit-IV
9. From Unit-V
10. From Unit-V

Section-B

Answer all the questions from below. **5X10=50**

11. From Unit-I

    or

12. From Unit-I
13. From Unit-II

    (or)

14. From Unit-II
15. From Unit-III

    (or)

16. From Unit-III
17. From Unit-IV

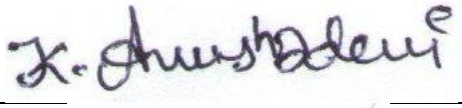
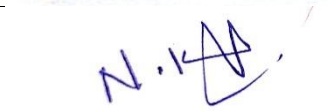
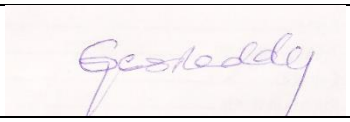
    (or)

18. From Unit-IV
19. From Unit-V

    (or)

20. From Unit-V

Board of Studies for Computer Science and Allied Courses, Computer Applications

Chair Person	Sri.G.Dayanandam	
Member	Smt.K.Anusha Devi	
Member	Smt.N.Kiranmai	
Member	Sk.Abjal Jeelani Basha	
Member	Sri.Chandra Sekhar Reddy	
University Nominee	Dr.M.Reddaiah