Major/DS Course (Core) COMP 6011 (Theory): Numerical Methods

Credit: 03 **45 Hours**

Course Objective

i. To understand the Importance of error analysis and their propagation.

ii. To introduce the basic concepts of solving algebraic, transcendental equations and system of linear and non-linear equations.

iii. To understand techniques of interpolation and polynomial fitting.

iv. To understand methods too numerical differentiation and integration.

v. To understand numerical solution of ordinary differential equations.

vi. To understand basic concepts of probability theory and distributions.

Course Outcomes:

After completion of the course students shall be able to:

i. Calculate errors induced in the values by truncation of a series expansion.

ii. Find roots of linear and non-linear system (algebraic and transcendental) equations.

iii. Fit polynomials to a given set of data points.

iv. Solve differential and integral equations numerically.

Syllabus:

Floating point representation and computer arithmetic, Significant digits, Errors: Round-off error, Local truncation error, Global truncation error, Order of a method, Convergence and terminal conditions, Efficient computations [10 Hours]

Bisection method, Secant method, Regula−Falsi method [3 Hours]

Newton−Raphson method, Newton„s method for solving nonlinear systems [3 Hours]

Gauss elimination method (with row pivoting) and Gauss−Jordan method, Gauss Thomas method for tridiagonal systems [3 Hours]

Iterative methods: Jacobi and Gauss-Seidel iterative methods [3 Hours]

Interpolation: Lagrange’s form and Newton’s form Finite difference operators, Gregory Newton

forward and backward differences Interpolation ,Piecewise polynomial interpolation: Linear interpolation, Cubic spline interpolation (only method), Numerical differentiation: First derivatives and second order derivatives, Richardson extrapolation . [5 Hours]

Numerical integration: Trapezoid rule, Simpson's rule (only method), Newton−Cotes open

formulas Extrapolation methods: Romberg integration, Gaussian quadrature, Ordinary differential equation: Euler's method [8 Hours]

Modified Euler's methods: Heun method and Mid-point method, Runge-Kutta second methods: Heun method without iteration, Mid-point method and Ralston's method Classical 4th order Runge-Kutta method, Finite difference method for linear ODE . [10 Hours]

Reference Books:

[1] Laurence V. Fausett, Applied Numerical Analysis, Using MATLAB, Pearson, 2/e (2012)

[2] M.K. Jain, S.R.K. Iyengar and R.K. Jain, Numerical Methods for Scientific and Engineering

Computation, New Age International Publisher, 6/e (2012)

[3] Steven C Chapra, Applied Numerical Methods with MATLAB for Engineers and Scientists, Tata McGraw Hill, 2/e (2010)

[4] William N. Venables and David M. Smith, An Introduction to R. 2nd Edition. Network Theory Limited.2009

[5] Norman Matloff, The Art of R Programming - A Tour of Statistical Software Design, No Starch Press.2011

COMP 6011 (Practical): Numerical Methods

Credit:1 **30 Hours**

All programs should be developed using C / Java / Python

1. Find the roots of the equation by bisection method.

2. Find the roots of the equation by secant/Regula−Falsi method.

3. Find the roots of the equation by Newton's method.

4. Find the solution of a system of nonlinear equation using Newto's method.

5. Find the solution of tridiagonal system using Gauss Thomas method.

6. Find the solution of system of equations using Jacobi/Gauss-Seidel method.

7. Find the cubic spline interpolating function.

8. Evaluate the approximate value of finite integrals using Gaussian/Romberg integration.

9. Solve the boundary value problem using finite difference method.

Major/DS Course (Core): COMP 6012 (Theory)- Microprocessors

Credit: 03 **45 Hours**

Course Objective

This course introduces internal architecture, programming models of Intel Microprocessors (8085 - Pentium) and assembly language programming. Students will also learn interfacing of memory and I/O devices with microprocessors.

Course Learning Outcomes

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

i. Describe the internal architecture of Intel microprocessors.

ii. Define and implement interfaces between the microprocessor and the devices.

iii. Write assembly language programs

Syllabus:

(All concepts should be studied in the context of the Intel 8085 Microprocessor.)

Microprocessor architecture: Internal architecture, system bus architecture, memory and I/O interfaces (15 Hour)

Microprocessor programming: Register Organization, instruction formats, assembly language Programming (15 Hour)

Interfacing: Memory address decoding, cache memory and cache controllers, I/O interface, keyboard, display, timer, interrupt controller, DMA controller, video controllers, communication interfaces (15 Hour)

Recommended Books:

1. Ramesh Gaonkar: Microprocessor Architecture, Programming and Application with the 8085

2. Barry B. Brey : The Intel Microprocessors : Architecture, Programming and Interfacing. Pearson Education, Sixth Edition,2009.

3. Walter A Triebel, Avtar Singh; The 8088 and 8086 Microprocessors Programming, Interfacing, Software, Hardware, and Applications. PHI, Fourth Edition 2005.

COMP 6012 (Practical): Microprocessor

Credit :1 **30 Hours**

All programs should be developed for the Intel 8085 Microprocessor.

ASSEMBLY LANGUAGE PROGRAMMING

1. Write a program for 32-bit binary division and multiplication
2. Write a program for 32-bit BCD addition and subtraction
3. Write a program for Linear search and binary search.
4. Write a program to add and subtract two arrays
5. Write a program for binary to BCD conversion
6. Write a program for BCD to binary conversion

Major/DS Course (Core) : COMP 6013 (Theory)- Theory of Computation

Credit: 03 **45 Hours**

Course Objective

This course introduces formal models of computation, namely, finite automaton, pushdown automaton, and Turing machine; and their relationships with formal languages. Students will learn about the limitations of computing machines as this course addresses the issue of which problems can be solved by computational means (decidability vs undecidability). The course is aimed to -

i. Formalizes the notion of problems via formal languages

ii. Make students aware of the notion of computation using "abstract computing devices" called automata

iii. familiarize students with the hierarchy of classes of problems or formal languages (regular, context-free, context-sensitive, decidable, and undecidable) and the hierarchy of classes of automata (finite automata, pushdown automata, and Turing machines)

Course Learning Outcomes

On successful completion of the course, a student will be able to:

i. Design a finite automaton, pushdown automaton or a Turing machine for a problem at hand.

ii. Apply pumping lemma to prove that a language is non-regular/non-context-free.

iii. Describe limitations of a computing machines and recognize what can be solved and what cannot be solved using these machines.

Syllabus:

1. Languages (5Hour)

Alphabets, string, language, Basic Operations on language, Concatenation, KleeneStar

2. Finite Automata and Regular Languages (15 Hour)

Regular Expressions, Transition Graphs, Deterministics and non-deterministic finite automata,

NFA to DFA Conversion, Regular languages and their relationship with finite automata, Pumping lemma and closure properties of regular languages, Moore & Mealy Machines.

3. Context free languages (15 Hour)

Context free grammars, parse trees, ambiguities in grammars and languages, Pushdown automata (Deterministic and Non-deterministic), Pumping Lemma, Properties of context free languages, normal forms.

4. Turing Machines and Models of Computations (10 Hour)

RAM, Turing Machine as a model of computation, Universal Turing Machine, Language

acceptability, decidability, halting problem, Recursively enumerable and recursive languages,

unsolvability problems.

Recommended Books:

1. Hoperoft, Aho, Ullman, Introduction to Automata theory, Language & Computation –3rd

Edition, Pearson Education. 2006

2. P. Linz, An Introduction to Formal Language and Automata 4th edition Publication Jones

Bartlett, 2006

3. Lewis & Papadimitriou, Elements of the theory of computation, PHI 1997.

4. Daniel I.A.Cohen, Introduction to computer theory, John Wiley, 1996

COMP 6013 (Tutorial): Theory of Computation

Tutorial: 15 Lectures

Tutorial based on theory classes.

Credit: 1

Major/DS Course (Core): COMP 6014 (Theory)- Artificial Intelligence

Credit: 03 **45 Hours**

Course Objective

The course objectives of this course are to:

Understand the foundations, basic concepts and techniques of Artificial Intelligence (AI).

Apply informed search techniques for different applications.

Impart knowledge about the use of core AI techniques, having applicability to a wide range of real-world problems.

Learn about various knowledge representation techniques and writing Prolog/LISP programs.

Learn about the latest techniques for developing AI systems.

Course Learning Outcomes

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

i. Identify problems that are amenable to solutions by specific AI methods.

ii. Appreciate the utility of different types of AI agents.

iii. Apply different informed search techniques for solving real world problems.

iv. Use knowledge representation techniques for AI systems.

v. Understand human level, data driven and end to end approaches to AI

Syllabus:

1. Introduction (5 Hours)

Introduction to Artificial Intelligence, Background and Applications, Turing Test and Rational

Agent approaches to AI, Introduction to Intelligent Agents, their structure, behavior and environment.

2. Problem Solving and Searching Techniques (10 Hour)

Problem Characteristics, Production Systems, Control Strategies, Breadth First Search, Depth First Search, Hill climbing and its Variations, Heuristics Search Techniques: Best First Search, A* algorithm, Constraint Satisfaction Problem, Means-End Analysis, Introduction to Game Playing, Min-Max and Alpha-Beta pruning algorithms.

3. Knowledge Representation (10 Hour)

Introduction to First Order Predicate Logic, Resolution Principle, Unification, Semantic Nets,

Conceptual Dependencies, Frames, and Scripts, Production Rules, Conceptual Graphs.

Programming in Logic (PROLOG) / LISP

4. Dealing with Uncertainty and Inconsistencies (10 Hour)

Truth Maintenance System, Default Reasoning, Probabilistic Reasoning, Bayesian Probabilistic Inference, Possible World Representations.

5. Understanding Natural Languages (10 Hour)

Parsing Techniques, Context-Free and Transformational Grammars, Recursive and Augmented

Transition Nets.

Books Recommended:

1. DAN.W. Patterson, Introduction to A.I and Expert Systems – PHI, 2007.
2. Russell &Norvig, Artificial Intelligence-A Modern Approach, LPE, Pearson Prentice Hall, 2nd edition, 2005.

3.Rich & Knight, Artificial Intelligence – Tata McGraw Hill, 2nd edition, 1991.

W.F. Clocksin and Mellish, Programming in PROLOG, Narosa Publishing

House, 3rd edition, 2001.

4.Ivan Bratko, Prolog Programming for Artificial Intelligence, Addison-Wesley, Pearson

Education, 3rd edition, 2000.

COMP 6014 (Practical): Artificial Intelligence

Credit: 1 **30 Hours**

All programs should be developed using Prolog or LISP

1. Write a prolog/lisp program to calculate the sum of two numbers.

2. Write a prolog/lisp program to find the maximum of two numbers.

3. Write a prolog/lisp program to calculate the factorial of a given number.

4. Write a prolog/lisp program to calculate the nth Fibonacci number.

5. Write a prolog/lisp program, insert n^{th}(item, n, into_list, result) that asserts that result is the list into_list with item inserted as the n„th element into every list at all levels.

6. Write a Prolog/lisp program to remove the n^{th} item from a list.

7. Write a Prolog/lisp program, remove-n^{th}(Before, After) that asserts the After list is the Before list with the removal of every n^{th} item from every list at all levels.

8. Write a Prolog/lisp program to implement append for two lists.

9. Write a Prolog/lisp program to implement palindrome (List).

10. Write a Prolog/lisp program to implement max(X,Y,Max) so that Max is the greater of two numbers X and Y.

11. Write a Prolog/lisp program to implement maxlist(List,Max) so that Max is the greatest number in the list of numbers List.

12. Write a Prolog/lisp program to implement sumlist(List,Sum) so that Sum is the sum of a given list of numbers List.

13. Write a Prolog/lisp program to implement two predicates even length(List) and Odd length(List) so that they are true if their argument is a list of even or odd length respectively.

14. Write a Prolog/lisp program to implement reverse(List,ReversedList) that reverses lists.

15. Write a Prolog/lisp program to implement maxlist(List,Max) so that Max is the greatest number in the list of numbers List using cut predicate.

16. Write a Prolog/lisp program to implement GCD of two numbers.

17. Write a prolog/lisp program that implements Semantic Networks/Frame Structures.