

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lecture-wise Plan

Subject Name: Cloud Computing

Year: 4th Year

Code-CS701

Semester: Seventh

Module Number	Topics	Number of Lectures
1	Cloud Computing Fundamental, Business Agility	6L
	Cloud computing definition, private, public and hybrid cloud. Cloud types, Iaas, Saas. Benefits and challenges of cloud computing. Role of virtualization in enabling the cloud.	
2	Cloud Applications	5L
	Technologies and the processes required when deploying web services; Deploying a web service from inside and outside a cloud architecture, advantages and disadvantages	
3	Cloud Services Management, Cloud Economics	9L
	Reliability, availability and security of services deployed from the cloud. Cloud Computing infrastructures available for implementing cloud based services. Economics of choosing a Cloud platform for an organization, based on application requirements, economic constraints and business needs.	
4	Application Development	8L
	Service creation environments to develop cloud based applications. Development environments for service development; Amazon, Azure, Google App.	
5	Best Practice Cloud IT Model	7L
	Analysis of Case Studies when deciding to adopt cloud computing architecture. How to decide if the cloud is right for your requirements. Cloud based service, applications and development platform deployment so as to improve the total cost of ownership (TCO)	
Total Number Of Hours = 35		

Faculty In-Charge

HOD, CSE Dept.

UNIVERSITY OF ENGINEERING AND MANAGEMENT, JAIPUR

Lecture-wise Plan

Subject Name: Cloud Computing

Code-CS701

Year: 4th Year

Semester: First

Assignment:

Module-1 (Cloud Computing Fundamental, Business Agility):

1. Explain private, public and hybrid cloud. What are the different types of Cloud are present?
2. Mention the Benefits and challenges of cloud computing.

Module-2 (Cloud Applications):

1. What are the technologies and the processes required when deploying web services in cloud?
2. Explain the deployment of web service from inside and outside a cloud architecture.

Module-3 (Cloud Services Management, Cloud Economics):

1. Explain reliability, availability and security of services deployed from the cloud.
2. What are the economic constraints and business needs to implement cloud environment?

Module-4 (Application Development):

1. How service creation environments are used to develop cloud based applications?
2. Explain how development environments for service development are used.

Module-5 (Best Practice Cloud IT Model):

1. How do you decide if the cloud is right for your requirements?
2. Explain Cloud based service.

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lecture-wise Plan

Subject Name: Compiler Design
Year: 4th Year

Subject Code-CS702
Semester: Seventh

Module Number	Course Details	Number of Lectures
UNIT 1	Introduction to Compiler:	4LH
1	<ul style="list-style-type: none">) Compiler Construction tools) Analysis of the source program) The Phases of a Compiler) Cousins of the Compiler) Grouping of phases – Front and back ends, passes) Introduction, Types of translators 	
2	Lexical Analysis:	6LH
	<ul style="list-style-type: none">) Role of Lexical Analyzer) Token, Patterns and Lexemes) Input buffering – buffer pairs and sentinels 	
UNIT 2	Syntax Analysis:	7LH
3	<ul style="list-style-type: none">) The role of a parser) Context free grammars, Writing a grammar) Top down Parsing) Non-recursive Predictive parsing (LL), Bottom up parsing, Handles) Viable prefixes) Operator precedence parsing) LR parsers (SLR, LALR), Parser generators (YACC)) Error Recovery strategies for different parsing techniques. 	
4	Syntax directed translation:	7LH
	<ul style="list-style-type: none">) Syntax directed definitions) Construction of syntax trees) Bottom-up evaluation of S attributed definitions) L attributed definitions) Bottom-up evaluation of inherited attributes. 	
UNIT 3	Type checking:	7LH
5	<ul style="list-style-type: none">) Type systems) Specification of a simple type checker) Equivalence of type expressions) Type conversions 	
	Run time environments:	

6	<ul style="list-style-type: none">) Source language issues (Activation trees, Control stack, scope of declaration, Binding of names)) Storage organization (Subdivision of run-time memory, Activation records)) Storage allocation strategies) Parameter passing (call by value, call by reference, copy restore, call by name)) Symbol tables) Dynamic storage allocation techniques. 	5LH
UNIT 4	Intermediate code generation:	
7	<ul style="list-style-type: none">) Intermediate languages) Graphical representation) Three-address code) Implementation of three address statements (Quadruples, Triples, Indirect triples). 	8LH
8	Code optimization and Code generations: <ul style="list-style-type: none">) Introduction) Basic blocks & flow graphs) Transformation of basic blocks) Dag representation of basic blocks,) The principle sources of optimization) Loops in flow graph) Peephole optimization) Code generations) Issues in the design of code generator, a simple code generator) Register allocation & assignment. 	
	Total Number Of Hours = 44	

Faculty In-Charge

HOD, CSE Dept.

Assignment:

Module-1(Introduction):

1. Find all strings in the language $(a+b)^*b(a+ab)^*$ of length less than 4.
2. With the help of a block diagram, show each phase of compiler including symbol table and error handling of a compiler.
3. Give the NFA for the following Regular Expression. Then find a DFA for the same language.
 $(a|b)^*abb$

Module-3 (Syntax Analysis):

1. Construct the Predictive Parsing table for the following grammar:

$E \rightarrow E+ T | T$

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lecture-wise Plan

Subject Name: Compiler Design
Year: 4th Year

Subject Code-CS702
Semester: Seventh

$T \rightarrow T * F | F$

$F \rightarrow (E) | id$

2. Parse the following string by operator precedence parsing:

$Id1 + id2 * id3$

3. What are the main contributions of syntax directed translation in compiler? Design a dependency graph and direct acyclic graph for the string

$a + a * (b - c) + (b - c) * d$

4. What is operator precedence parsing? Discuss about the advantage and disadvantage of operator precedence parsing. consider the following grammar:

$E \rightarrow TA$

$A \rightarrow +TA | \epsilon$

$T \rightarrow FB$

$B \rightarrow *FB | \epsilon$

$F \rightarrow id$

Test whether this grammar is operator precedence grammar or not and show how the string $w = id + id * id + id$ will be processed by this grammar.

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lecture-wise Plan

Subject Name: Image Processing
Year: 4th Year

Subject Code-CS703A
Semester: Seventh

Module Number	Topics	Number of Lectures
1.	Introduction:	4L
	1. Background, Digital Image Representation,	1
	2. Fundamental steps in Image Processing	1
	3. Elements of Digital Image Processing – Image Acquisition, Storage, Processing, Communication, Display.	2
2.	Digital Image Formation	6L
	1. A Simple Image Model, Geometric Model- Basic Transformation (Translation, Scaling, Rotation),	3
	2. Perspective Projection,	1
	3. Sampling & Quantization - Uniform & Non-uniform	2
3.	Mathematical Preliminaries	8L
	1. Neighbour of pixels, Connectivity, Relations, Equivalence & Transitive Closure;	2
	2. Distance Measures, Arithmetic/Logic Operations, Fourier Transformation,	3
	3. Properties of The Two Dimensional Fourier Transform, Discrete Fourier Transform, Discrete Cosine & Sine Transform	3
4.	Image Enhancement	10L
	1. Spatial Domain Method, Frequency Domain Method, Contrast Enhancement – Linear & Nonlinear Stretching	3
	2. Histogram Processing; Smoothing - Image Averaging, Mean Filter, Low-pass Filtering; Image Sharpening. High-pass Filtering, High-boost Filtering	4
	3. Derivative Filtering, Homomorphic Filtering; Enhancement in the frequency domain - Low pass filtering, High pass filtering.	3
5.	Image Restoration	6L
	1. Degradation Model, Discrete Formulation, Algebraic Approach to Restoration – Unconstrained & Constrained	2
	2. Constrained Least Square Restoration, Restoration by Homomorphic Filtering,	2
	3. Geometric Transformation - Spatial Transformation, Gray Level Interpolation	2
6.	Image Segmentation	8L
	1. Point Detection, Line Detection, Edge detection, Combined detection, Edge Linking & Boundary Detection - Local Processing	3
	2. Global Processing via The Hough Transform; Thresholding - Foundation, Simple Global Thresholding, Optimal Thresholding	3
	3. Region Oriented Segmentation - Basic Formulation, Region Growing by Pixel Aggregation, Region Splitting & Merging	2
Total Number Of Hours = 42		

Faculty In-Charge

HOD, CSE Dept.

Assignment :

Module -1 (Introduction)

Module -2 (Digital Image Formation)

Module -3 (Mathematical Preliminaries)

Module -4 (Image Enhancement)

Module -5 (Image Restoration)

Module -6 (Image Segmentation)

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lecture-wise Plan

Subject Name: Pattern Recognition
Year: 4th Year

Subject Code-CS703B
Semester: Seventh

Module Number	Topics	Number of Lectures
1	Introduction:	7L
	Introduction – Definitions, data sets for Pattern Recognition. Different Paradigms of Pattern Recognition. Representations of Patterns and Classes. Metric and non-metric proximity measures.	7
2	Feature extraction:	7L
	Feature extraction Different approaches to Feature Selection Nearest Neighbour Classifier and variants Efficient algorithms for nearest neighbour classification	7
3	Prototype Selection:	11L
	Different Approaches to Prototype Selection Bayes Classifier Decision Trees Linear Discriminant Function	11
4	Clustering:	11L
	Support Vector Machines Clustering. Clustering Large datasets. Combination of Classifiers. Applications – Document Recognition	11
Total Number Of Hours = 36		

Assignment:

Module-1:

1. Describe the basic modules in designing a pattern recognition system.
2. Briefly explain what is generalization in the context of pattern recognition problems?

Module-2:

1. Explain the concept of feature extraction in pattern recognition system with examples.
2. Explain the Nearest Neighbor Rule used in Density Estimation.

Module-3:

1. Write a short note with diagrams on Decision trees which are nonlinear, nonmetric classifiers.
2. State the Bayes Rule and explain how it is applied to pattern classification problems.

Module-4:

1. To which category of clustering schemes does the k-means algorithm belong? What is its major advantage?
2. Which are the two schemes of Hierarchical clustering algorithm? Give brief descriptions.

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lecture-wise Plan

Subject Name: Soft Computing
Year: 4th Year

Subject Code-CS703C
Semester: Seventh

Module Number	Topics	Number of Lectures
1	Introduction:	2L
	1. Introduction to soft computing; introduction to fuzzy sets and fuzzy logic systems.	1
	2. Introduction to biological and artificial neural network; introduction to Genetic Algorithm.	1
2	Fuzzy sets and Fuzzy logic systems:	10L
	1. Classical Sets and Fuzzy Sets and Fuzzy relations: Operations on Classical sets, properties of classical sets, Fuzzy set operations, properties of fuzzy sets, cardinality, operations, and properties of fuzzy relations.	2
	2. Membership functions : Features of membership functions, standard forms and boundaries, different fuzzification methods.	1
	3. Fuzzy to Crisp conversions: Lambda Cuts for fuzzy sets, fuzzy Relations, Defuzzification methods.	1
	4. Classical Logic and Fuzzy Logic: Classical predicate logic, Fuzzy Logic, Approximate reasoning and Fuzzy Implication	1
	5. Fuzzy Rule based Systems: Linguistic Hedges, Fuzzy Rule based system- Aggregation of fuzzy Rules, Fuzzy Inference SystemMamdani Fuzzy Models – Sugeno Fuzzy Models.	3
	6. Applications of Fuzzy Logic: How Fuzzy Logic is applied in Home Appliances, General Fuzzy Logic controllers, Basic Medical Diagnostic systems and Weather forecasting	2
3.	Neural Network	10L
	1. Introduction to Neural Networks: Advent of Modern Neuroscience, Classical AI and Neural Networks, Biological Neurons and Artificial neural network, model of artificial neuron.	2
	2. Learning Methods : Hebbian, competitive, Boltzman etc.,	2
	3. Neural Network models: Perceptron, Adaline and Madaline networks; single	2

	layer network; Back-propagation and multi layer networks.	
	4. Competitive learning networks: Kohonen self organizing networks, Hebbian learning; Hopfield Networks.	2
	5. Neuro-Fuzzy modelling: Applications of Neural Networks: Pattern Recognition and classification	2
4	Genetic Algorithms	10L
	1. Genetic Algorithms: Simple GA, crossover and mutation, Multi-objective Genetic Algorithm (MOGA).	4
	2. Applications of Genetic Algorithm: genetic algorithms in search and optimization, GA based clustering Algorithm, Image processing and pattern Recognition	6
5	Soft Computing techniques	4L
	1. Other Soft Computing techniques: Simulated Annealing, Tabu search, Ant colony optimization (ACO), Particle Swarm Optimization (PSO).	4

Faculty In-Charge
CSE Dept.

HOD,

Assignment:

Module-1(Introduction):

1. Soft Computing vs Hard Computing.

Module-2 (Fuzzy sets and Fuzzy logic systems):

1. Classical Sets vs Fuzzy Sets
2. Explain different membership functions.
3. Fuzzy to Crisp conversions
4. Fuzzy Rule based Systems
5. Applications of Fuzzy Logic

Module-3(Neural Network):

1. Biological Neurons vs artificial neural network
2. Different Learning Methods
3. Different Neural Network models
4. Competitive learning networks
5. Applications of Neural Networks

Module-4(Genetic Algorithms):

1. Genetic Algorithms and application of Genetic Algorithms

Module-5(Soft Computing techniques):

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lecture-wise Plan

1. Soft Computing techniques

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lecture-wise Plan

Subject Name: Artificial Intelligence
Year: 4th Year

Subject Code-CS703D
Semester: Seventh

Module Number	Topics	Number of Lectures
1	Introduction:	2L
	Overview of Artificial intelligence- Problems of AI, AI technique, Tic - Tac - Toe problem.	
2	Intelligent Agents	2L
	Agents & environment, nature of environment, structure of agents, goal based agents, utility based agents, learning agents.	
3	Problem Solving	2L
	Problems, Problem Space & search: Defining the problem as state space search, production system, problem characteristics, issues in the design of search programs.	
4	Search techniques	5L
	Solving problems by searching: problem solving agents, searching for solutions; uniform search strategies: breadth first search, depth first search, depth limited search, bidirectional search, comparing uniform search strategies.	
5	Heuristic search strategies	5L
	Greedy best-first search, A* search, memory bounded heuristic search: local search algorithms & optimization problems: Hill climbing search, simulated annealing search, local beam search, genetic algorithms; constraint satisfaction problems, local search for constraint satisfaction problems	
6	Adversarial search	3L
	Games, optimal decisions & strategies in games, the mini max search procedure, alpha-beta pruning, additional refinements, iterative deepening.	
7	Knowledge & reasoning	3L
	Knowledge representation issues, representation & mapping, approaches to knowledge representation, issues in knowledge representation.	

8	Using predicate logic	2L
	Representing simple fact in logic, representing instant & ISA relationship, computable functions & predicates, resolution, natural deduction.	
9	Representing knowledge using rules	3L
	Procedural verses declarative knowledge, logic programming, forward verses backward reasoning, matching, control knowledge.	
10	Probabilistic reasoning	4L
	Representing knowledge in an uncertain domain, the semantics of Bayesian networks, Dempster-Shafer theory, Fuzzy sets & fuzzy logics.	
11	Planning	2L
	Overview, components of a planning system, Goal stack planning, Hierarchical planning, other planning techniques.	
12	Natural Language processing	2L
	Introduction, Syntactic processing, semantic analysis, discourse & pragmatic processing.	
13	Learning	2L
	Forms of learning, inductive learning, learning decision trees, explanation based learning, learning using relevance information, neural net learning & genetic learning.	
14	Expert Systems	2L
	Representing and using domain knowledge, expert system shells, knowledge acquisition	
Total Number Of Hours = 39		

Faculty In-Charge

HOD, CSE Dept.

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lecture-wise Plan

Subject Name: Artificial Intelligence

Year: 4th Year

Assignments:

Subject Code-CS703D

Semester: Seventh

Module-I: Introduction

1. What do you mean by Artificial intelligence?
2. Explain Tic - Tac - Toe problem.

Module-II: Intelligent Agents

1. Explain nature of environment
2. Discuss the followings:
 -) structure of agents
 -) goal based agents
 -) utility based agents
 -) Learning agents

Module-III: Problem Solving

1. Explain how the problem as state space search has defined?
2. Define problem characteristics and issues in the design of search programs.

Module-IV: Search techniques

1. What do you mean by problem solving agents? searching for solutions
2. Explain depth limited search, bidirectional search.

Module-V: Heuristic search strategies

1. Explain Greedy best-first search
2. How Hill climbing search and simulated annealing search are different from each other?

Module-VI: Adversarial search

1. What do you mean by optimal decisions & strategies in games?
2. Explain the mini max search procedure, alpha-beta pruning.

Module-VII: Knowledge & reasoning

1. Explain different knowledge representation issues, representation & mapping.
2. Mention different approaches to knowledge representation. What are the issues in knowledge representation?

Module-VIII: Using predicate logic

1. How you represent simple facts in logic?

2. Explain ISA relationship, computable functions & predicates.

Module-IX: Representing knowledge using rules

1. Differentiate Procedural and declarative knowledge
2. Explain logic programming. What are the differences between forward and backward reasoning?

Module-X: Probabilistic reasoning

1. How you represent knowledge in an uncertain domain?
2. Explain the semantics of Bayesian networks. What do you mean by Dempster-Shafer theory?

Module-XI: Planning

1. Explain the components of a planning system. What is Goal stack planning?
2. What do you mean by Hierarchical planning?

Module-XII: Natural Language processing

1. Explain Syntactic processing in NLP.
2. What do you mean by semantic analysis?

Module-XIII: Learning

1. Explain the different forms of learning. What do you mean by inductive learning, learning decision trees, explanation based learning?
2. Differentiate neural net learning & genetic learning.

Module-XIV: Expert Systems

1. How do you representing and use domain knowledge?
2. Explain expert system shells, knowledge acquisition.

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lecture-wise Plan

Subject Name: Distributed Operating System

Subject Code-CS704A

Year: 4thYear

Semester: **Seventh**

Module Number	Topics	Number of Lectures
1	Introduction to Distributed System	2L
	1. Introduction, Examples of distributed system,	1
	2. Resource sharing, Challenges	1
2	Operating System Structures	3L
	1. Review of structures: monolithic kernel, layered systems, virtual machines.	1
	2. Process based models and client server architecture;	1
	3. The micro-kernel based client-server approach.	1
3	Communication	4L
	1. Inter-process communication,	1
	2. Remote Procedure Call, Remote Object Invocation,	1
	3. Tasks and Threads. Examples from LINUX, Solaris 2 and Windows NT.	2
4	Theoretical Foundations	4L
	1. Introduction. Inherent Limitations of distributed Systems.	2
	2. Lamport's Logical clock. Global State	2
5	Distributed Mutual Exclusion	4L
	1. Classification of distributed mutual exclusion algorithm.	1
	2. NonToken based Algorithm: Lamport's algorithm, Ricart-Agrawala algorithm.	1
	3. Token based Algorithm: Suzuki-Kasami's broadcast algorithm.	2
6	Distributed Deadlock Detection	4L
	1. Deadlock handling strategies in distributed systems. Control organizations for distributed deadlock detection.	1
	2. Centralized and Distributed deadlock detection algorithms: Completely Centralized algorithms, path pushing, and edge chasing, global state detection algorithm.	3
7	Protection and Security	4L
	1. Requirements for protection and security regimes. The access matrix model of protection.	2
	2. System and user modes, rings of protection, access lists, capabilities. User authentication, passwords and signatures. Use of single key and public key encryption.	2

UNIVERSITY OF ENGINEERING AND MANAGEMENT, JAIPUR

Lecture-wise Plan

Subject Name: Mobile Computing

Subject Code-CS704D

Year: 4thYear Semester: 7th

8	Distributed file systems	6L
	<ol style="list-style-type: none">1. Issues in the design of distributed file systems: naming, transparency, update semantics and fault resilience.2. Use of the Virtual File System layer. Examples of distributed systems including Sun NFS, the Andrew filestore, CODA file system and OSF DCE.	3 3
9	Distributed Shared Memory	4L
	<ol style="list-style-type: none">1. Architecture and motivations. Algorithms for implementing DSM.2. Memory Coherence	2 2
10	CORBA	3L
	<ol style="list-style-type: none">1. The Common Object Request Broker Architecture model2. Software and its relationship to Operating Systems.	1 2
Total Number Of Lectures = 38		

Faculty In-Charge

HOD, CSE Dept.

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lecture-wise Plan

Subject Name: Distributed Operating System

Year: 4th Year

Assignment:

Subject Code-CS704A

Semester: **Seventh**

Module-1(Introduction to Distributed System):

1. Write down the examples of distributed system,
2. What do you mean by Resource sharing? What are the Challenges?

Module-2 (Operating System Structures):

1. Explain monolithic kernel, layered systems, virtual machines.
2. What do you mean by Process based models and client server architecture?

Module-3(Communication):

1. What is Inter-process communication?
2. Explain Remote Procedure Call, Remote Object Invocation.

Module-4(Theoretical Foundations):

1. Explain the Inherent Limitations of distributed Systems.
2. What is Lamport's Logical clock?

Module-5(Distributed Mutual Exclusion):

1. Classify the distributed mutual exclusion algorithm.
2. Explain Ricart-Agrawala algorithm and Suzuki-Kasami's broadcast algorithm.

Module-6 (Distributed Deadlock Detection):

1. Explain the Deadlock handling strategies in distributed systems. Also explain the Control organizations for distributed deadlock detection.
2. Mention the differences between Centralized and Distributed deadlock detection algorithms.

Module-7 (Protection and Security):

1. What do you mean by rings of protection?
2. How User authentication is done? Is there any differences between passwords and signatures?

Module-8 (Distributed file systems):

1. Write down the issues to the design the distributed file systems.
2. Explain how you use the Virtual File System layer. Mention the examples of distributed systems including Sun NFS.

Module-9 (Distributed Shared Memory):

1. Explain the Algorithms for implementing DSM.
2. What do you mean by Memory Coherence?

Module-10 (CORBA):

1. What is CORBA?
2. How Software are related to Operating Systems?

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lecture-wise Plan

Subject Name: Data Warehousing & Data Mining
Year: 4th Year

Subject Code- CS704B
Semester: Seventh

Module Number	Topics	Number of Lectures
1	Overview of Data warehousing	4L
	Strategic information and the need for Data warehousing, ..	1
	Defining a Data warehouse, Evolution of Data warehousing	2
	Data warehousing and Business Intelligence	1
2	The Building Blocks of Data warehouse	5L
	Defining features – Subject-oriented data, Integrated data, Time-variant data.	2
	Nonvolatile data, Data granularity Data warehouses and Data marts Architectural Types – Centralized, Independent data marts, Federated, Hub-and-Spoke, Data mart bus Overview of components - Source Data, Data Staging,	2
	Data Storage, Information Delivery, Metadata, and Management and Control components.	1
3.	Business Requirements and Data warehouse	6L
	Dimensional nature of Business data and Dimensional Analysis, Dimension hierarchies and categories, Key Business Metrics (Facts),	3
	Requirement Gathering methods and Requirements Definition Document (contents) Business Requirements and Data Design – Structure for Business Dimensions and Key Measurements, Levels of detail Business Requirements and the Architecture plan Business Requirements and Data Storage Specifications Business Requirements and Information Delivery Strategy	3
4.	Architectural components	7L
	Concepts of Data warehouse architecture – Definition and architecture in the areas of Data acquisition, Data storage, and Information delivery Distinguishing characteristics – Different objectives and scope.	2
	Data content, Complex analysis for faster response, Flexible and Dynamic, Metadata-driven etc. Architectural Framework – supporting flow of data, and the Management and Control module Technical architecture – Data acquisition, Data storage, and Information delivery Overview of the components of Architectural.	3
	Metadata types by functional areas – Data acquisition, Data storage, and Information delivery Business Metadata – overview of content and examples Technical Metadata – overview of content and examples	2

	Metadata Requirements, Sources of Metadata, Metadata management – challenges, Metadata Repository, Metadata integration and standards	
5.	Matching information to classes of users	12L
	Information from Data warehouse versus Operational systems, Users of information – their needs and how to provide information, Information delivery – queries, reports, analysis, and applications, Information delivery tools – Desktop environment, Methodology and criteria for tool selection, Information delivery framework, Business Activity Monitoring, Dashboards and Scorecards,.	3
	OLAP in Data warehouse Overall concept of Online Analytical Processing (OLAP), OLAP definitions and rules, OLAP characteristics Major features and functions of OLAP – General features, Dimensional analysis, Hypercubes, Drill Down and Roll Up, Slice and Dice, Rotation, Uses and Benefits,	3
	Data Warehouse and the web Web-enabled Data Warehouse – adapting data warehouse for the web, Web-based information delivery – Browser technology for data warehouse and Security issues, OLAP and Web – Enterprise OLAP, Web-OLAP approaches, OLAP Engine design	3
	Data Mining Overview of Data mining – Definition, Knowledge Discovery Process (Relationships, Patterns, Phases of the process), OLAP versus Data mining, Some aspects of Data mining – Association rules, Outlier analysis, Predictive analytics etc), Concepts of Data mining in a Data warehouse environment, Major Data Mining techniques – Cluster Detection, Decision Trees, Memory-based Reasoning, Link Analysis, Neural Networks, Genetic Algorithms etc, Data Mining Applications in industry – Benefits of Data mining, Discussion on applications in Customer Relationship Management (CRM), Retail, Telecommunication, Biotechnology, Banking and Finance etc.	3
Total Number Of Hours = 34		

Faculty In-Charge

HOD, CSE Dept.

Assignment:

1. What is Data Mining? What are the advantages of data mining over traditional approach?
2. What do you understand by Web mining? Compare web mining with data mining.
3. Write down the differences between star schema and snowflake schema.
4. Discuss different types of Meta data with proper example.
5. What is clustering? Why do we need Cluster for a large set of data?
6. In data warehousing technology explain ROLAP, MOLAP and HOLAP techniques of implementing a multidimensional view.
7. Generate all Frequent Item sets from the following transaction data given minimum support=0.3.

TID	ITEMS
T ₁	{ CSE,EE,CE,IT }
T ₂	{ EE,PE,IT }
T ₃	{ EE,CE }
T ₄	{ CSE,EE,PE }
T ₅	{ CSE,CE }
T ₆	{ EE,CE }
T ₇	{ CSE,CE,IT }
T ₈	{ CSE,EE,CE,IT }
T ₉	{ CSE,EE,CE }
T ₁₀	{ CE,PE,IT }

Find five Association rules from the above Frequent sets at min. 50% confidence.

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lecture-wise Plan

Subject Name: Sensor Networks
Year: 4th Year

Subject Code-CS704C
Semester: Seventh

Module Number	Topics	Number of Lectures
1	Introduction and Overview:	4L
	Overview of wireless networks, types, infrastructure-based and infrastructure-less,	1
	introduction to MANETs (Mobile Ad-hoc Networks), characteristics, reactive and proactive routing protocols with examples.	1
	Introduction to sensor networks, commonalities and differences with MANETs,	1
	constraints and challenges, advantages, applications, enabling technologies for WSNs.	1
2	Architectures:	9L
	Single-node architecture – hardware components	1
	design constraints, energy consumption of sensor nodes	1
	operating systems and execution environments, examples of sensor nodes	1
	sensor network scenarios, types of sources and sinks	1
	single hop vs. multi hop networks, multiple sources and sinks	1
	mobility, optimization goals and figures of merit	1
	gateway concepts	1
	design principles for WSNs	1
	service interfaces for WSNs	1
	Communication Protocols:	9L
3.	Physical layer and transceiver design considerations	1
	MAC protocols for wireless sensor networks	1
	low duty cycle protocols and wakeup concepts - S-MAC	1
	the mediation device protocol, wakeup radio concepts, address and name management, assignment of MAC addresses	1
	routing protocols- classification, gossiping, flooding, energy-efficient routing	1
	unicast protocols, multi-path routing, data-centric routing	1
	data aggregation, SPIN,	1
	LEACH	1
	Directed-Diffusion, geographic routing.	1

4	Infrastructure Establishment:	9L
	Topology control, flat network topologies	1
	hierarchical networks by clustering, time synchronization, properties	1
	protocols based on sender-receiver and receiver-receiver synchronization	1
	LTS, TPSN	1
	RBS, HRTS	1
	localization and positioning, properties and approaches, single-hop localization	1
	positioning in multi-hop environment	1
	range based localization algorithms – location services	1
	sensor tasking and control	1
5	Sensor Network Platforms and Tools:	9L
	Sensor node hardware	1
	Berkeley motes	1
	programming challenges, node-level software platforms	1
	node-level simulators	1
	state-centric programming	1
	Tiny OS	1
	nesC components	1
	NS2 simulator	1
	TOSSIM	1
Total Number Of Hours = 40		

Faculty In-Charge

HOD, CSE Dept.

Assignment:

Module-1:

1. Define MANET? Write down the characteristics of MANET.
2. Define mote? Describe the architecture of a sensor node.

Module-2:

1. Define mobility of sensor device
2. Discuss energy consumption issues of a sensor node

Module-3:

1. Write Short notes on the following topics

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lecture-wise Plan

A. LEACH

B. SPIN

Module-4:

1. Discuss Protocol based communication between two sensor nodes
2. Differentiate between LTS and HRTS

Module-5:

1. What do you mean by state centric programming?
2. Write Short notes on the following topics
 - A. TinyOS
 - B. TOSSIM
 - C. NS2
 - D. Berkeley Motes

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lecture-wise Plan

Subject Name: Mobile Computing
Year: 4th Year

Subject Code-CS704D
Semester: **Seventh**

Module Number	Topics	Number of Lectures
1	PCS Architecture, GSM	6L
	1. Introduction to Personal Communications Services (PCS): PCS Architecture,	2L
	2. Mobility management, Networks signalling.	2L
	3. Global System for Mobile Communication (GSM) system overview: GSM Architecture, Mobility management, Network signalling.	2L
2	GPRS	5L
	1. General Packet Radio Services (GPRS): GPRS Architecture, GPRS Network Nodes.	2L
	2. Mobile Data Communication: WLANs (Wireless LANs) IEEE 802.11 standard, Mobile IP.	3L
3	Wireless Application Protocol	7L
	1. Wireless Application Protocol (WAP): The Mobile Internet standard, WAP Gateway and Protocols,	3L
	2. Wireless mark-up Languages (WML).	2L
	3. Wireless Local Loop (WLL): Introduction to WLL Architecture, Wireless Local Loop Technologies.	
4	3G Services	7L
	1. Third Generation (3G) Mobile Services: Introduction to International Mobile Telecommunications 2000 (IMT 2000) vision,	4L
	2. Wide band Code Division Multiple Access (WCDMA), and CDMA 2000, Quality of services in 3G.	3L
5	GMSS, Bluetooth	7L
	1. Global Mobile Satellite Systems; case studies of the IRIDIUM and GLOBALSTAR systems.	2L
	2. Wireless Enterprise Networks: Introduction to Virtual Networks.	2L
	3. Bluetooth technology, Bluetooth Protocols.	3L
6	Server side concept, Applications	8L
	1. Server-side programming in Java,	4L
	2. Pervasive web application architecture	2L
	3. Device independent example application	2L
Total Number Of Lectures = 40		

Faculty In-Charge

HOD, CSE Dept.

UNIVERSITY OF ENGINEERING AND MANAGEMENT, JAIPUR

Lecture-wise Plan

Subject Name: Mobile Computing

Subject Code-CS704D

Year: 4thYear Semester: 7th

Assignment:

Module-1(PCS Architecture, GSM):

1. List all the services provided by GSM. What are the services provided by supplementary services?What are main subsystems of GSM architecture?
2. What are frequencies used in forward and reverse link frequency in GSM? Explain PCS architecture.

Module-2 (GPRS):

1. What is meant by GPRS? What is the function of an AuC?
2. What are the three types of switching methods? What is QoS in GPRS? Explain WLAN.

Module-3(Wireless Application Protocol):

1. Explain WAP. What do you mean by WML?
2. Explain WLL architecture with proper diagram.

Module-4(3G Services):

1. Explain the concepts of CDMA. What are its merits and demerits? Explain the working principle of RAKE receiver.
2. Explain WCDMA. How quality of service is managed in 3G network?

Module-5(GMSS, Bluetooth):

1. State the modes possible when the slave is in connection state in Bluetooth.What are elements available under link security of Bluetooth technology?
2. List few functions of Bluetooth.Differentiate piconet and scatternet in Bluetooth technology.What re the two kinds of profiles in Bluetooth 1.1 version?

Module-6 (Server side concept, Applications):

1. How do you prepare Server-side programming in Java?
2. Explain Pervasive web application architecture. Give an example of Device independent application.

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lecture-wise Plan

Subject Name: Internet Technology

Year: 4th Year

Subject Code-CS705A

Semester: Seventh

Module Number	Topics	Number of Lectures
1	Introduction:	6L
	1. Overview, Network of Networks, Intranet, Extranet and Internet.	1L
	2. World Wide Web: Domain and Sub domain, Address Resolution, DNS, Telnet, FTP, HTTP.	1L
	3. Review of TCP/IP: Features, Segment, Three-Way Handshaking, Flow Control, Error Control, Congestion control, IP Datagram, IPv4 and IPv6.	1L
	4. IP Subnetting and addressing: Classful and Classless Addressing, Subnetting. NAT, IP masquerading, IP tables.	1L
	5. Internet Routing Protocol: Routing - Intra and Inter Domain Routing, Unicast and Multicast Routing, Broadcast.	1L
	6. Electronic Mail: POP3, SMTP.	1L
2	HTML, Image Maps, Extensible Markup Language, CGI Scripts:	9L
	1. Introduction of Editors, Elements, Attributes, Heading, Paragraph. Formatting, Link, Head, Table, List, Block, Layout, CSS. Form, Iframe, Colors, Colorname, Colorvalue.	3L
	2. Map, area, attributes of image area.	1L
	3. Introduction of Tree, Syntax, Elements, Attributes, Validation, Viewing. XHTML in brief.	4L
	4. Introduction, Environment Variable, GET and POST Methods.	1L
3	PERL, JavaScript, Cookies, Java Applets:	10L
	1. Introduction of Variable, Condition, Loop, Array, Implementing data structure, Hash, String, Regular Expression, File handling, I/O handling.	3L
	2. Basics, Statements, comments, variable, comparison, condition, switch, loop, break. Object – string, array, Boolean, reg-ex. Function, Errors, Validation.	4L
	3. Definition of cookies, Create and Store a cookie with example.	1L

	4. Container Class, Components, Applet Life Cycle, Update method; Parameter passing applet, Applications.	2L
4	Client-Server programming In Java, Threats, Network security techniques, Firewall:	4L
	1. A) Java Socket, Java RMI. 1. B) Malicious code-viruses, Trojan horses, worms; eavesdropping, spoofing, modification, denial of service attacks.	2L
	2. A) Password and Authentication; VPN, IP Security, security in electronic transaction, Secure Socket Layer (SSL), Secure Shell (SSH). B) Introduction of Packet filtering, Stateful, Application layer, Proxy	2L
5	Internet Telephony, Multimedia Applications, Search Engine and Web Crawler:	5L
	1. Introduction of VoIP.	1L
	2. Multimedia over IP: RSVP, RTP, RTCP and RTSP. Streaming media, Codec and Plugins, IPTV.	2L
	3. Definition, Meta data, Web Crawler, Indexing, Page rank, overview of SEO.	2L
Total Number Of Hours = 34		

Faculty In-Charge

HOD, CSE Dept.

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lecture-wise Plan

Subject Name: Microelectronics & VLSI Design

Subject Code-CS705B

Year: 4th Year

Semester: Seventh

Module Number	Topics	Number of Lectures
1	Introduction to VLSI Design	6L
	1. VLSI Design Concepts, Moor's Law, Scale of Integration (SSI, MSI, LSI, VLSI, ULSI – basic idea only), Types of VLSI Chips (Analog & Digital VLSI)	1L
	2. Design styles- ASIC, PLA, PAL etc.	1L
	3. FPGA, Gate Array based design etc.	1L
	4. Top down, Bottom up, Semi custom, Full custom etc.	1L
	5. Design principles (Digital VLSI-Concept of Regularity etc.)	1L
	6. Design Domains (Behavioral, Structural, physical-Y chart)	1L
2	MOS Structures	12L
	1. E-MOS & D-MOS	1L
	2. Charge inversion in E-MOS	1L
	3. Threshold voltage, Flat-Band voltage, Potential balance & Charge balance	1L
	4. Inversion, MOS capacitances	1L
	5. three-terminal MOS structure with Body-effect	1L
	6. four-terminal MOS transistor: Drain current	1L
	7. I-V characteristics, Current-voltage equations (simple derivation),	1L
	8. scaling in MOSFET: General scaling, Constant voltage scaling & Constant field scaling	1L
	9. Short channel effects	2L
	10. CMOS inverter, Simple Combinational Gates-NAND gate and NOR gate using CMOS	2L
3	Micro-electronic Processes for VLSI Fabrication	10L
	1. Silicon Semiconductor Technology- An Overview, Wafer processing	1L
	2. Oxidation, Epitaxial deposition, Ion-implantation, Diffusion	1L
	3. Cleaning, Etching	1L
	4. Photo-lithography– Positive & Negative photo-resist	1L
	5. Basic CMOS Technology – Steps in fabricating CMOS	1L

	6. Basic n-well CMOS process, p-well CMOS process, Twin tub process	1L
	7. Silicon on insulator (SoI)	1L
	8. Layout Design Rules	1L
	9. Stick diagram with examples	1L
	10. Continue: Stick diagram with examples	1L
4	Hardware Description Language	12L
	1. Introduction, HDL and software languages, simulation, synthesis, VHDL, capabilities	1L
	2. Basic terminologies, entity, architecture,	1L
	3. Dataflow, structural, behavioural, mixed	1L
	4. Configuration declaration, package declaration, package body	1L
	5. Basic language elements	2L
	6. Details of Behavioural	L
	7. Details of Dataflow	1L
	8. Details of structural	L
Total Number Of Hours = 37L		

Angshuman Khan
Faculty In-Charge

Sandip Das
HOD, ECE Dept.

Assignment:

Module-1(Introduction to VLSI Design):

1. State Moore's law.
2. Describe VLSI design cycle. Why is it called cycle?
3. Write short note: regularity, modularity, locality.
4. What is hierarchical decomposition?
5. Describe 'Y' chart.
6. Discuss: CPLD, ROM, PLA, PAL, top-down & bottom-up approach, semicustom & full custom design.

Module-2 (MOS structure)

1. Describe accumulation, depletion, inversion, and pinch-off conditions of NMOS.
2. Draw C-V characteristics of MOS capacitor.
3. Discuss threshold voltage.

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lecture-wise Plan

4. Derive the drain current equation of nMOS.
5. Design XOR gate using CMOS.
6. Prove that size of PMOS is 2.5 times of NMOS.

Module-3 (Micro-electronic Processes for VLSI Fabrication):

1. State Lambda rule and Micron rule?
2. Draw the lay-out of NAND2 and NOR2.
3. Draw the stick diagram of CMOS.
4. What is SOI and twin tub process?
5. Describe n-well fabrication process.
6. Describe the fabrication process of CMOS.

Module-4(HDL):

1. State the difference between hardware and software language.
2. What is the difference between synthesis and simulation?
3. What is the difference between variable and signal?
4. What is '9 value logic'?
5. Design D latch and D flipflop in dataflow style.
6. Write a short note on derived datatypes.

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lecture-wise Plan

Subject Name:-Control System

Subject Code:-CS705C

Year:4th Year

Semester: -Seventh

Module No.	Topics	Planned Lectures(H)
	<u>INTRODUCTION TO CONTROL SYSTEM:</u>	5 H
1.	a) . Concept of feedback and Automatic control, Effects of feedback, Definition of linear and nonlinear systems, Elementary concepts of sensitivity and robustness. Types of control systems	01
	b) Objectives of control system, Servomechanisms and regulators, examples of feedback control systems	02
	c) Transfer function concept. Pole and Zeroes of a transfer function. Properties of Transfer function	02
	<u>MATHEMATICAL MODELING OF DYNAMIC SYSTEMS:</u>	5H
2.	a) Translational systems, Rotational systems, Mechanical coupling, Liquid level systems	02
	b) Electrical analogy of Spring–Mass–Dashpot system	01
	c) Block diagram representation of control systems. Block diagram algebra. Signal flow graph. Mason's gain formula	02
	<u>CONTROL SYSTEM COMPONENTS:</u>	4H
3.	a) Potentiometer, Synchros, Resolvers, Position encoders. DC and AC tachogenerators. Actuators. Block diagram level description of feedback control systems for position control	02
	b) speed control of DC motors, temperature control, liquid level control, voltage control of an Alternator.	02
	<u>TIME DOMAIN ANALYSIS:</u>	6H
4.	a) Time domain analysis of a standard second order closed loop system. Concept of undamped natural frequency	02
	b) damping, overshoot, rise time and settling time. Dependence of time domain performance parameters on natural frequency and damping ratio.	01
	c) Step and Impulse response of first and second order systems. Effects of Pole and Zeros on transient response.	02

	Stability by pole location. d) Routh-Hurwitz criteria and applications	01
5.	<u>ERROR ANALYSIS:</u> a) Steady state errors in control systems due to step, ramp and parabolic inputs. Concepts of system types and error constants	3H 03
6.	<u>STABILITY ANALYSIS:</u> a) Root locus techniques, construction of Root Loci for simple systems. Effects of gain on the movement of Pole and Zeros	3H 03
Module No.	TOPICS	Planed Lectures
7.	<u>FREQUENCY DOMAIN ANALYSIS OF LINEAR SYSTEM:</u> a) Bode plots b) Polar plots, Nichols chart, Concept of resonance frequency of peak magnification c) Nyquist criteria, measure of relative stability, phase and gain margin d) Determination of margins in Bode plot. Nichols chart. M-circle and M-Contours in Nichols chart.	7h 03 02 02 01
8.	<u>CONTROL SYSTEM PERFORMANCE MEASURE:</u> a) Improvement of system performance through compensation. Lead, Lag and Lead- lag compensation, PI, PD and PID control	3H 03
TOTAL HOUR REQUIRED=36 h		

ASSIGNMENTS:

MODULE1:

1. What is the effect of adding feedback to a control system?
2. Explain sensitivity and robustness.
3. What are the differences between open and closed loop control system?
4. What is stochastic and adaptive control system?

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lecture-wise Plan

Subject Name:-Control System

Subject Code:-CS705C

Year:4th Year

Semester: -Seventh

MODULE2 & 3:

1. What is an electrical analogous of spring, mass and damper system? Explain.
2. What are the prime differences between block diagram reduction and signal flow graph?
3. Write a short note on potentiometer.
4. Write a short note on resolver.
5. Write a short note on synchro.
6. What are the major methods for the speed control of DC motor? Explain the methods.

MODULE 4:

1. What is natural frequency of oscillation?
2. Explain damping ratio.
3. Deduce the expression for step response in a second order system.
4. Deduce the expression for peak time, peak overshoot and rise time.
5. Analyse the system from stability point of view with the help of Routh – Hurwitz criteria:

$$S^6+2s^5+5s^4+3s^3+s^2+s+4 = 0$$

MODULE 5:

1. Explain the terms: Position error constant, velocity error constant, acceleration error constant.
2. What is the significance of steady state error?
3. What is the effect on adding a pole or a zero in a transfer function?

MODULE 6:

1. What is root locus?
2. What is break away and break in point?
3. How the gain of the system varies with the variation of pole and zero?
4. What is an asymptote? Explain the significance.

MODULE 7:

1. What is gain and phase margin? How they effect stability?
2. What is the significance of gain crossover frequency and phase crossover frequency?
3. What is resonant frequency and bandwidth?
4. Explain Nyquist's criterion.
5. Compare relative stability with absolute stability.

MODULE 8:

1. Write short notes on lead and lag compensator.
2. Explain the significance of P, PI and PID controllers.

Faculty In-Charge

HOD, EE Dept.

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lecture-wise Plan

Subject Name: Modelling & Simulation

Year: 4thYear

Subject Code-CS705D

Semester: Seventh

Module Number	Topics	Number of Lectures
1	Unit 1: Introduction to Modelling and Simulation:	6L
	Nature of Simulation. Systems, Models and Simulation, Continuous and Discrete Systems, system modelling, Components of a simulation study, Introduction to Static and Dynamic System simulation, Application areas, Advantages, Disadvantages and pitfalls of Simulation.	6
2	Unit 2: System Dynamics & Probability concepts in Simulation:	10L
	Exponential growth and decay models, Generalization of growth models, Discrete and Continuous probability functions, Continuous Uniformly Distributed Random Numbers, Generation of a Random numbers, Generating Discrete distributions, Non-Uniform Continuously Distributed Random Numbers, Rejection Method.	10
3	Unit 3: Simulation of Queuing Systems and Discrete System Simulation:	14L
	Poisson arrival patterns, Exponential distribution, Service times, Normal Distribution Queuing Disciplines, Simulation of single and two server queue. Application of queuing theory in computer system. Discrete Events, Generation of arrival patterns, Simulation programming tasks, gathering statistics, measuring occupancy and Utilization, Recording Distributions and Transit times.	14
4	Unit 4: Analysis of Simulation output:	6L
	Sensitivity Analysis, Validation of Model Results	6
Total Number Of Hours = 36		

Faculty In-Charge

HOD, CSE Dept.

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

Title of Course: Image Processing Lab

Course Code: CS793A

L-T-P Scheme: 0-0-3

Course Credits: 2

Objectives:

Digital image processing deals with manipulation of digital images through a digital computer. It is a subfield of signals and systems but focus particularly on images. DIP focuses on developing a computer system that is able to perform processing on an image. The input of that system is a digital image and the system process that image using efficient algorithms, and gives an image as an output.

Learning Outcomes:

Students will be able to apply various image processing concepts and models to input images or input signals for various purposes. For example :image compression, image de-noising, image enhancement, edge detection and sharpening etc.

Course Contents:

List of Experiments

1. Display of Grayscale Images.
2. Histogram Equalization.
3. Non-linear Filtering.
4. Edge detection using Operators.
5. 2-D DFT and DCT.
6. Filtering in frequency domain.
7. Display of color images.
8. conversion between color spaces.
9. DWT of images.
10. Segmentation using watershed transform.

REFERENCE:

1. Rafael C. Gonzalez, Richard E. Woods, Steven Eddins,' Digital Image Processing using MATLAB', Pearson Education, Inc., 2004.

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

LIST OF EQUIPMENTS:

Computer, Software MATLAB

Text Books

1. Digital Image Processing, Gonzalves, Pearson
2. Digital Image Processing, Jahne, Springer India
3. Digital Image Processing & Analysis, Chanda & Majumder, PHI
4. Fundamentals of Digital Image Processing, Jain, PHI
5. Image Processing, Analysis & Machine Vision, Sonka, VIKAS
6. Getting Started with GIS- Clarke Keith. C; PE.
7. Concepts & Techniques of GIS - Lo C.P, Albert, Yeung K.W- PHI.

List of Experiments

1. Display of Grayscale Images.
2. Histogram Equalization.
3. Non-linear Filtering.
4. Edge detection using Operators.
5. 2-D DFT and DCT.
6. Filtering in frequency domain.
7. Display of color images.
8. conversion between color spaces.
9. DWT of images.
10. Segmentation using watershed transform.

REFERENCE:

1. Rafael C. Gonzalez, Richard E. Woods, Steven Eddins, ' Digital Image Processing using MATLAB', Pearson Education, Inc., 2004.

LIST OF EQUIPMENTS:

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

Computer, Software MATLAB

Experiment No.1 Display of Gray scale Images.

Aim:

To display the Gray scale images.

Apparatus Required:

Computer, Matlab Software

Syntax

`imshow(I)`

`imshow(I,[low high])`

`imshow(RGB)`

`imshow(BW)`

`imshow(X,map)`

`imshow(filename)`

`himage = imshow(...)`

`imshow(..., param1, val1, param2, val2,...)`

Theory:

`imshow(I)` displays the grayscale image I.

`imshow(I,[low high])` displays the grayscale image I, specifying the display range for I in [low high]. The value low (and any value less than low) displays as black; the value high (and any value greater than high) displays as white. Values in between are displayed as intermediate shades of gray, using the default number of gray levels. If you use an empty matrix ([]) for [low high], `imshow` uses [`min(I(:))` `max(I(:))`]; that is, the minimum value in I is displayed as black, and the maximum value is displayed as white.

`imshow(RGB)` displays the truecolor image RGB.

`imshow(BW)` displays the binary image BW. `imshow` displays pixels with the value 0 (zero) as black and pixels with the value 1 as white.

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

`imshow(X,map)` displays the indexed image `X` with the colormap `map`. A color map matrix may have any number of rows, but it must have exactly 3 columns. Each row is interpreted as a color, with the first element specifying the intensity of red light, the second green, and the third blue. Color intensity can be specified on the interval 0.0 to 1.0.

`imshow(filename)` displays the image stored in the graphics file `filename`. The file must contain an image that can be read by `imread` or `dicomread`. `imshow` calls `imread` or `dicomread` to read the image from the file, but does not store the image data in the MATLAB workspace. If the file contains multiple images, `imshow` displays the first image in the file. The file must be in the current directory or on the MATLAB path.

`himage = imshow(...)` returns the handle to the image object created by `imshow`.

`imshow(..., param1, val1, param2, val2,...)` displays the image, specifying parameters and corresponding values that control various aspects of the image display.

Converting RGB Image into gray scale image & extracting the color Spaces

```
image1=imread('dse_college.jpg');
```

```
image2=rgb2gray (image1);
```

```
[r c d]=size (image1);
```

```
z=zeros(r,c);
```

```
tempr=image1;
```

```
tempr(:,2)=z;
```

```
tempr(:,3)=z;
```

```
imshow(tempr)
```

```
tempg=image1;
```

```
tempg(:,1)=z;
```

```
tempg(:,3)=z;
```

```
imshow(tempg)
```

```
tempb=image1;
```

```
tempb(:,1)=z;
```

```
tempb(:,2)=z;
```


UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

`imshow(tempb)`

Result:

Thus the gray scale image is displayed.

Experiment No.2 Histogram Equalization.

Aim:

To enhance contrast using Histogram Equalization.

Apparatus Required:

Computer, Matlab Software

Syntax

`J = histeq(I, hgram)`

`J = histeq(I, n)`

`[J, T] = histeq(I,...)`

`newmap = histeq(X, map, hgram)`

`newmap = histeq(X, map)`

`[newmap, T] = histeq(X,...)`

Theory

histeq enhances the contrast of images by transforming the values in an intensity image, or the values in the colormap of an indexed image, so that the histogram of the output image approximately matches a specified histogram.

`J = histeq(I, hgram)` transforms the intensity image `I` so that the histogram of the output intensity image `J` with `length(hgram)` bins approximately matches `hgram`.

histeq automatically scales `hgram` so that `sum(hgram) = prod(size(I))`. The histogram of `J` will better match `hgram` when `length(hgram)` is much smaller than the number of discrete levels in `I`.

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

$J = \text{histeq}(I, n)$ transforms the intensity image I , returning in J an intensity image with n discrete gray levels. A roughly equal number of pixels is mapped to each of the n levels in J , so that the histogram of J is approximately flat. (The histogram of J is flatter when n is much smaller than the number of discrete levels in I .) The default value for n is 64.

$[J, T] = \text{histeq}(I, \dots)$ returns the grayscale transformation that maps gray levels in the image I to gray levels in J .

$\text{newmap} = \text{histeq}(X, \text{map}, \text{hgram})$ transforms the colormap associated with the indexed image X so that the histogram of the gray component of the indexed image (X, newmap) approximately matches hgram . The histeq function returns the transformed colormap in newmap . $\text{length}(\text{hgram})$ must be the same as $\text{size}(\text{map}, 1)$.

$\text{newmap} = \text{histeq}(X, \text{map})$ transforms the values in the colormap so that the histogram of the gray component of the indexed image X is approximately flat. It returns the transformed colormap in newmap .

$[\text{newmap}, T] = \text{histeq}(X, \dots)$ returns the grayscale transformation T that maps the gray component of map to the gray component of newmap .

Examples

Enhance the contrast of an intensity image using histogram equalization.

```
I = imread('tire.tif');
```

```
J = histeq(I);
```

```
imshow(I)
```

```
figure, imshow(J)
```

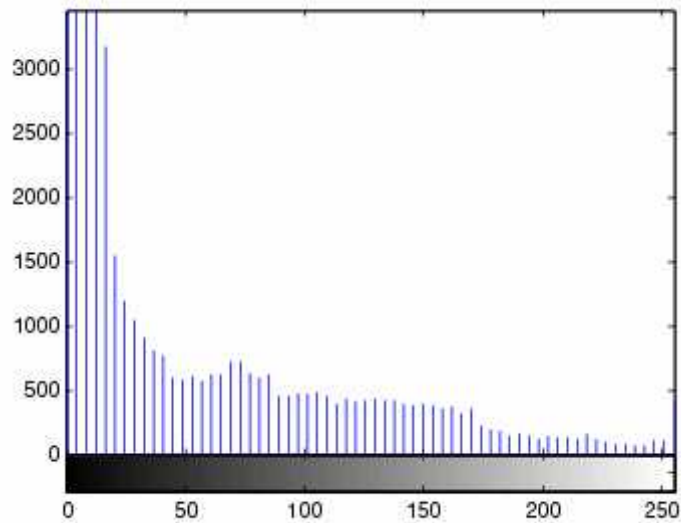


UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

Display a histogram of the original image.

```
figure; imhist(I,64)
```

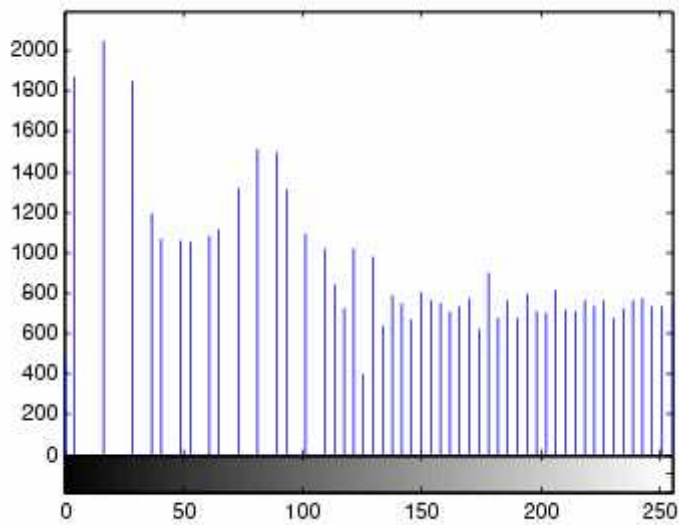


Compare it to a histogram of the processed image.

```
figure; imhist(J,64)
```

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual



Algorithm

When you supply a desired histogram `hgram`, `histeq` chooses the grayscale transformation T to minimize where $c0$ is the cumulative histogram of A , $c1$ is the cumulative sum of `hgram` for all intensities k . This minimization is subject to the constraints that T must be monotonic and $c1(T(a))$ cannot overshoot $c0(a)$ by more than half the distance between the histogram counts at a . `histeq` uses the transformation $b = T(a)$ to map the gray levels in X (or the colormap) to their new values. If you do not specify `hgram`, `histeq` creates a flat `hgram`,

```
hgram = ones(1,n)*prod(size(A))/n;
```

Result

The histogram equalization is done.

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

Experiment No.3 Edge detection using Operators.

Aim:

To detect the edge of the Gray scale images.

Apparatus Required:

Computer, Matlab Software

Syntax

```
To demonstrate edge detection
% numbers of colors
sncols=128;
ncols=32;
% get image from MATLAB image
load('trees');
% show original image
figure(1);
showimg(real(X),sncols);
drawnow;
% construct convolution functions
[m,n] = size(X);
gs = [1 -1]; ge = [];
hs = [1 -1]; he = [];
g = [gs,zeros(1,m-length(gs)-length(ge)),ge];
h = [hs,zeros(1,n-length(hs)-length(he)),he];
% construct convolution matrices as sparse matrices
Y = spcnvmat(g);
Z = spcnvmat(h);
Wg = Y*X;
Wh = X*Z';
```

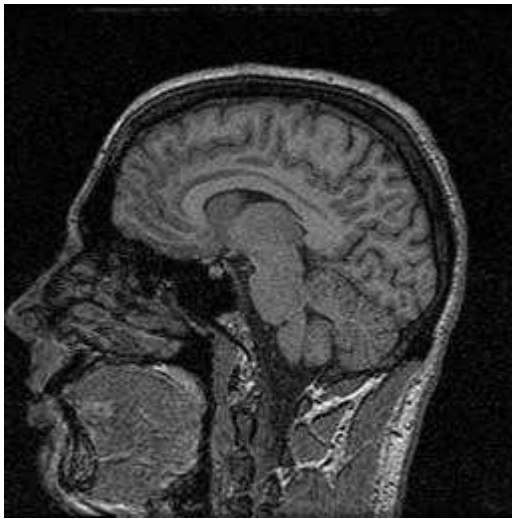
UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

```
% show transformed images
figure(2);
showgimg(Wg,ncols);
drawnow;
figure(3)
showgimg(Wh,ncols);
drawnow;
figure(4)
showgimg(abs(Wg)+abs(Wh),ncols);
drawnow;
```

Theory

Edges characterize boundaries and are therefore a problem of fundamental importance in image processing. Edges in images are areas with strong intensity contrasts – a jump in intensity from one pixel to the next. Edge detecting an image significantly reduces the amount of data and filters out useless information, while preserving the important structural properties in an image. There are many ways to perform edge detection. However, the majority of different methods may be grouped into two categories, gradient and Laplacian. The gradient method detects the edges by looking for the maximum and minimum in the first derivative of the image. The Laplacian method searches for zero crossings in the second derivative of the image to find edges. An edge has the one-dimensional shape of a ramp and calculating the derivative of the image can highlight its location. Suppose we have the following signal, with an edge shown by the jump in intensity below: The intensity changes thus discovered in each of the channels are then represented by oriented primitives called zero-crossing segments, and evidence is given that this representation is complete. (2) Intensity changes in images arise from surface discontinuities or from reflectance or illumination boundaries, and these all have the property that they are spatially localized. Because of this, the zero-crossing segments from the different channels are not independent, and rules are deduced for combining them into a description of the image. This description is called the raw primal sketch.



Result

The edge detection of the image is done.

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

Title of Course: Pattern Recognition Lab

Course Code: CS793B

L-T-P scheme: 0-0-3

Course Credit: 2

Objectives:

To introduce the most important concepts, techniques, and algorithms for digital image processing, and implement them using image processing software tools, particularly MATLAB. More specifically, it should enable students to:

-) Assess and understand the challenges behind the design of machine vision systems.
-) Understand the general processes of image acquisition, storage, enhancement, segmentation, representation, and description.
-) Implement filtering and enhancement algorithms for monochrome as well as color images.

Appreciate the challenges and understand the principles and applications of visual pattern recognition.

Learning Outcomes:

1. To implement efficient algorithms for nearest neighbour classification.
2. To construct decision trees.
3. To implement of Linear Discriminate Function and Support Vector Machines.
4. Formulate and describe various applications in pattern recognition
5. Understand the Bayesian approach to pattern recognition
6. Be able to mathematically derive, construct, and utilize Bayesian-based classifiers, and non-Bayesian classifiers both theoretically and practically.
7. Be able to identify the strengths and weaknesses of different types of classifiers
8. Understand basic concepts such as the central limit theorem, the curse of dimensionality, the bias-variance dilemma, and cross-validation
9. Validate and assess different clustering techniques
10. Apply various dimensionality reduction methods whether through feature selection or feature extraction
11. Assess classifier complexity and regularization parameters
12. Be able to combine various classifiers using fixed rules or trained combiners and boost their performance
13. Understand the possibilities and limitations of pattern recognition

Course Contents:

Exercises that must be done in this course are listed below:

Exercise No.1: Data visualization, central limit theorem, multivariate normal distribution, data whitening, non-parametric density estimation: Parzen, nearest neighbour.

Exercise No. 2: Forward selection, backward selection, take l-add-r selection, branch & bound, genetic algorithms. PCA, Fisher mapping, nonlinear feature extraction, multidimensional scaling, dissimilarity representation.

Exercise No. 3: Hierarchical clustering, k-means, fuzzy c-means, Gaussian mixture model, expectation-maximization, Davies-Bouldin index, self-organizing maps.

Exercise No. 4: Implementation of Bayesian classifier, Parzen classifier, k-NN classifier, logistic classifier, quadratic/linear/nearest-mean classifiers, and Fisher classifier. Curse of dimensionality.

Exercise No. 5: Linear regression, MMSE, MAP, MLE, quality measures. Nonlinear regression: kernel smoothing/local weighted regression.

Exercise No. 6: Simulate Banker's Algorithm for Dead Lock PreventiSVM, ANN, ensemble classification complexity: bias-variance trade-off improving performance (implement either

Text Book:

1. Sergios Theodoridis, Pattern Recognition, 4th edition, Elsevier, 2009.
2. Richard O. Duda, Peter E. Hart and David G. Stork, Pattern Classification, 2nd edition, Wiley Interscience, 2001.

Recommended Systems/Software Requirements:

1. Basic knowledge in multi-variables Calculus and Engineering Mathematics.
2. Basic knowledge in Linear Algebra.
3. Fundamentals of Probability theory and Statistics.
4. Programming knowledge of MATLAB or C+.

Experiment No: 1

Aim: To browse Genomic databases using Map Viewer & ensembl, viewing genetic, linkage maps for human and other model organisms.

REQUIREMENT: Computer system with (legal software) equipped with Internet Connection preferably fast Broadband.

WEB RESOURCES USED: <http://www.ncbi.nlm.nih.gov/>
<http://www.ebi.ac.uk/embl.html>

THEORY AND PRINCIPLE:

Genomes: A genome is all of a living thing's genetic material. It is the entire set of hereditary instructions for building, running, and maintaining an organism, and passing life on to the next generation. In short, it is the complete set of chromosomes with all the genes (for diploid organisms, often it is given as a haploid genome) for that species.

Genomic Resources: The organism's genomic resources are stored as Genome databases and these are a collection of complete and incomplete large-scale sequencing, assembly, annotation and mapping projects for cellular organisms. The genome database provides views for a variety of genomes, complete chromosomes, sequence maps and integrated genetic and physical maps, organelles, plasmids as well as genome assemblies.

Map Viewer: It is a tool to visualize integrated views of chromosome maps for many organisms and is particularly useful for identification and localization of genes and other biological features.

Ensembl: Ensembl is a genome browser project developed by EMBL-EBI and the Sanger Institute. It provides free access to all the data produced by the project as well as the tools used to analyze and present data. It has software system that produces and maintains automatic annotation on selected eukaryotic genomes.

Model Organisms: Model organisms are the much-studied organisms having one or more characteristics that makes it suitable for laboratory study viz., having small genome, easy to handle in culture or otherwise, abundantly available, and harmless to humans and other organisms (preferably).

Genetic Maps: Genetic maps are the position of the genes relative to each other, and relative to the ends and centre of the chromosome and other sequence features on a genome. These are based on genetic techniques like cross-breeding experiments like meiotic recombination frequencies [in case of humans based on pedigrees (family histories)]. Genes serve as the markers along the length of the chromosomes. Linkage maps are one example. Other DNA markers include RFLPs (Restriction Fragment Length Polymorphisms), SSLPs (Simple Sequence Length Polymorphisms) and SNPs (Single Nucleotide Polymorphisms).

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

based on the fact that linked genes tend to remain together and less likely to participate in a recombination event. With help of genetic markers linkage maps are constructed and provide estimate of distance between two genes.

Genome Maps: A genome map is a graphical representation that provides information about the location and the sequence of genes along the length of each chromosome and the distance between two adjacent genes in a genome (linkage maps for all the linkage groups in a genome of a species)

PROCEDURE:

1. Start the computer and establish Internet connection.
2. Use any search engine like Yahoo / Google or otherwise directly open NCBI/EBI web page.,
3. Double click on map viewer to study genetic and linkage maps of human and other model organisms. The submission Window of Ensembl Genome Browser –

RESULTS: The results are shown in the frames.

REMARKS: The genome data base or genetic / linkage maps for human and other model organisms were studied by using tools like MapViewer and Ensemble to present and use the data.

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

Title of Course: Soft Computing Lab

Course Code: CS793C

L-T-P scheme: 0-0-3

Course Credit: 2

Objectives:

This course introduces soft computing techniques that are different from conventional AI techniques. This course also provides necessary mathematical background for understanding and implementing soft computing Techniques, such as neural networks, fuzzy systems, and genetic algorithms. This course

Learning Outcomes:

1. Understand importance of soft computing.
2. Understand different soft computing techniques like Genetic Algorithms, Fuzzy Logic, Neural Networks and their combination.
3. Implement algorithms based on soft computing.
4. Apply soft computing techniques to solve engineering or real life problems.

Course Contents:

Exercises that must be done in this course are listed below:

Experiment 1: Write a program in MATLAB to plot various membership functions.

Experiment 2: Use Fuzzy toolbox to model tip value that is given after a dinner which can be-not good, satisfying, good and delightful and service which is poor, average or good and the tip value will range from Rs. 10 to 100.

Experiment 3: Implement FIS Editor.

Experiment 4: Generate AND, NOT function using McCulloch-Pitts neural net by MATLAB program.

Experiment 5: Generate XOR function using McCulloch-Pitts neural net by MATLAB program.

Experiment 6: Write a MATLAB program for Perceptron net for an AND function with bipolar inputs and targets.

Experiment 7: Write a MATLAB program for Hebb Net to classify two dimensional input patterns in bipolar with their given targets

Experiment 8: Write a program of Perceptron Training Algorithm

Experiment 9: Write a program to implement Hebb's rule

Experiment 10: Write a program of Back Propagation Algorithm.

Text Books

1. Fuzzy logic with engineering applications, Timothy J. Ross, John Wiley and Sons.
2. S. Rajasekaran and G.A.V.Pai, "Neural Networks, Fuzzy Logic and Genetic Algorithms", PHI.
3. Principles of Soft Computing, S N Sivanandam, S. Sumathi, John Wiley & Sons
4. Genetic Algorithms in search, Optimization & Machine Learning by David E. Goldberg
5. Neuro-Fuzzy and Soft computing, Jang, Sun, Mizutani, PHI
6. Neural Networks: A Classroom Approach, 1/e by Kumar Satish, TMH
7. Genetic Algorithms in search, Optimization & Machine Learning by David E. Goldberg, Pearson/PHI
8. A beginners approach to Soft Computing, Samir Roy & Udit Chakraborty, Pearson

Recommended Systems/Software Requirements:

1. In this laboratory the students need to implement the soft computing tools in Matlab. Some exposure in C also can be used for neural network and Genetic Algorithm.

Experiment No: 1: Membership Functions

Aim: Write a program in MATLAB to plot various membership functions.

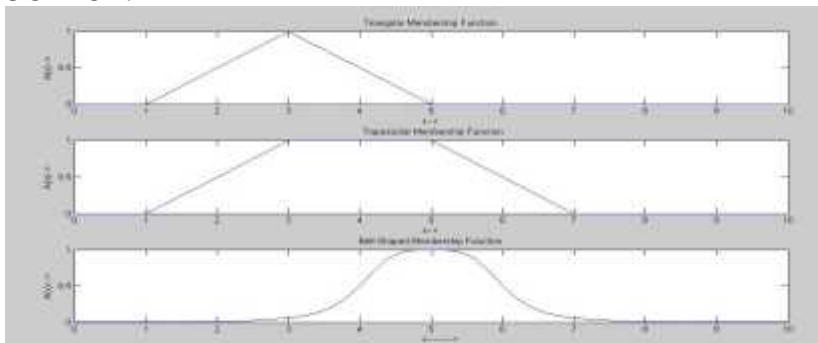
Description:

The membership function of a fuzzy set is a generalization of the indicator function in classical sets. In fuzzy logic, it represents the degree of truth as an extension of valuation. Degrees of truth are often confused with probabilities, although they are conceptually distinct, because fuzzy truth represents membership in vaguely defined sets, not likelihood of some event or condition. Membership functions were introduced by Zadeh in the first paper on fuzzy sets (1965). Zadeh, in his theory of fuzzy sets, proposed using a membership function (with a range covering the interval (0,1)) operating on the domain of all possible values.

/* Program in MATLAB to plot various membership functions */

```
%Triangular Membership Function
x=(0.0:1.0:10.0)';
y1=trimf(x, [1 3 5]);
subplot(311)
plot(x,[y1]);
%Trapezoidal Membership Function
x=(0.0:1.0:10.0)';
y1=trapmf(x, [1 3 5 7]);
subplot(312)
plot(x,[y1]);
%Bell-Shaped Membership Function
x=(0.0:0.2:10.0)';
y1=gbellmf(x, [1 2 5]);
subplot(313)
plot(x,[y1]);
```

OUTPUT:



Experiment No: 2: Use Fuzzy toolbox

Aim: Use Fuzzy toolbox to model tip value that is given after a dinner which can be-not good, satisfying, good and delightful and service which is poor, average or good and the tip value will range from Rs. 10 to 100.

Description:

The toolbox lets you model complex system behaviors using simple logic rules, and then implement these rules in a fuzzy inference system. You can use it as a stand-alone fuzzy inference engine. Alternatively, you can use fuzzy inference blocks in Simulink and simulate the fuzzy systems within a comprehensive model of the entire dynamic system.

/* Process */

We are given the linguistic variables quality of food and service as input variables which can be written as:

Quality(not good,satisfying,good,delightful)

Service(poor,average,good)

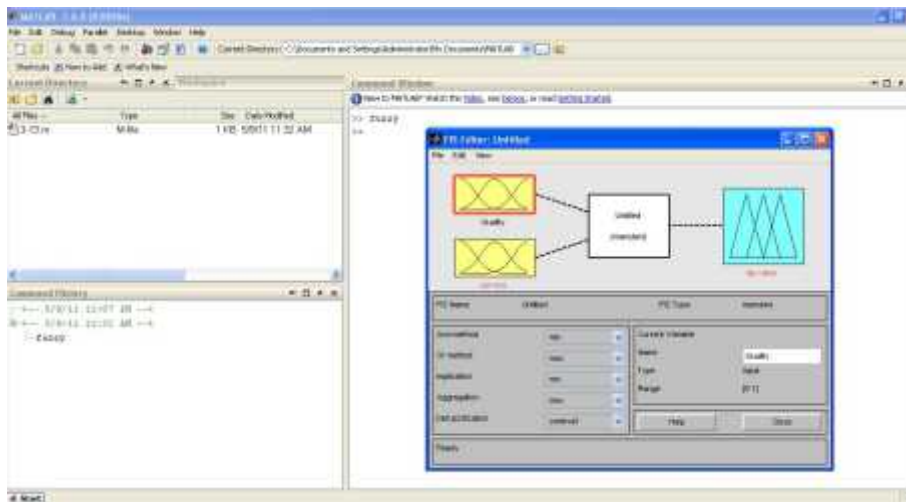
UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

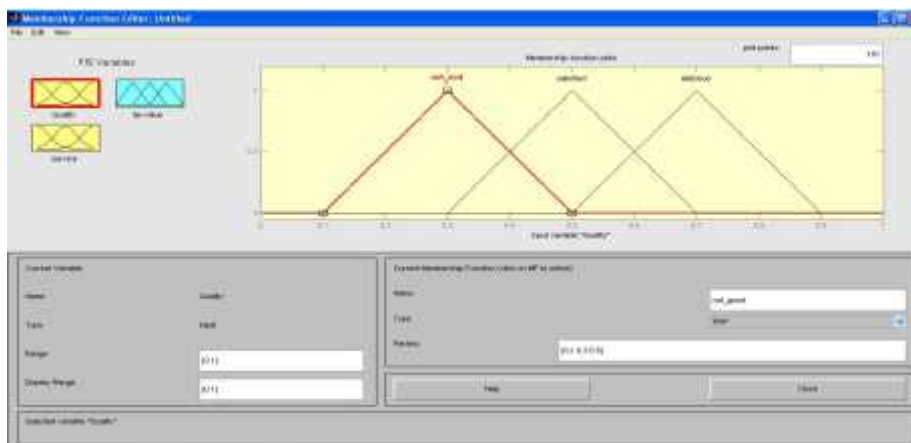
A Fuzzy system comprises the following modules:-

1. Fuzzification Interface
2. Fuzzy Inference Engine
3. Defuzzification Interface

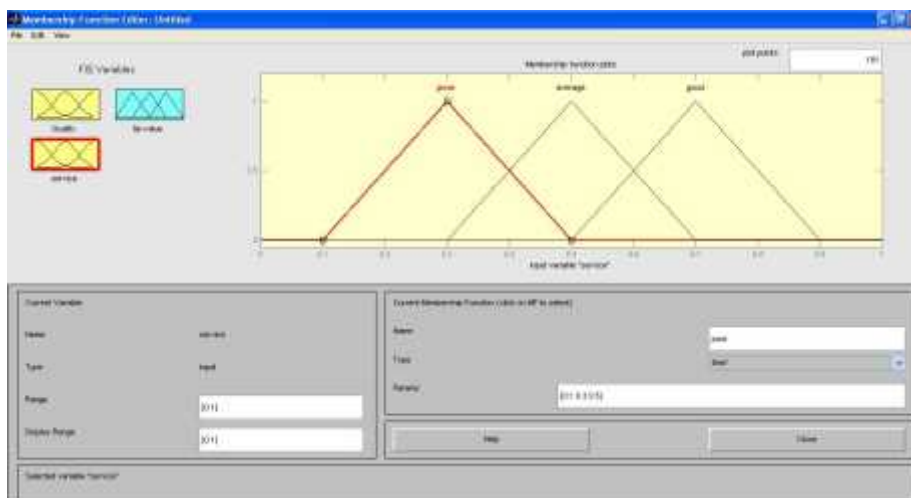
Fuzzy sets are defined on each of the universe of discourse:-
Quality, service and tip value.

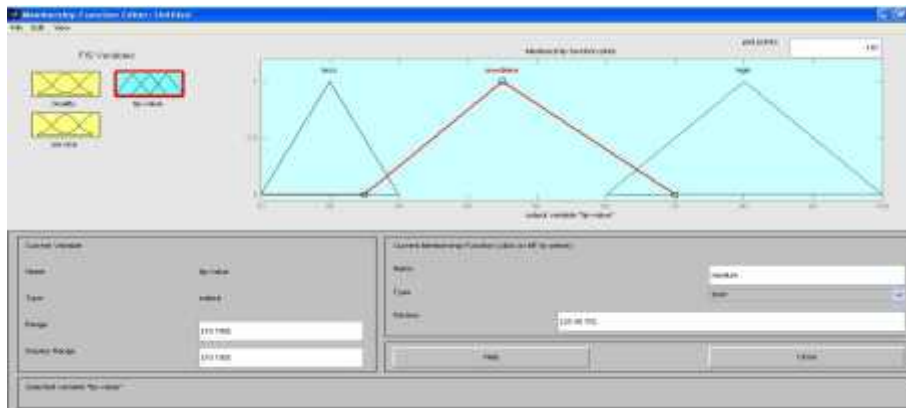


The values for Quality variable are selected for their respective ranges:



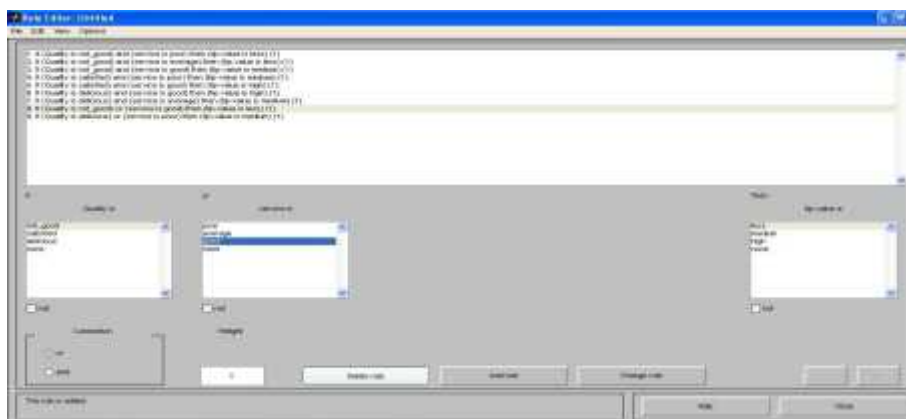
Similarly values for Service variable are selected for their respective ranges :-





In general a compositional rule for inference involves the following procedure:

1. Compute memberships of current inputs in the relevant antecedent fuzzy set of rule.
2. If the antecedents are in conjunctive form, the AND operation is replaced by a minimum, if OR then by Maximum and similarly other operations are performed.
3. Scale or clip the consequent fuzzy set of the rule by a minimum value found in step 2 since this gives the smallest degree to which the rule must fire.
4. Repeat steps 1-3 for each rule in the rule base. Superpose the scaled or clipped consequent fuzzy sets formed by such a superposition. There are numerous variants of the defuzzifications.

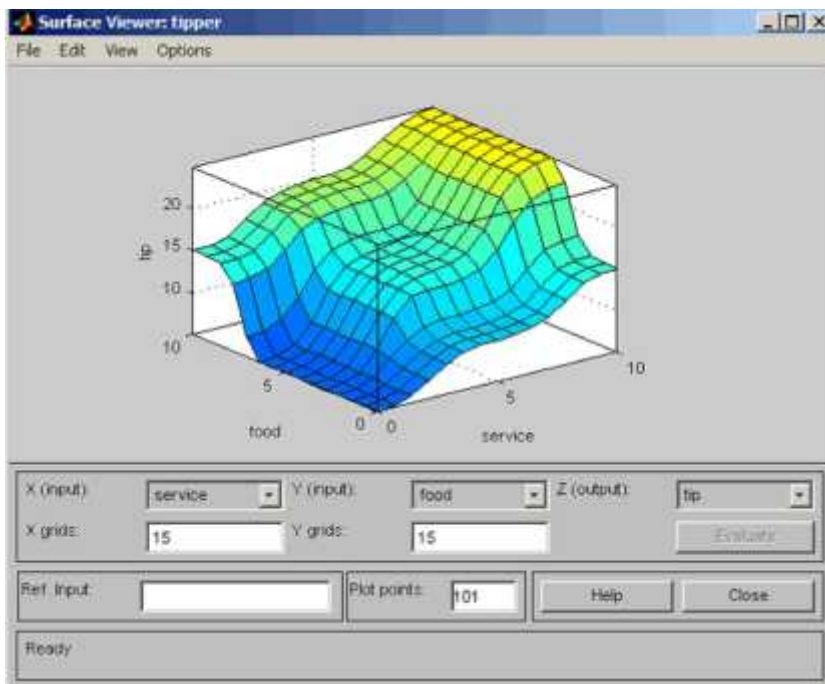


OUTPUT:



UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual



Experiment No: 3: FIS Editor

Aim: Implement FIS Editor.

Description:

FIS stands for Fuzzy Inference System. In FIS fuzzy rules are used for approximate reasoning. It is the logical framework that allows us to design reasoning systems based on fuzzy set theory.

/* Process */

To illustrate these concepts we use example of Water Tank:

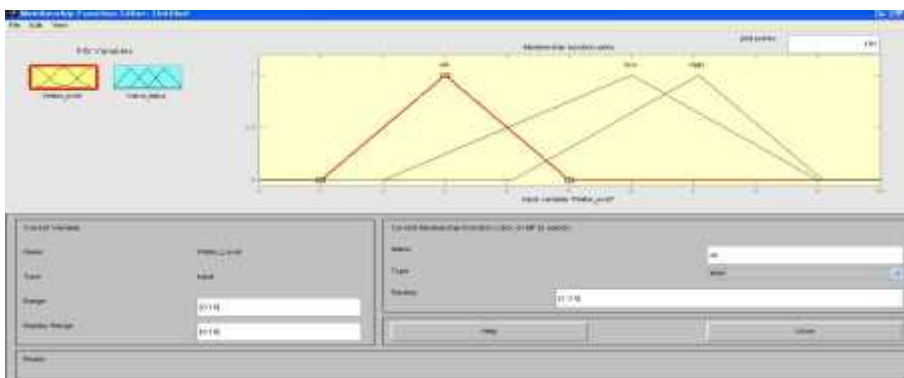
FIS editor consists of following units:

- i) Input

The Water Level is considered as the Input variable and Valve status is taken as Output Variable.

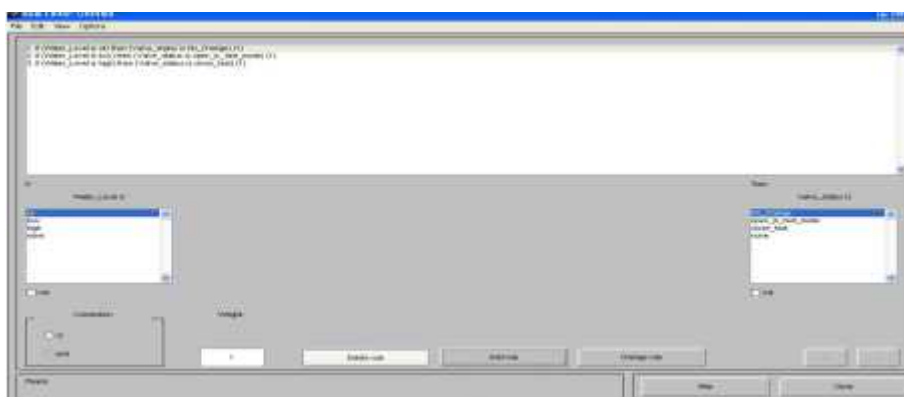


The Input-Output Variable's Membership functions should be plotted along with their ranges:-



The following screen appearance is obtained by clicking on the FIS Rule system indicator:
Rules are added by selecting variable's values and clicking on add rule menu each time a new rule is added.

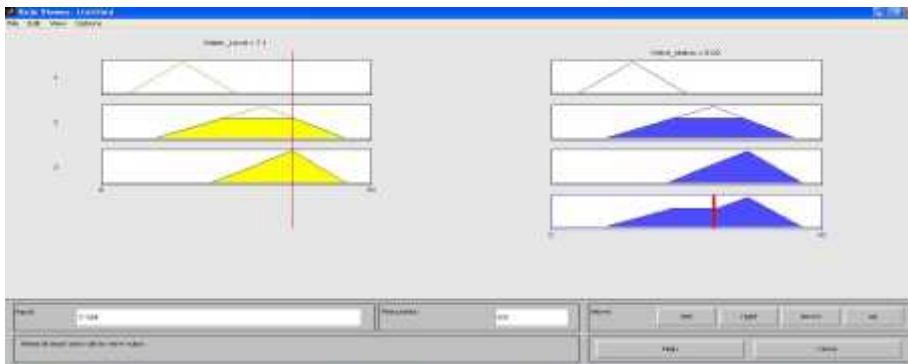
The fuzzy Rules defined for water tank are:
IF level is ok, THEN there is no change in valve.
IF level is low, THEN valve is open in fast mode.
IF level is high, THEN valve is closed in fast mode.



The result is displayed as plots of input-output membership functions :-

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

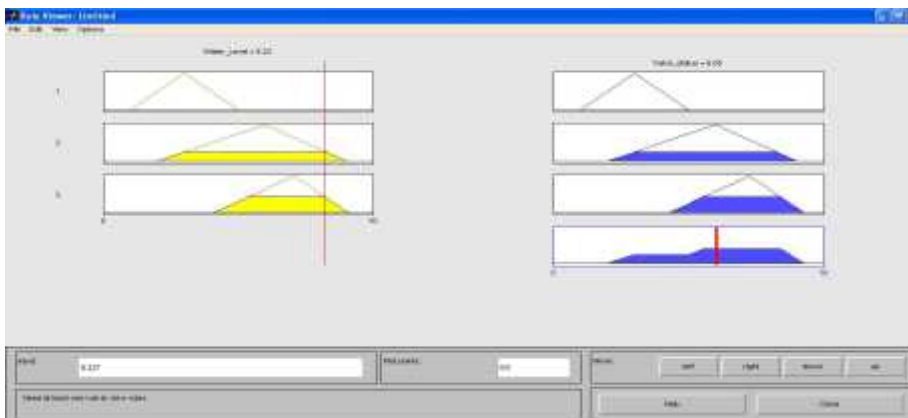
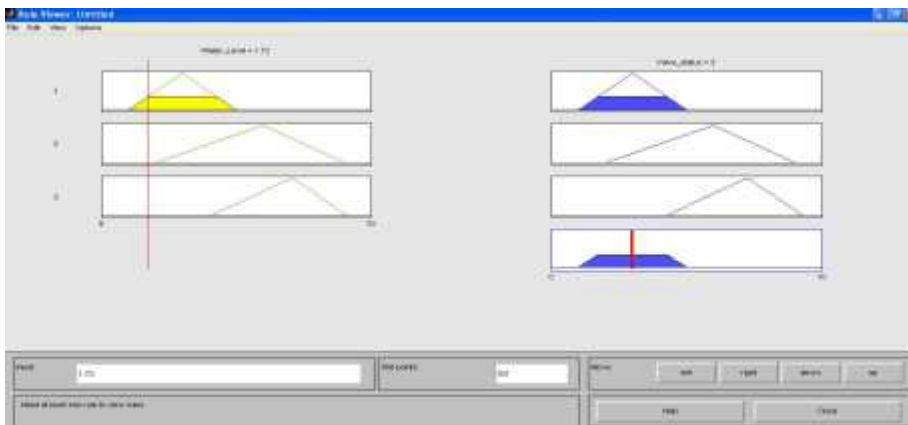
Lab Manual



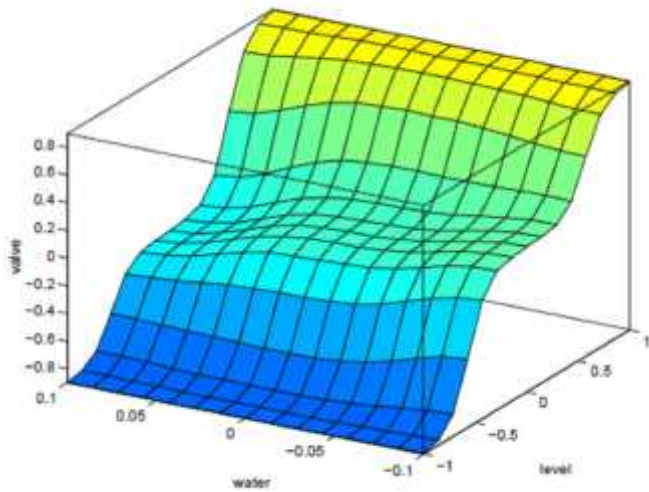
Water Level(ok,low,high)

Valve Status(no change,open fast,closed fast)

The output in accordance with the input and rules provided by user is shown as(view-rule viewer):



OUTPUT:



Experiment No: 4: Generate AND, NOT function using McCulloch-Pitts neural net.

Aim: Generate AND, NOT function using McCulloch-Pitts neural net by MATLAB program.

Description:

We can use McCulloch-Pitts neurons to implement the basic logic gates. All we need to do is find the appropriate connection weights and neuron thresholds to produce the right outputs for each set of inputs. We shall see explicitly how one can construct simple networks that perform NOT, AND, and OR. It is then a well known result from logic that we can construct any logical function from these three operations. The resulting networks, however, will usually have a much more complex architecture than a simple Perceptron. We generally want to avoid decomposing complex problems into simple logic gates, by finding the weights and thresholds that work directly in a Perceptron architecture.

Logical **AND** function

patterns (bipolar) decision boundary

x1	x2	y	w1 = 1
-1	-1	-1	w2 = 1
-1	1	-1	b = -1
1	-1	-1	t = 0
1	1	1	$-1 + x1 + x2 = 0$

/* Program to Generate ANDNOT function using McCulloch-Pitts neural net */

%ANDNOT function using McCulloch-Pitts neuron

clear;

clc;

% Getting weights and threshold value

disp('Enter the weights');

w1=input('Weight w1=');

w2=input('Weight w2=');

disp('Enter threshold value');

theta=input('theta=');

y=[0 0 0 0];

x1=[0 0 1 1];

x2=[0 1 0 1];

z=[0 0 1 0];

con=1;

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

```
zin = x1*w1+x2*w2;
for i=1:4
if zin(i)>=theta
y(i)=1;
else y(i)=0;
end
end
disp('Output of net=');
disp(y);
if y==z
con=0;
else
disp('Net is not learning Enter another set of weights and threshold value');
w1=input('Weight w1=');
w2=input('Weight w2=');
thete=input('theta=');
end
end
disp('McCulloch Pitts Net for ANDNOT function');
disp('Weights of neuron');
disp(w1);
disp(w2);
disp('Threshold value=');
disp(theta);
```

INPUT:

Enter the weights
Weight w1=1
Weight w2=1
Enter threshold value
theta=1

OUTPUT:

Output of net= 0 1 1 1
Net is not learning Enter another set of weights and threshold value
Weight w1=1
Weight w2=-1
theta=1
Output of net=0 0 1 0
McCulloch Pitts Net for ANDNOT function
Weights of neuron
1
-1
Threshold value=
1

Experiment No: 5: Generate XOR function using McCulloch-Pitts neural net

Aim: Generate XOR function using McCulloch-Pitts neural net by MATLAB program.

Description:

We can use McCulloch-Pitts neurons to implement the basic logic gates. All we need to do is find the appropriate connection weights and neuron thresholds to produce the right outputs for each set of inputs. We shall see explicitly how one can construct simple networks that perform NOT, AND, and OR. It is then a well known result from logic that we can construct any logical function from these three operations. The resulting networks, however, will usually have a much more

problems into simple logic gates, by finding the weights and thresholds that work directly in a Perceptron architecture.

Logical **XOR** (exclusive OR) function

patterns (bipolar) decision boundary

x1	x2	y
-1	-1	-1
-1	1	1
1	-1	1
1	1	-1

No line can separate these two classes, as can be seen from the fact that the following linear inequality system has no solution

because we have $b < 0$ from
(1) + (4), and $b \geq 0$ from
(2) + (3), which is a
contradiction

/* Program to Generate XOR function using McCulloch-Pitts neural net by MATLAB

program.*/

% XOR function using McCulloch-Pitts neuron

clear;

clc;

% Getting weights and threshold value

disp('Enter the weights');

w11=input('Weight w11=');

w12=input('Weight w12=');

w21=input('Weight w21=');

w22=input('Weight w22=');

v1=input('Weight v1=');

v2=input('Weight v2=');

disp('Enter threshold value');

theta=input('theta=');

x1=[0 0 1 1];

x2=[0 1 0 1];

z=[0 1 1 0];

con=1;

while con

zin1 = x1*w11+x2*w21;

zin2 = x1*w21+x2*w22;

for i=1:4

if zin1(i)>=theta

y1(i)=1;

else y1(i)=0;

end

if zin2(i)>=theta

y2(i)=1;

else y2(i)=0;

end

end

yin=y1*v1+y2*v2;

for i=1:4

if yin(i)>=theta;

y(i)=1;

else

y(i)=0;

end

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

```
disp('Output of net=');
disp(y);
if y==z
con=0;
else
disp('Net is not learning Enter another set of weights and threshold value');
w11=input('Weight w11=');
w12=input('Weight w12=');
w21=input('Weight w21=');
w22=input('Weight w22=');
v1=input('Weight v1=');
v2=input('Weight v2=');
theta=input('theta=');
end
end
disp('McCulloch Pitts Net for XOR function');
disp('Weights of neuron Z1');
disp(w11);
disp(w21);
disp('Weights of neuron Z2');
disp(w12);
disp(w22);
disp('Weights of neuron Y');
disp(v1);
disp(v2);
disp('Threshold value=');
disp(theta);
```

INPUT:

Enter the weights
Weight w11=1
Weight w12=-1
Weight w21=-1
Weight w22=1
Weight v1=1
Weight v2=1
Enter threshold value
theta=1

OUTPUT:

Output of net= 0 1 1 0
McCulloch Pitts Net for XOR function
Weights of neuron z1
1
-1
Weights of neuron z2
-1
1
Weights of neuron y
1 1
Threshold value= 1

Experiment No: 6: Perceptron net of an AND function with bipolar inputs and targets

Aim: Write a MATLAB program for Perceptron net for an AND function with bipolar inputs and targets.

Description:

In machine learning, the perceptron is an algorithm for supervised learning of binary classifiers (functions that can decide whether an input, represented by a vector of numbers, belongs to some specific class or not). It is a type of linear classifier, i.e. a classification algorithm that makes its predictions based on a linear predictor function combining a set of weights with the feature vector. The algorithm allows for online learning, in that it processes elements in the training set one at a time.

/* Program of Perceptron net for an AND function with bipolar inputs and targets */

```
% Perceptron for AND Function
clear;
clc;
x=[1 1 -1 -1;1 -1 1 -1];
t=[1 -1 -1 -1];
w=[0 0];
b=0;
alpha=input('Enter Learning rate=');
theta=input('Enter Threshold Value=');
con=1;
epoch=0;
while con
    con=0;
    for i=1:4
        yin=b+x(1,i)*w(1)+x(2,i)*w(2);
        if yin>theta
            y=1;
        end
        if yin<=theta & yin>=-theta
            y=0;
        end
        if yin<-theta
            y=-1;
        end
        if y-t(i)
            con=1;
            for j=1:2
                w(j)=w(j)+alpha*t(i)*x(j,i);
            end
            b=b+alpha*t(i);
        end
    end
    epoch=epoch+1;
end
disp('Perceptron for AND Function');
disp('Final Weight Matrix');
disp(w);
disp('Final Bias');
disp(b);
```

INPUT:

Enter Learning rate=1
Enter Threshold Value=0.5

OUTPUT:

Perceptron for AND Function
Final Weight Matrix
1 1

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

Experiment No: 7: Hebb Net to classify two dimensional input patterns in bipolar with their targets.

Aim: Write a MATLAB program for Hebb Net to classify two dimensional input patterns in bipolar with their targets given below:

‘*’ indicates a ‘+’ and ‘.’ Indicates ‘-’

```
*****          *****
*....          *....
*****          *****
*....          *....
*****          *
```

Description:

Hebb, in his influential book *The organization of Behavior* (1949), claimed

- Behavior changes are primarily due to the changes of synaptic strengths () between neurons I and j
- increases only when both I and j are “on”: the Hebbian learning law
- In ANN, Hebbian law can be stated: increases only if the outputs of both units and have the same sign.

$$\Delta w_{ij} = w_{ij}(\text{new}) - w_{ij}(\text{old}) = x_i y$$

$$\text{or, } \Delta w_{ij} = w_{ij}(\text{new}) - w_{ij}(\text{old}) = \Gamma x_i y$$

Step 0: Initialization: $b = 0$, $w_i = 0$, $i = 1$ to n

Step 1: For each of the training sample $s:t$ do steps 2 -4

/* s is the input pattern, t the target output of the sample */

Step 2: $x_i := s_i$, $i = 1$ to n /* set s to input units */

Step 3: $y := t$ /* set y to the target */

Step 4: $w_i := w_i + x_i * y$, $i = 1$ to n /* update weight */

$b := b + x_i * y$ /* update bias */

/* Program for Hebb Net to classify two dimensional input patterns in bipolar with their targets */

% Hebb Net to classify Two -Dimensional input patterns.

clear;

clc;

% Input Pattern

E=[1 1 1 1 1 -1 -1 -1 1 1 1 1 1 -1 -1 -1 1 1 1 1];

F=[1 1 1 1 1 -1 -1 1 1 1 1 1 1 -1 -1 -1 1 -1 -1 -1];

X(1,1:20)=E;

X(2,1:20)=F;

w(1:20)=0;

t=[1 -1];

b=0;

for i=1:2

w=w+X(i,1:20)*t(i);

b=b+t(i);

end

disp('Weight Matrix');

disp(w);

disp('Bias');

disp(b);

OUTPUT:

? Weight Matrix

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2 2 2

Bias

0

Experiment No: 8: Genetic Algorithm

Aim: Write the algorithm of Genetic Algorithm.

Description:

Genetic Algorithm is a search heuristic (experience) that follows the process of natural evolution. This heuristic is used to generate useful solutions to optimization and search problems. Genetic Algorithm belong to the larger class of evolutionary algorithm (EA) which generate solutions to optimization problems and using techniques inspired by natural evolution like – inheritance, mutation, selection, crossover. Genetic Algorithm need design space to be converted into genetic space. Genetic Algorithm works with coding variables. Genetic Algorithm uses population of point at one time in contrast to the single point approach. It means that genetic algorithm processes a number of designs at the same time. The advantage of coding of variable is that coding discretizes the search space even though the function may be continuous. Traditional optimization methods use transition rules that are deterministic in nature. While genetic algorithm uses randomize operators. Randomize operator improve the search space in an adaptive manner.

There are three important aspects of Genetic Algorithm are:

2. Definition of objective function.
3. Definition and implementation of genetic representation.
4. Definition and implementation of Genetic operators.

Advantages of Genetic Algorithm (GA):

1. It shows simplicity.
2. Ease of operation.
3. Minimal requirement.
4. Global perspective.
5. It does not guarantee to find global minimum solutions but acceptably good solutions to "acceptably quickly".

/* Genetic Algorithm */

Genetic Algorithm Steps :

1. BEGIN
2. Create initial population.
3. Compute fitness of each individuals.
4. WHILE NOT finished DO Loop
5. BEGIN
6. Select individuals from old generation for mating.
7. Create offspring by applying crossover or mutation to the selected individuals.
8. Compute fitness of new individuals.
9. Kill old individuals to make a room for new chromosomes and insert offspring in the new generation.
10. If population has converged
11. Then fitness=TRUE.
12. END
13. END

Experiment No: 9: Perceptron Training Algorithm

Aim: Write a program of Perceptron Training Algorithm

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

Start with a randomly chosen weight vector w_0 ;
Let $k=1$;
While there exists input vector that are misclassified by: W_{k-1} do
Let i be a misclassified input vector
Let $X_k = \text{class}(ij)$, implying that $W_{k-1} \cdot X_k < 0$
Update the weight vector to $W_k = W_{k-1} + nX_k$;
increment k ;
End while;

/* Program of Perceptron Training Algorithm */

```
#include<iostream.h>
#include<conio.h>
Void main( )
{
clrscr( );
int in[3],d,w[3],a=0;
for(int i=0;i<3;i++)
{
cout<<"\n initialize the weight vector w"<<i;
cin>>w[i]
}
for(i=0;i<3;i++)
{
cout<<"\n enter the input vector i"<<i;
cin>>in[i];
}
cout<<"\n enter the desired output";
cin>>d;
int ans=1;
while(ans== 1)
{
for (a= 0, i=0;i<3;i++)
{
a = a + w[i] * in[i];
}
clrscr( );
cout<<"\n desired output is"<<d;
cout<<"\n actual output is "<<a;
int e;
e=d-a;
cout<<"\n error is "<<e;
cout<<"\n press 1 to adjust weight else 0";
cin>>ans;
if (e<0)
{
for(i=0;i<3;i++)
{
w[i]=w[i]-1;
}
}
else if (e>0)
{
for(i=0;i<3;i++)
{
w[i]=w[i]+1;
}
```

```

}
getch( );
}

```

OUTPUT:

```

desire output is 2
actual output is 17
error is -15
press 1 to adjust weight else 0

```

Experiment No: 9: Hebb's rule

Aim: Write a program to implement Hebb's rule

Description:

The Hebb rule determines the change in the weight connection from u_i to u_j by $\Delta w_{ij} = r * a_i * a_j$, where r is the learning rate and a_i, a_j represent the activations of u_i and u_j respectively. Thus, if both u_i and u_j are activated the weight of the connection from u_i to u_j should be increased. Examples can be given of input/output associations which can be learned by a two-layer Hebb rule pattern associator. In fact, it can be proved that if the set of input patterns used in training are mutually orthogonal, the association can be learned by a two-layer pattern associator using Hebbian learning. However, if the set of input patterns are not mutually orthogonal, interference may occur and the network may not be able to learn associations. This limitation of Hebbian learning can be overcome by using the delta rule.

/* Program to implement Hebb's rule */

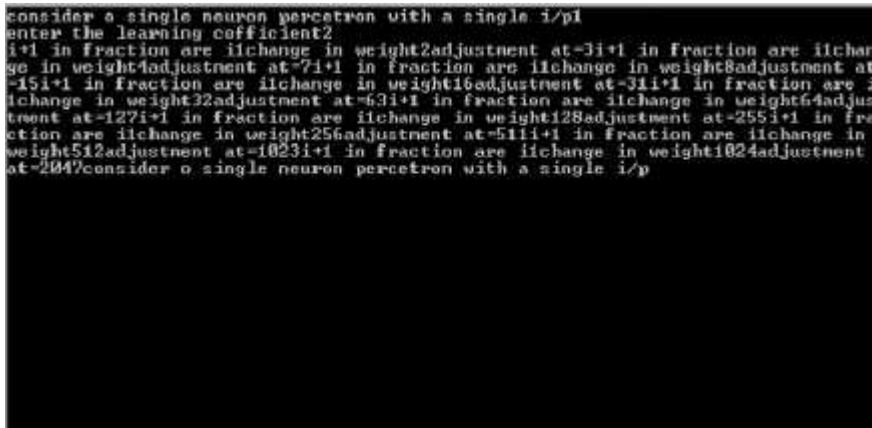
```

#include<iostream.h>
#include<conio.h>
void main()
{
float n,w,t,net,div,a,al;
cout<<"consider o single neuron percetron with a single i/p";
cin>>w;
cout<<"enter the learning cofficient";
cin>>d;
for (i=0;i<10;i++)
{
net = x+w;
if(wt<0)
a=0;
else
a=1;
div=at+a+w;
w=w+div;
cout<<"i+1 in fraction are i"<<a<<"change in weight"<<dw<<"adjustment at="<<w;
}
}

```

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual



```
consider a single neuron perceptron with a single i/p
enter the learning coefficient2
i+1 in fraction are ilchange in weight2adjustment at-3i+1 in fraction are ilchange
in weight4adjustment at-7i+1 in fraction are ilchange in weight8adjustment at
-15i+1 in fraction are ilchange in weight16adjustment at-31i+1 in fraction are ilchange
in weight32adjustment at-63i+1 in fraction are ilchange in weight64adjustment
at-127i+1 in fraction are ilchange in weight128adjustment at-255i+1 in fraction are ilchange
in weight256adjustment at-511i+1 in fraction are ilchange in weight512adjustment at-1023i+1 in fraction are ilchange
in weight1024adjustment at-2047consider a single neuron perceptron with a single i/p
```

Experiment No: 10: Back Propagation Algorithm

Aim: Write a program of Back Propagation Algorithm.

Description:

Backpropagation is a kind of neural network. A Neural Network (or artificial neural network) is a collection of interconnected processing elements or nodes. The nodes are termed simulated neurons as they attempt to imitate the functions of biological neurons. The nodes are connected together via links. We can compare this with axon-synapse-dendrite connections in the human brain. Initially, a weight is assigned at random to each link in order to determine the strength of one node's influence on the other. When the sum of input values reaches a threshold value, the node will produce the output 1 or 0 otherwise. By adjusting the weights the desired output can be obtained. This training process makes the network learn. The network, in other words, acquires knowledge in much the same way human brains acquire namely learning from experience. Backpropagation is one of the powerful artificial neural network technique which is used acquire knowledge automatically.

/* Program of Back Propagation Algorithm */

```
#include <iostream.h>
#include <conio.h>
void main ()
{
    int i ;
    float delta, com, coeff = 0.1;
    struct input
    {
        float val,out,wo, wi;
        int top;
    } s[3] ;
    cout<< "\n Enter the i/p value to target o/p" << "\t";
    for (i=0; i<3 ; i++)
        cin>> s [i], val>> s[i], top);
    i = 0;
    do
    {
        if (i == 0)
        {
            W0 = -1.0;
            W1 = -0.3;
        }
        else
```

```

W1 = del [i - 1] , Wi ;
}
del [i]. aop = w0 + (wi * del [i]. val);
del [i].out = del [i]. aop);
delta = (top - del [i]. out) * del [i].out * (1 - del [i].out);
corr = coeff * delta * del [i].out];
del [i].w0 = w1 + corr;
del [i]. w1 = w1 + corr;
i++;
}while ( i != 3)
cout<< "VALUE"<<"Target"<<"Actual"<<"w0" <<"w1"<<'\\n';
for (i=0; i=3; i++)
{
cout<< s [i].val<< s[i].top<<s[i].out << s[i]. w0<< s[i]. w1;
cout<< "\\n";
}
getch ();
}

```

OUTPUT:

```

backpropagation network
1. load data
2. learn from data
3. compute output pattern
4. make new data file
5. save data
6. print data
7. change learning rate
8. exit
Enter your choice (1-8)

```

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

Title of Course: Artificial Intelligence Lab

Course Code: CS793D

L-T-P Scheme: 0-0-2

Course Credit: 2

Objectives: In this course we will implement the basic components of an intelligent system, their functions, mechanisms, policies and techniques used in their implementation and examples.

Learning Outcomes: The students will have a detailed knowledge of the concepts of artificial intelligence. Various applications of AI in different fields, aware of a variety of approaches to AI techniques.

Course Contents:

Unit-1: Introduction to AI and intelligent agents

Unit-2: Problem solving, Problem spaces and blind search techniques, informed search techniques, Constraint satisfaction problems

Unit-3: Knowledge representation and reasoning techniques, Logic programming, Logical agents, Game playing, planning,

Unit-4: Learning, Reasoning under uncertain situations,

Unit-5: Expert systems, Decision support systems, Domain specific AI applications.

List of AI Problems for Lab-

Problem 1: Solve “Tower of Hanoi” with only 2 disks.

Problem 2: Solve “Tower of Hanoi” with only 3 disks.

Problem 3: Solve “4-Queens” puzzle.

Problem 4: Solve “8-Queens” puzzle.

Problem 5: Solve “4-color map” problem.

Problem 6: Solve “8 – puzzle” take any initial and goal state.

Problem 7: Calculate the sum of n elements in an integer array. Also calculate its Polynomial function the determine its complexity using “*Big-O*”.

Problem 8: Find out the largest element in an square 2-D array. Also determine the “Big- O” of the algorithm. [Take size greater the 2x2]

Problem 9: Solve “Latin Square” problem.

Problem 10: Solve “Sudoku Problem” use any initial positions.

Problem 11: Solve “15-puzzle” problem using any initial and goal state.

Problem 12: Solve “Sudoku Problem” use any initial positions.

Problem 13: Code the following games software: Checkers, Chess.

Problem 14: Code the following games using software: Othello, Backgammon.

Problem 15: Code the following games using software: Bridge, Go.

Problem 16: Code the following games using software: Hex, 6x7.

Problem 17: Code the following games using software: Tetris, Tick-Tack-Toe.

Problem 18: Code the following games using software: rubik-cube, same game, mines.

Problem 19: Code the following games using software: Matches, Mines.

Text Books

1. Rich, Elaine Knight, Kevin, Artificial Intelligence, Tata McGraw Hill.

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

2. Luger, George F, Artificial Intelligence: Structures and Strategies for Complex Problem Solving, Pearson Education.

References

1. Nilsson, Nils J, Artificial Intelligence, Morgan Kaufmann
2. Russell, Stuart J. Norvig, Peter, Artificial Intelligence: A Modern Approach, Pearson Education
3. Pearson Education
4. Negnevitsky, Michael, Artificial Intelligence: A Guide to Intelligent Systems, Addison-Wesley.

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

Title of Course: Internet Technology Lab

Course Code: CS795

L-T-P scheme: 0-0-3

Course Credit: 2

Objectives:

To develop the ability to design and implement web enabled applications.

Learning Outcomes:

The student shall acquire the skill to design and develop web based applications with high usability, scalability and efficiency. They shall be exposed to various technologies required to design web sites. They shall acquire the skill to choose the technology to use based on the requirements and functionality of the web site.

Course Contents:

Exercises that must be done in this course are listed below:

Exercise No.1: Design a web page using HTML, CSS and use JavaScript for validation.

Exercise No.2: Use Image tag.

Exercise No.3: Use PHP to connect with MySql database for some operation.

Exercise No.4: Use different database operation.

Text Book:

1. "Web Enabled commercial Application development using HTML,DHTML, Java Script", Perl CGI" by Ivan Bayross, BPB Publication
2. "Internet and World Wide Web – How to Program" by Deitel, Deitel and Nieto ,Pearson Education Asia Publication
3. "PHP and MYSQL Manual" by Simon Stobart and Mike Vassileiou
4. "PHP and MYSQL Web Development" by Luke Welling and Laura Thomson(Pearson Education
5. "The XML Bible", by Elliotte Rusty Harold
6. "Step by Step XML" by Michael J. Young Prentice Hall Of India
7. "XML How to Programme" Deitel Pearson Edition
8. "XML Hand Book" 3rd Edition Pearson Edition

Recommended Systems/Software Requirements:

1. Editor and browser.
2. Apache server, MySQL server.

Exercise No.1: Design a web page using HTML, CSS and use JavaScript for validation.

Aim: Design a login page using HTML, CSS and use JavaScript for validation.

Description: HTML marks the content up into different structural types, like paragraphs, blocks, lists, images, tables, forms, comments etc.

CSS tells the browser how each type of element should be displayed, which may vary for different media (like screen, print or handheld device).

JavaScript tells the browser how to change the web page in response to events that happen (like clicking on something, or changing the value in a form input).

HTML, CSS and JavaScript code is given below:

/*HTML, CSS, JavaScript Code*/

```
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en-gb" lang="en-gb" dir="ltr" >
```

```
<head>
```

```
<title>Student</title>
```

```
<!-- css main start-->
```

```
<link rel="stylesheet" type="text/css" href="css/sadmin.css" />
```

```
<!-- css main end -->
```

```

        function validate(){
            if(document.frm.txtStudentId.value==""){
                alert("Please Enter Student ID!");
                return false;
            }else if(document.frm.txtPasswd.value==""){
                alert("Please Enter Password!");
                return false;
            }else{
                return false;
            }
        }
    </script>
</head>
<body class="body">
    <!-- main div start -->
    <div class="main_div">
        <!-- top div start -->
        <div class="main_sub_top_div">
            <!-- top-top-top div start -->
            <div class="main_sub_top_top_div">
                <!-- Header Right-->
            </div>
            <!-- top-top-top div start -->
            <!-- top-top-mid div start -->
            <div class="main_sub_top_mid_div">
                <!-- top-top-mid-left div start -->
                <div class="main_sub_top_mid_left_div">

                    </div>
                    <div class="main_sub_top_mid_right_div">
                        Student Login Area</div>
                    <!-- top-top-mid-left div start -->
                </div>
                <!-- top-top-mid div start -->
            <!-- top-top-bottom div start -->
            <!--div class="main_sub_top_bottom_div">
                <!-- menu bar start -->

                <!-- menu bar end -->
            <!--/div-->
            <!-- top-top-bottom div start -->
        </div>
        <!-- top div end -->
        <!-- mid div start -->
        <div class="main_sub_mid_div">
            <div class="main_sub_mid_left_div">
                <div class="main_content_div">
                    This is for Student Login
                </div>
            </div>
        </div>
        <div class="main_sub_mid_mid_div">
            <form name="frm" method="post" action="<? $_SERVER['PHP_SELF']?>"
onSubmit="return validate()">
                <input type="hidden" name="action" value="login" />
                <table width="100%">
                    <tr>
                        <td width="20%">

```


UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

```
<td width="80%">
  <table width="100%">
    <tr>
      <td width="20%">
        Enrollment No.
      </td>
      <td width="80%">
        <input type="text" name="txtStudentId" />
      </td>
    <tr>
    <tr>
      <td width="20%">
        Password
      </td>
      <td width="80%">
        <input type="password" name="txtPasswd" />
      </td>
    <tr>
    <tr>
      <td>
      </td>
      <td colspan="2">
        <input type="submit" name="txtSubmit" class="button" value="Login" />
        <?=$msg?>
      </td>
    <tr>
    </table>
  </td>
</tr>
</table>

    </form>
  </div>
</div>

<!-- mid div end -->
<!-- buttom div start -->
<div class="main_sub_buttom_div">
  <!-- top-top div start -->
  <div class="main_sub_buttom_buttom_div">
    <!-- links start for footer-->
    <font color="#000000">

                                &copy;by UEM, Jaipur(Department Of CSE)
                                </font>

    <!-- links end --></div>
  <!-- top-top div start -->
</div>
<!-- buttom div end -->
</div>
<!-- main div end -->
</body>
</html>
```

/*CSS Code*/

```
.body{
    text-align:center;
    /*background:#1A1411:*/
```

```

        border:2px thick #000;
    }
    .main_div{
        width:100%;
        height:auto;
        margin:0 auto;
        clear:both;
        border:2px solid #000;
    }

    .main_sub_mid_div{
        width:100%;
        height:auto;
        margin:0 auto;
        clear:both;
        text-align:left;
    }
    .main_sub_mid_left_div{
        width:20%;
        height:auto;
        float:left;
        padding:0px;
        /*margin:0 0 0 2px; user for margin*/
    }
    .main_sub_mid_mid_div{
        width:79%;
        height:auto;
        float:left;
        background-color:#FFF;
        padding:5px;
        text-align:justify;
    }
    .main_sub_bottom_div{
        width:100%;
        height:auto;
        margin:0 auto;
        clear:both;
        border:2px solid #000;
        background-color:FCF;
    }
    .main_sub_top_top_div{
        width:100%;
        height:20px;
        margin:0 auto;
        clear:both;
        background-color:#FCF;
        text-align:right;
        vertical-align:middle;
        border-bottom:1px solid;
        border-color:#000;
        color:#EEE;
        border:none;
    }
    .main_sub_top_mid_div{
        width:100%;
        height:50px;
        margin:0 auto;

```

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

```
}  
.main_sub_top_mid_left_div{  
    width:20%;  
    height:50px;  
    background-color:#FCF;  
    float:left;  
    text-align:left;  
}  
.main_sub_top_mid_right_div{  
    width:80%;  
    height:40px;  
    background-color:#FCF;  
    float:left;  
    padding-top:10px;  
    text-align:center;  
    vertical-align:middle;  
    font-size:32px;  
    font: bold Arial, Helvetica, sans-serif;  
    color:#000;  
    border:none;  
}  
.main_sub_bottom_bottom_div{  
    width:100%;  
    height:auto;  
    margin:0;  
    padding:0;  
    clear:both;  
    background-color:#FCF;  
    text-align:center;  
    color:#DDD;  
    text-decoration:none;  
    text-transform:  
}  
.button{  
    width:125px;  
    height:25px;  
    border:none;  
    background-color:#FCF;  
    color:#000;  
}
```

OUTPUT:



Exercise No.2: Use Image tag.

Aim: Use Image in the previous login page.

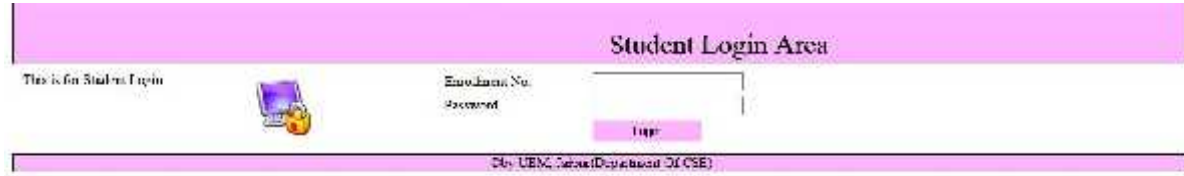
Description: The tag defines an image in an HTML page. The tag has two required attributes: src and alt. Images are not technically inserted into an HTML page, images are linked to HTML pages. The tag creates a holding space for the referenced image.

```

/*Image Code*/
<td width="20%">
    
</td>

```

OUTPUT:



Exercise No.3: Use PHP to connect with MySql database for some operation.

Aim: Use PHP and MySql to validate Student Id and Password with Existing Student data.

Description: PHP is a script language and interpreter that is freely available and used primarily on Linux Web servers. PHP, originally derived from Personal Home Page Tools, now stands for PHP: Hypertext Preprocessor, which the PHP FAQ describes as a "recursive acronym".

SQL stands for Structured Query Language. It lets you access and manipulate databases and it is an ANSI (American National Standards Institute) standard.

What Can SQL do?

1. SQL can execute queries against a database
2. SQL can retrieve data from a database
3. SQL can insert records in a database
4. SQL can update records in a database
5. SQL can delete records from a database
6. SQL can create new databases
7. SQL can create new tables in a database
8. SQL can create stored procedures in a database
9. SQL can create views in a database
10. SQL can set permissions on tables, procedures, and views

In computer science, a database connection is the means by which a database server and its client software communicate with each other. The term is used whether or not the client and the server are on different machines.

The client uses a database connection to send commands to and receive replies from the server. A database is stored as a file or a set of files on magnetic disk or tape, optical disk, or some other secondary storage device. The information in these files may be broken down into records, each of which consists of one or more fields.

Fields are the basic units of data storage, and each field typically contains information pertaining to one aspect or attribute of the entity described by the database. Records are also organized into tables that include information about relationships between its various fields. Although database is applied loosely to any collection of information in computer files, a database in the strict sense provides cross-referencing capabilities.

Connections are a key concept in data-centric programming. Since some DBMSs require considerable time to connect, connection pooling is used to improve performance. No command can be performed against a database without an "open and available" connection to it.

Connections are built by supplying an underlying driver or provider with a connection string, which is used to address a specific database or server and to provide instance and user authentication credentials (for example, Server=sql_box;Database=Common;User ID=uid;Pwd=password;).

Once a connection has been built, it can be opened and closed at will, and properties (such as the command time-out length, or transaction, if one exists) can be set. The connection string consists of a set of key-value pairs, dictated by the data access interface of the data provider.

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

database and a result set is returned, the connection is open but not available for other operations until the client finishes consuming the result set.

Other databases, such as SQL Server 2005 (and later), do not impose this limitation. However, databases that allow multiple concurrent operations on each connection usually incur far more overhead than those that only allow one operation at a time.

PHP, SQL and database connection code is given below:

/*php code to validate student*/

```
<?php
    $action=$_REQUEST['action'];
    $msg="";
    if($action=='login'){
        $studentId=$_REQUEST['txtStudentId'];
        $passwd=$_REQUEST['txtPasswd'];

        $sql="SELECT student_id, name, is_active, CONCAT(degree_id,'-',dept_id,'-',
batch_id) AS exam_group_id FROM dt_student WHERE student_id='".$studentId.'" AND
passwd='".$passwd.'"";

        $f_sql=f(q($sql));

        if($f_sql['student_id']){
            if($f_sql['is_active']==1){

                $_SESSION['student_id']=$f_sql['student_id'];
                $_SESSION['name']=$f_sql['name'];
                $_SESSION['exam_group_id']=$f_sql['exam_group_id'];

                session_write_close();
                header("Location:index.php");
                exit();
            }else if($f_sql['is_active']==0){
                $msg="<font                face=verdana                size=1
color=red><b>User is not activated</b>";
            }
        }else{
            $msg="<font                face=verdana                size=1                color=red><b>Invalid
username/password given</b>";
        }
    }
?>
```

/*Database connection Code*/

```
<?php

define("DB_NAME", $db_name);                // db name
define("DB_USER", $db_login);                // db username
define("DB_PASS", $db_pswd);                // db password
define("DB_HOST", $db_host);

mysql_connect(DB_HOST, DB_USER, DB_PASS) or die(mysql_error('could not connect to the
database'));

mysql_select_db(DB_NAME) or die(mysql_error('could not select the database'));
?>
```

OUTPUT:

The screenshot shows a web application interface with a pink header and footer. The header contains the text 'DIPAK KATRAL' and '17011003001063' on the left, and 'Student Panel' in the center, with 'Home' and 'Logout' links on the right. The main content area is divided into two sections. The left section has a 'info' tab and a 'View' button, with links for 'Assignment View', 'Student Status View', and 'Send Feedback'. The right section displays student details: 'Student ID: 17011003001063', 'Name: DIPAK KATRAL', 'Degree: B Tech', 'Department: CSE', 'Batch: 2015', and 'Email ID:'. Below these details is a 'Back' button. The footer contains 'Home' and 'Logout' links and the text 'Copyright © 2015. All Rights Reserved'.

Exercise No.4: Use different database operation.

Aim: Use different database operation of a student details.

Description: A transaction symbolizes a unit of work performed within a database management system (or similar system) against a database, and treated in a coherent and reliable way independent of other transactions. A transaction generally represents any change in database. Transactions in a database environment have two main purposes:

To provide reliable units of work that allow correct recovery from failures and keep a database consistent even in cases of system failure, when execution stops (completely or partially) and many operations upon a database remain uncompleted, with unclear status. To provide isolation between programs accessing a database concurrently. If this isolation is not provided, the programs' outcomes are possibly erroneous. A database transaction, by definition, must be atomic, consistent, isolated and durable.[1] Database practitioners often refer to these properties of database transactions using the acronym ACID.

Transactions provide an "all-or-nothing" proposition, stating that each work-unit performed in a database must either complete in its entirety or have no effect whatsoever. Further, the system must isolate each transaction from other transactions, results must conform to existing constraints in the database, and transactions that complete successfully must get written to durable storage.

Database operation code for insertion of student details is given below:

/*Database operation Code*/

```
<?php
$action=$_REQUEST['action'];
$valid=$_REQUEST['valid'];

if($action=="add" && $valid==1){
    $student_id=trim($_REQUEST['txtStudentId']);
    $name=trim($_REQUEST['txtName']);
    $degree_id=trim($_REQUEST['selDegree']);
    $dept_id=$_REQUEST['selDept'];
    $batch_id=$_REQUEST['selBatch'];
    $passwd=trim($_REQUEST['txtPasswd']);
    $email_id=trim($_REQUEST['txtEmailId']);

    $sql="INSERT INTO dt_student SET student_id='".addslashes($student_id)."',
name='".addslashes($name)."', degree_id='".addslashes($degree_id)."',
dept_id='".addslashes($dept_id)."',batch_id='".addslashes($batch_id)."',is_active=1,
passwd='".$passwd."', email_id='".$email_id."', dt = SYSDATE()";
?>
```

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

OUTPUT:

[Home](#) [Logout](#)

Administrator Panel

Student ID

Name

Degree

B.Tech ▾

Department

EEH ▾

Batch

2010 ▾

Password

Email ID

Save

Data has been saved successfully.

[Home](#) [Logout](#)

© 2016 Samrat (sham.besak@gmail.com)

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

Title of Course: Microelectronics & VLSI Design Lab

Course Code: CS795B

L-T-P scheme: 0-0-3

Course Credit: 2

Objectives: The overall course objective is to teach electrical engineering students fundamental concepts of hardware description languages and advanced techniques in digital system design. Specific objectives include the following:

1. Learn VHDL (Very high speed integrated circuit Hardware Description Language).
2. Utilize VHDL to design and analyze digital systems including arithmetic units and state machines.
3. Learn field programmable gate array (FPGA) technologies and utilize associated computer aided design (CAD) tools to synthesize and analyze digital systems.
4. Learn testing strategies and construct test-benches.
5. Conduct laboratory experiments using an FPGA based development board to prototype digital systems and to confirm the analysis done in class.
6. Prepare informative and organized lab reports that describe the methodologies employed, the results obtained, and the conclusions made in a laboratory experiment.

Learning Outcomes: The students will have a detailed knowledge of the concepts of IEEE and ANSI standard HDL. Upon the completion of Operating Systems practical course, the student will be able to:

-) **Understand** and implement basic digital logic circuits of VLSI.
-) **Model** complex digital systems at several levels of abstractions; behavioral and structural, synthesis and rapid system prototyping.
-) **Develop and Simulate** register-level models of hierarchical digital systems.
-) **Design and model** complex digital system independently or in a team
-) Carry out **implementations** of registers and counters.
-) **Simulate and synthesize** all type of digital logic circuits used in VLSI.
-) Finally **design** a CPU.

Course Contents:

Exercises that must be done in this course are listed below:

Exercise No.1: Design of basic Gates: AND, OR, NOT.

Exercise No. 2: Design of universal gates

Exercise No. 3: Design of XOR and XNOR gate.

Exercise No. 4: Design of 2:1 MUX .

Exercise No. 5: Design of 2 to 4 Decoder.

Exercise No. 6: Design of Half-Adder and Full Adder.

Exercise No. 7: Design of 8:3 Priority Encoder.

Exercise No. 8: Design of 4 Bit Binary to Grey code Converter.

Exercise No. 9: Design of all Flip-Flops.

Exercise No. 10: Design of Shift register.

Exercise No. 11: Design of ALU.

Text Book:

1. J. Bhaskar, A VHDL Primer, 3rd edition, Prentice Hall.

Recommended Systems/Software Requirements:

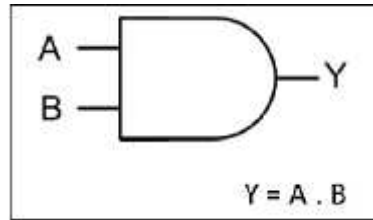
1. Intel based desktop PC with minimum of 1GHZ or faster processor with at least 1GB RAM and 8 GB free disk space.
2. Xilinx ISE14.2 software in Windows XP or Linux Operating System.

Experiment No: 1 Design of basic Gates: AND, OR, NOT.

Aim: Write VHDL code for designing AND, OR, NOT

AND Gate

2 Input AND gate		
A	B	A.B
0	0	0
0	1	0
1	0	0
1	1	1



VHDL codes:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity AND1 is
port (a : in STD_LOGIC; b : in STD_LOGIC; c : out STD_LOGIC) ;
end AND1;
architecture behavioral of AND1 is
begin
process (a, b)
begin
if (a= "1" and b="1")
then c<="1"; else c<="0";
end if;
end process;
end behavioral;
```

OR Gate:

2 Input OR gate		
A	B	A+B
0	0	0
0	1	1
1	0	1
1	1	1



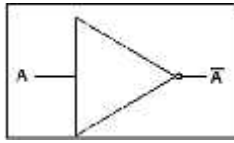
VHDL codes:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity OR1 is
port (a : in STD_LOGIC; b : in STD_LOGIC; c : out STD_LOGIC) ;
end OR1;
architecture behavioral of OR1 is
begin
process (a, b)
begin
if (a="0" and b="0") then c<= "0"; else c<="1";
end if;
end process;
```

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

NOT gate	
A	\bar{A}
0	1
1	0



VHDL Codes:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity NOT1 is
port (a : in STD_LOGIC; c : out STD_LOGIC) ;
end NOT1;
architecture behavioral of NOT1 is
begin
process (a)
begin
if (a="0") then c<="1";
else c<="0";
end if;
end process;
end behavioral;
```

Test Bench codes: Student will write/modify test bench codes in Xilinx ISE.

Output: Student will check the output.

Discussion: Student will conclude here.

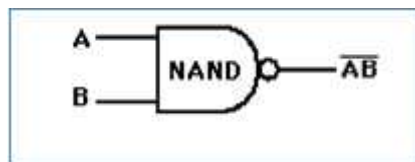
Experiment No.-2 : Design Universal gates

Aim: Write VHDL code for universal gates: NAND and NOR gate.

Apparatus: Xilinx ISE14.2 software

NAND gate:

2 Input NAND gate		
A	B	\overline{AB}
0	0	1
0	1	1
1	0	1
1	1	0



VHDL codes:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity NAND1 is
port (a : in STD_LOGIC; b : in STD_LOGIC; c : out STD_LOGIC) ;
end NAND1;
architecture
behavioral of NAND1 is
```

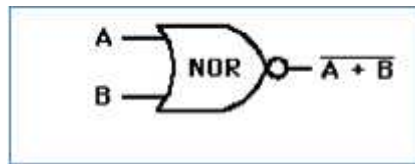
```

process (a, b)
begin
  if (a= "1"and b="1")then c<=  "0";
  else c<="1";
  end if;
end process;
end behavioral;

```

NOR gate:

2 Input NOR gate		
A	B	$\overline{A+B}$
0	0	1
0	1	0
1	0	0
1	1	0



VHDL codes:

```

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity NOR1 is
port (a : in STD_LOGIC; b : in STD_LOGIC; c : out STD_LOGIC) ;
end NOR1;
architecture behavioral of NOR1 is
begin
  process (a, b)
  begin
    if (a="0"and b= "0") then c<="0";
    else c<="0";
    end if;
  end process;
end behavioral;

```

Test Bench codes: Student will write/modify test bench codes in Xilinx ISE.

Output: Student will check the output.

Discussion: Student will conclude here.

Experiment No.-3 : Design XOR and XNOR gate

Aim: Write VHDL code for XOR and XNOR gate.

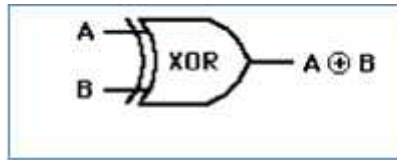
Apparatus: Xilinx ISE 14.2 software

XOR gate:

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

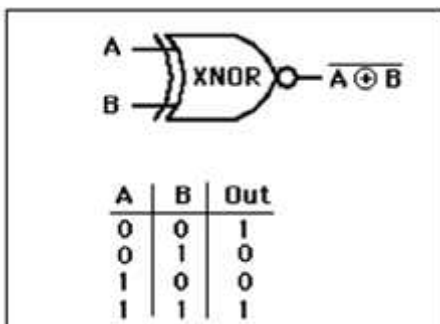
2 Input EXOR gate		
A	B	$A \oplus B$
0	0	0
0	1	1
1	0	1
1	1	0



VHDL codes:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity XOR1 is
port (a : in STD_LOGIC; b : in STD_LOGIC; c : out STD_LOGIC) ;
end XOR1;
architecture behavioral of XOR1 is
begin
process (a, b)
variable (s1, s2, s3, s4:STD_LOGIC)
begin
s1:=NOT a;
s2:=NOT b;
s3:=s1 AND b;
s4:=s2 AND a;
c<=s3 OR s4;
end process;
end behavioral;
```

XNOR gate:



VHDL codes:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity XNOR1 is
port (a : in STD_LOGIC; b : in STD_LOGIC; c : out STD_LOGIC) ;
end XNOR1;
architecture behavioral of XNOR1 is
begin
```

```

variable (s1, s2, s3, s4:STD_LOGIC)
begin
s1:=NOT a;
s2:=NOT b;
s3:=a AND b;
s4:=s1 AND s2;
c<=s3 OR s4;
end process;
end behavioral;

```

Test Bench codes: Student will write/modify test bench codes in Xilinx ISE.

Output: Student will check the output.

Discussion: Student will conclude here.

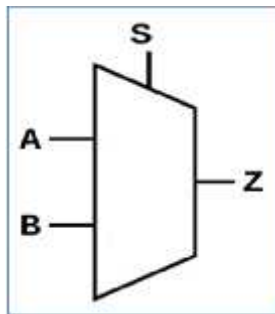
Experiment No.-4: Design 2:1 MUX

Aim: Write VHDL code for 2:1 mux using other basic gates.

Apparatus: Xilinx ISE 14.2 software

2:1 MUX:

A digital multiplexer is a combinational circuit that selects binary information from one of many input lines and directs it to a single output line.



S	Z
0	A
1	B

$$Z = A\bar{S} + BS$$

VHDL Codes:

```

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity mux_2 to 1 is
port (a : in STD_LOGIC; b : in STD_LOGIC; s : in STD_LOGIC z : out STD_LOGIC) ;
end mux_2 to 1;
architecture behavioral of mux_2 to 1 is
begin
process (a, b, s)
begin
if (s="0")then
z<=a;
else z<=b;
end if;
end process;
end behavioral;

```

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

2:1 mux using BASIC gates:

VHDL Codes:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity mux_2 to 1 is
port (a : in STD_LOGIC; b : in STD_LOGIC; s : in STD_LOGIC z : out STD_LOGIC) ;
end mux_2 to 1 ;
architecture behavioral of mux_2 to 1 is
begin
process (a, b, s)
variable (s1, s2, s3:STD_LOGIC)
begin s1:=NOT s; s2:=s1 AND a; s3:=s AND b; z<=s2 OR s3;
end process;
end behavioral;
```

Test Bench codes: Student will write/modify test bench codes in Xilinx ISE.

Output: Student will check the output.

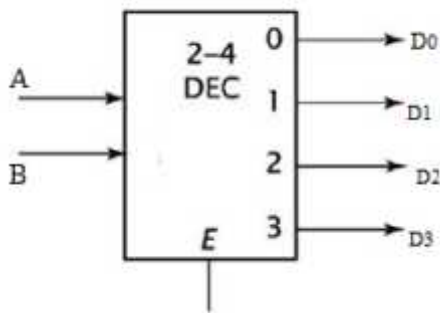
Discussion: Student will conclude here.

Experiment No.-5 :Design 2:4 Decoder

Aim: Write VHDL code for 2:4 decoder.

Apparatus: Xilinx ISE 14.2 software

2:4 decoder: A decoder is a combinational circuit that converts binary information from n inputs line to a maximum of 2^n unique output lines.



E	A	B	D0	D1	D2	D3
0	X	X	0	0	0	0
1	0	0	0	0	0	1
1	0	1	0	0	1	0
1	1	0	0	1	0	0
1	1	1	1	0	0	0

VHDL Codes:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity decoder_2_to_4 is
port (a : in STD_LOGIC_VECTOR; E : in STD_LOGIC; d : out STD_LOGIC_VECTOR (3 downto 0) ;
end decoder_2_to_4 ;
architecture behavioral of decoder_2_to_4 is
begin process (a)
begin
case a is when "00" => d<="0001";
when "01" => d<="0010";
```

```

when others=> d<="1000";
end case;
end process;
end behavioral;

```

Using DATA FLOW approach

```

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity decoder_2_to_4 is
port (a : in STD_LOGIC; b : in STD_LOGIC; E : in STD_LOGIC d : out STD_LOGIC_VECTOR (3 downto 0)) ;
end decoder_2_to_4;
architecture dataflow of decoder_2_to_4 is
signal(abar, bbar: STD_LOGIC)
begin
abar<=NOT a;
bbar<=NOT b;
d(0)<=abar AND bbar AND E;
d(1)<=abar AND b AND E;
d(2)<=a AND bbar AND E;
d(3)<=a AND b AND E;
end dataflow;

```

Test Bench codes: Student will write/modify test bench codes in Xilinx ISE.

Output: Student will check the output.

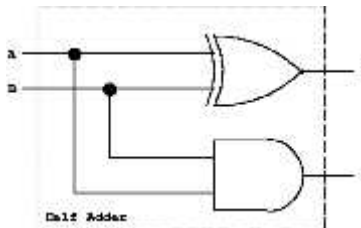
Discussion: Student will conclude here.

Experiment No.-6: Design Half adder and Full adder

Aim: Write VHDL code for Half-adder, full-adder.

Apparatus: Xilinx ISE 14.2 software

Half-adder:



A	B	Sum	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

VHDL codes:

```

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
Entity half_adder is
port (a : in STD_LOGIC; b : in STD_LOGIC; s : out STD_LOGIC; c : out STD_LOGIC);
end half_adder;

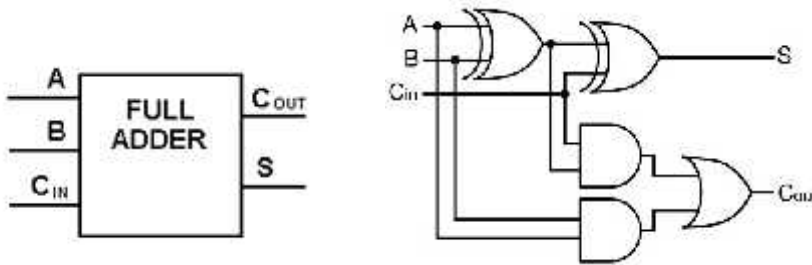
```

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

```
begin
  process (a,b)
  begin
    if (a="0" and b="0") then s<="0"; c<="0";
  elsif (a="1" and b="1") s<="0"; c<="1";
  else s<="1"; c<="0";
  end if;
  end process;
end behavioral;
```

Full-Adder:



A	B	C	Sum	Carry
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

VHDL codes:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
Entity full_adder is
  port (a : in STD_LOGIC_VECTOR (0 to 2); s : out STD_LOGIC_VECTOR (0 to 1));
end full_adder;
architecture behavioral of full_adder is
begin
  process (a)
  begin
    case a is
      when "000" => s<="00";
      when "001" => s<="10";
      when "010" => s<="10";
      when "011" => s<="01";
      when "100" => s<="10";
      when "101" => s<="01";
      when "110" => s<="01";
```


end case;
end process;
end behavioral;

Test Bench codes: Student will write/modify test bench codes in Xilinx ISE.

Output: Student will check the output.

Discussion: Student will conclude here.

Experiment No-7 : Design 3:8 Decoder

Aim: Write VHDL code for 3:8decoder.

Apparatus: Xilinx ISE 14.2 software

3:8 decoder

Inputs			outputs							
A	B	C	D7	D6	D5	D4	D3	D2	D1	D0
0	0	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	1	0	0	0	0	0	0	1	0	0
0	1	1	0	0	0	0	1	0	0	0
1	0	0	0	0	0	1	0	0	0	0
1	0	1	0	0	1	0	0	0	0	0
1	1	0	0	1	0	0	0	0	0	0
1	1	1	1	0	0	0	0	0	0	0

VHDL Codes:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL; use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity decoder_3_to_8 is
port (a : in STD_LOGIC_VECTOR (2 downto 0);
d : out STD_LOGIC_VECTOR (7 downto 0);
end decoder_3_to_8;
architecture Behavioural of decoder_3_to_8 is
begin
process(a)
begin
case a is
when "000"=> d<="00000001";
when "001"=> d<="00000010";
```

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

```

when "011"=> d<="00001000";
when "100"=> d<="00010000";
when "101"=> d<="00100000";
when "110"=> d<="01000000";
when others=> d<="10000000";
end case;
end process;
end Behavioural;

```

Test Bench codes: Student will write/modify test bench codes in Xilinx ISE.

Output: Student will check the output.

Discussion: Student will conclude here.

Experiment No.-8 : Design 8:3 priority encoder

Aim: Write VHDL code for 8:3priority encoder.

Apparatus: Xilinx ISE 8.1 software

8:3 priority encoder:

An encoder is a digital circuit that performs inverse operation of decoder. An encoder has 2^n input lines and n output lines. The output lines generate the binary code corresponding to the input value.

Truth-table for 8:3priority encoder

Inputs								outputs		
A7	A6	A5	A4	A3	A2	A1	A0	D2	D1	D0
0	0	0	0	0	0	0	0	X	X	X
0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	1	X	0	0	1
0	0	0	0	0	1	X	X	0	1	0
0	0	0	0	1	X	X	X	0	1	1
0	0	0	1	X	X	X	X	1	0	0
0	0	1	X	X	X	X	X	1	0	1
0	1	X	X	X	X	X	X	1	1	0
1	X	X	X	X	X	X	X	1	1	1

VHDL Codes:

```

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

```

```
entity p_encoder_8_to_3 is
port (a : in STD_LOGIC_VECTOR (7downto 0);
d : out STD_LOGIC_VECTOR (2downto 0));
endp_encoder_8_to_3;
```

architecture behavioral of p_encoder_8_to_3 is

```
begin
process (a)
begin
case a is
when "00000001"=> d<="000";
when "0000001X"=> d<="001";
when "000001XX"=> d<="010";
when "00001XXX"=> d<="011";
when "0001XXXX"=> d<="100";
when "001XXXXX"=>d<="101";
when "01XXXXXX"=> d<="110";
when "1XXXXXXX"=> d<= "111";
when others=> d<="XXX";
end case;
end process;
```

end behavioral;

Test Bench codes: Student will write/modify test bench codes in Xilinx ISE.

Output: Student will check the output.

Discussion: Student will conclude here.

Experiment No.-8: Design Binary to Gray converter

Aim: Design of 4 Bit Binary to Grey code Converter.

Apparatus: Xilinx ISE 14.2 software

Binary top gray converter:

The binary to grey converter is a combinational circuit that takes binary number as input and converts it into grey code. Grey code differs from the preceding and succeeding number by a single bit.

VHDL Codes:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL; use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
```

entity b2g is

```
port (b : in std_logic_vector (3 downto0);
g : out std_logic_vector (3 downto 0));
end b2g;
```

architecture behavioral of b2g is

```
begin
```

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

```
begin
case b is
when "0000" => g<= "0000";
  when "0001" => g<= "0001";
when "0010" => g<= "0011";
when "0011" => g<= "0010";
  when "0100" => g<= "0110";
when "0101" => g<= "0111";
when "0110" => g<= "0101";
  when "0111" => g<= "0100";
when "1000" => g<= "1100";
when "1001" => g<= "1101";
  when "1010" => g<= "1111";
when "1011" => g<= "1110";
  when "1100" => g<= "1010";
when "1101" => g<= "1011";
  when "1110" => g<= "1001";
when others => g<= "1000";
end case;
end process;
end behavioral;
```

Data flow model for binary to grey code converter:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL; use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity bin2grey_conv is
  port (b : in std_logic_vector (3 downto 0);
        g : out std_logic_vector (3 downto 0));
end bin2grey_conv;
architecture dataflow of bin2grey_conv is
begin
  g(3)<=b(3);
  g(2)<=(b(3)) xor (b(2));
  g(1)<=b(2) xor b(1);
  g(0)<=b(1) xor b(0);
end dataflow;
```

Test Bench codes: Student will write/modify test bench codes in Xilinx ISE.

Output: Student will check the output.

Discussion: Student will conclude here.

Experiment No.-9: Design flip-flops

Aim: Study all Flip-flops using VHDL

Apparatus: Xilinx ISE 14.2 software

(1) S-R flip-flop:

S	R	Q_{n+1}
0	0	Q _n
0	1	0
1	0	1
1	1	

VHDL Codes:

```

library IEEE;
use IEEE.STD_LOGIC_1164.ALL; use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL; entity flipflop_SR is
port (s, r, clk, rst : in std_logic; q : out std_logic);
end flipflop_SR;
architecture behavioral of flipflop_SR is
begin
process (s, r, clk, rst)
begin
if (clk= "1" and clk"event) then if (rst= "1") then
q<= "0";
elsif (rst= "0") then
q<= "1";
elsif (s= "0" and r= "0" and rst= "0") then
q<=q;
elsif (s= "0" and r= "1" and rst= "0") then
q<= "0";
elsif (s= "1" and r= "0" and rst= "0") then
q<= "1";
elsif (s= "1" and r= "1" and rst= "0") then
q<= "U";
end if;
end if;
end process;
end behavioral;

```

(2) J-K flip-flop:

J	K	Q_{n+1}

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

0	1	0
1	0	1
1	1	Not Q_n

VHDL Codes:

```

library IEEE;
use IEEE.STD_LOGIC_1164.ALL; use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL; entity flipflop_JK is
port (j, k, clk, rst : in std_logic; q : inoutstd_logic);
end flipflop_JK;
architecture behavioral of flipflop_JK is
begin
process (j, k, clk, rst)
begin
if (clk= "1" and clk"event) then if (rst= "1") then
q<= "0";
elsif (rst= "0") then
q<= "1";
elsif (j= "0" and k= "0" and rst= "0") then
q<=q;
elsif (j= "0" and k= "1" and rst= "0") then
q<= "0";
elsif (j= "1" and k= "0" and rst= "0") then
q<= "1";

elsif (j= "1" and k= "1" and rst= "0") then
q<= NOT q;
end if;
end if;

end process;
end behavioral;

```

(3) D flip-flop:



D	Q_{n+1}
0	0

VHDL Codes:

```

library IEEE;
use IEEE.STD_LOGIC_1164.ALL; use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL; entity flipflop_Dis
port (d,clk, rst : in std_logic; q : inout std_logic);
end flipflop_D;
architecture behavioral of flipflop_D is
begin
process (d,clk, rst)
begin
if (clk="1" and clk'event) then if (rst="1") then
q<= "0";
else
q<=q;
end if;
end if;
end process;
end behavioral;

```

Test Bench codes: Student will write/modify test bench codes in Xilinx ISE.

Output: Student will check the output.

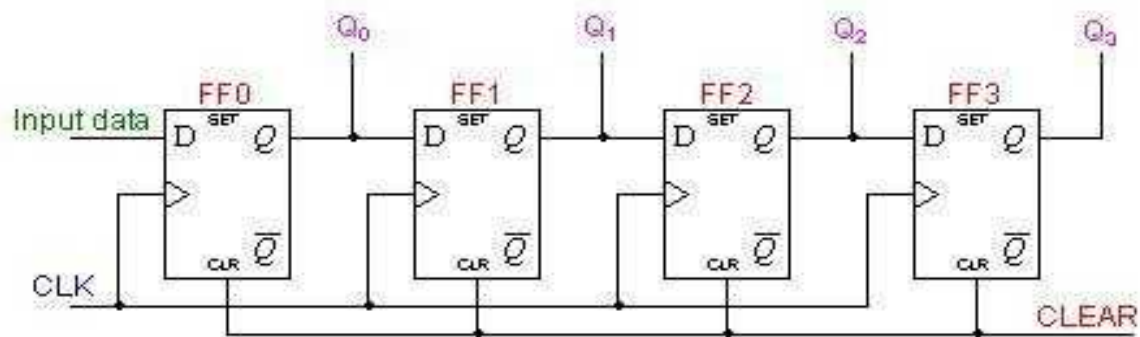
Discussion: Student will conclude here.

Experiment No.-10 : Design Shift register

Aim: Design of 8-bit shift register using VHDL.

Apparatus: Xilinx ISE 14.2 software

Shift register

**VHDL Codes:**

```

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity eftshift is
port (a : inout bit_vector (0 to 7);
r, l, rst, load, clk : in bit;
q : out bit_vector (0 to 7));
end leftshift;
architecture behavioral of leftshift is
begin
process (load, rst, a, clk)
begin
if (load="1" and rst="0" and clk'event) then
q<=a;
end if;
end process;
end leftshift;

```

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

```
elsif(load= "0") then if (rst= "1") then q<= "00000000";
else
if (l= "1") then
q<=a slll;
end if;
if (r= "1") then
q<= a srl;
end if;
end if;
end if;
end if;
end process;
end behavioral;
```

Test Bench codes: Student will write/modify test bench codes in Xilinx ISE.

Output: Student will check the output.

Discussion: Student will conclude here.

Experiment No. 11: Design ALU

Aim: Write VHDL program to perform Arithmetic Logic Unit (ALU) operation.

Apparatus: Xilinx ISE 14.2 software

ALU:

An ALU performs arithmetic and logical operation. It receives data from register file and perform operations on it given by control signals generated through control unit.

Sel	Unit	Operation
000 001 010 011	Arithmetic Unit	$z \leq x$ $z \leq x+1$ $z \leq y$ $z \leq x+y$
100 101 110 111	Logic Unit	$z \leq \text{not } x$ $z \leq x \text{ and } y$ $z \leq x \text{ or } y$ $z \leq x \text{ xor } y$

VHDL codes:

Entity ALU is

Port(x,y : in std_logic_vector(0 to 7);

sel : in std_logic_vector (0 to 2);

z : out std_logic_vector (0 to 7));

end ALU;

architecture dataflow of ALU is

signal arith, logic : std_logic_vector (0 to 7);

begin

with sel (0 to 1) select

arith <= x when "00"; x+1 when "01"; y when "10"; x+y when others;

with sel (0 to 1) select

logic <= not x when "00"; x and y when "01"; x or y when "10";

x xor y when others;

with sel (2) select

z <= arith when "0"; logic when others;

end dataflow;

Output: Student will check the output.

Discussion: Student will conclude here.

UNIVERSITY OF ENGINEERING AND MANAGEMENT, JAIPUR

Lab Manual

Title of Course: Control System Lab

Course Code: CS795C

L-T-P scheme: 0-0-3

Course Credit: 2

Objectives:

1. To provide the students with a hands-on experience on the theoretical concepts through simple experiments.
2. To develop the ability to design and validate their knowledge through open ended experiments.

Learning Outcomes:

On successful completion of this lab course, the students would be able to

1. Demonstrate and analyze the response of Transfer function for various input.
2. Analyze the response of various signal like Impulse Ramp etc.
3. Carry out the root locus of given signal.
4. Analyse different plot and state model.
5. Conduct an open ended experiment in a group of 2 to 3.

Course Contents:

List of Experiments:

1. To obtain a transfer function from given poles and zeroes using MATLAB
2. To obtain zeros and poles from a given transfer function using MATLAB
3. To obtain the step response of a transfer function of the given system using MATLAB
4. To obtain the impulse response of a transfer function of the given system using MATLAB
5. To obtain the ramp response of a transfer function of the given system using MATLAB.
6. To plot the root locus for a given transfer function of the system using MATLAB.
7. To obtain bode plot for a given transfer function of the system using MATLAB.
8. To obtain the transfer function from the state model.
9. To obtain the state model from the given transfer function.
10. To design a lag compensator for a closed loop system.

Text Book:

- 1) Katsuhiko Ogata, (2002), Modern Control Engineering, Prentice Hall of India Private Ltd., New Delhi.
- 2) Nagrath I.J. and Gopal M., (2006), Control Systems Engineering, New Age International Publisher, New Delhi.

Recommended Systems/Software Requirements:

SCILAB, MATLAB

UNIVERSITY OF ENGINEERING AND MANAGEMENT, JAIPUR

Lab Manual

1 .TRANSFER FUNCTION FROM ZEROS AND POLES

AIM: To obtain a transfer function from given poles and zeroes using MATLAB

APPARATUS:

Software: MATLAB

THEORY: A transfer function is also known as the network function is a mathematical representation, in terms of spatial or temporal frequency, of the relation between the input and output of a (linear time invariant) system. The transfer function is the ratio of the output Laplace Transform to the input Laplace Transform assuming zero initial conditions. Many important characteristics of dynamic or control systems can be determined from the transfer function. The transfer function is commonly used in the analysis of single-input single-output electronic system, for instance. It is mainly used in signal processing, communication theory, and control theory. The term is often used exclusively to refer to linear time-invariant systems (LTI). In its simplest form for continuous time input signal $x(t)$ and output $y(t)$, the transfer function is the linear mapping of the Laplace transform of the input, $X(s)$, to the output $Y(s)$. Zeros are the value(s) for z where the numerator of the transfer function equals zero. The complex frequencies that make the overall gain of the filter transfer function zero. Poles are the value(s) for z where the denominator of the transfer function equals zero. The complex frequencies that make the overall gain of the filter transfer function infinite. The general procedure to find the transfer function of a linear differential equation from input to output is to take the Laplace Transforms of both sides assuming zero conditions, and to solve for the ratio of the output Laplace over the input Laplace.

MATLAB PROGRAM:

```
z=input('enter zeroes')
p=input('enter poles')
k=input('enter gain')
[num,den]=zp2tf(z,p,k)
tf(num,den)
```

PROCEDURE:

1. Write MATLAB program in the MATLAB editor document.
2. Then save and run the program
3. Give the required input.
4. The syntax “zp2tf(z,p,k)” and “tf(num,den)” solves the given input poles and zeros and gives the transfer function.
5. zp2tf forms transfer function polynomials from the zeros, poles, and gains of a system in factored form

EXAMPLE:

Given poles are $-3.2+j7.8, -3.2-j7.8, -4.1+j5.9, -4.1-j5.9, -8$ and the zeroes are $-0.8+j0.43, -0.8-j0.43, -0.6$ with a gain of 0.5

UNIVERSITY OF ENGINEERING AND MANAGEMENT, JAIPUR

Lab Manual

THEORITICAL CALCULATIONS:

Enter zeros

Z=

Enter poles

P =

Enter gain

K=

num =

den =

Transfer function=

RESULT:

2. ZEROS AND POLES FROM TRANSFER FUNCTION

AIM:

To obtain zeros and poles from a given transfer function using MATLAB.

APPARATUS:

Software: MATLAB

THEORY:

The transfer function provides a basis for determining important system response characteristics without solving the complete differential equation. As defined, the transfer function is a rational function in the complex variable that is It is often convenient to factor the polynomials in the numerator and the denominator, and to write the transfer function in terms of those factors: where, the numerator and denominator polynomials, $N(s)$ and $D(s)$, have real coefficients defined by the system's differential equation.

MATLAB PROGRAM:

```
num = input('enter the numerator of the transfer function')
den = input('enter the denominator of the transfer function')
[z,p,k] = tf2zp(num,den)
```

PROCEDURE:

- 1.Type the program in the MATLAB editor that is in M-file.
- 2.Save and run the program.

3. Give the required inputs in the command window of MATLAB in matrix format.
3. `tf2zp` converts the transfer function filter parameters to pole-zero-gain form.
4. `[z,p,k] = tf2zp(b,a)` finds the matrix of zeros `z`, the vector of poles `p`, and the associated vector of gains `k` from the transfer function parameters `b` and `a`:
5. The numerator polynomials are represented as columns of the matrix `b`.
6. The denominator polynomial is represented in the vector `a`.
7. Note down the output of the program that is zeros, poles and gain obtained in MATLAB.
8. The zeros, poles and gain are also obtained theoretically.

THEORITICAL CALCULATIONS:

Enter the numerator of the transfer function

num =

Enter the denominator of the transfer function

den =

z =

p =

RESULT:

3. STEP RESPONSE OF A TRANSFER FUNCTION

AIM: To obtain the step response of a transfer function of the given system using MATLAB

APPARATUS: Software: MATLAB

THEORY: A step signal is a signal whose value changes from one level to another level in zero time.

MATLAB PROGRAM:

```
num = input('enter the numerator of the transfer function')
den = input('enter the denominator of the transfer function')
step(num,den)
```

PROCEDURE:

- ❑ Type the program in MATLAB editor that is in M-file.
- ❑ Save and run the program.
- ❑ Give the required inputs in the command window of MATLAB in matrix format.
- ❑ 'step' function calculates the unit step response of a linear system.
- ❑ Zero initial state is assumed in state-space case.
- ❑ When invoked with no output arguments, this function plots the step response on the screen.

- ❑ step (sys) plots the response of an arbitrary LTI system.
- ❑ This model can be continuous or discrete, and SISO or MIMO.
- ❑ The step response of multi-input systems is the collection of step responses for each input channel.
- ❑ The duration of simulation is determined automatically based on the system poles and zeroes.
- ❑ Note down the response of the transfer function obtained in MATLAB.
- ❑ The response of the transfer function is also obtained theoretically.
- ❑ Both the responses are compared.

THEORETICAL CALCULATIONS: Calculation will be in the form of graph.

RESULT:

4. IMPULSE RESPONSE OF A TRANSFER FUNCTION

AIM: To obtain the impulse response of a transfer function of the given system using MATLAB.

APPARATUS:

Software: MATLAB

THEORY:

An impulse signal is a signal whose value changes from zero to infinity in zero time.

MATLAB PROGRAM:

```
num = input('enter the numerator of the transfer function')  
den = input('enter the denominator of the transfer function')  
impulse(num,den)
```

PROCEDURE:

- ❑ Type the program in the MATLAB editor that is in M-file.
- ❑ Save and run the program.
- ❑ Give the required inputs in the command window of MATLAB in matrix format.
- ❑ 'impulse' calculates the impulse response of a linear system.
- ❑ The impulse response is the response to the Dirac input, $\delta(t)$ for continuous time systems and to a unit pulse at for discrete time systems.
- ❑ Zero initial state is assumed in the state space case.
- ❑ When invoked without left hand arguments, this function plots the impulse response on the screen.
- ❑ 'impulse(sys)' plots the impulse response of an arbitrary LTI model sys.
- ❑ This model can be continuous or discrete, SISO or MIMO.
- ❑ The impulse response of multi-input systems is the collection of impulse responses for each input channel.
- ❑ The duration of simulation is determined automatically to display the transient behavior of the response.

UNIVERSITY OF ENGINEERING AND MANAGEMENT, JAIPUR

Lab Manual

- ☐ Note down the response of the given transfer function obtained in MATLAB.
- ☐ The response of the transfer function is also obtained theoretically.
- ☐ Both the responses are compared.

GRAPH-

RESULT:

5. RAMP RESPONSE OF A TRANSFER FUNCTION

AIM:

To obtain the ramp response of a transfer function of the given system using MATLAB.

APPARATUS:

Software: MATLAB

THEORY:

A ramp signal is a signal which changes with time gradually in a linear fashion

MATLAB PROGRAM:

```
num = input('enter the numerator of the transfer function')  
den = input('enter the denominator of the transfer function')  
lsim(num,den,u,t)
```

PROCEDURE:

- ☐ Type the program in the MATLAB editor that is in M-file.
- ☐ Save and run the program.
- ☐ Give the required inputs in the command window of MATLAB in matrix format.
- ☐ lsim simulates the (time) response of continuous or discrete linear systems to arbitrary inputs.
- ☐ When invoked without left-hand arguments, lsim plots the response on the screen.
- ☐ lsim(sys,u,t) produces a plot of the time response of the LTI model sys to the input time historyt,u.
- ☐ The vector t specifies the time samples for the simulation and consists of regularly spaced time samples.
- ☐ $t = 0:dt:T_{final}$
- ☐ The matrix u must have as many rows as time samples (length(t)) and as many columns as system inputs.
- ☐ Each row u(i,:) specifies the input value(s) at the time sample t(i).
- ☐ Note down the response of the transfer function obtained in MATLAB.
- ☐ The response of the transfer function is also obtained theoretically.
- ☐ Both the responses are compared.

GRAPH:

UNIVERSITY OF ENGINEERING AND MANAGEMENT, JAIPUR

Lab Manual

RESULT:

6.ROOT LOCUS FROM A TRANSFER FUNCTION

AIM:To plot the root locus for a given transfer function of the system using MATLAB.

APPARATUS:

Software: MATLAB

THEORY:

rlocus computes the Evans root locus of a SISO open-loop model. The root locus gives the closed-loop pole trajectories as a function of the feedback gain k (assuming negative feedback).

Root loci are used to study the effects of varying feedback gains on closed-loop pole locations.

In turn, these locations provide indirect information on the time and frequency responses.

rlocus(sys) calculates and plots the rootlocus of the open-loop SISO model sys. This function can be applied to any of the following feedback loops by setting sys appropriately.

MATLAB PROGRAM:

```
num=input('enter the numerator of the transfer function')
den=input('enter the denominator of the transfer function')
h=tf(num,den)
rlocus(h)
```

PROCEDURE:

❑ Write MATLAB program in the MATLAB specified documents.

❑ Then save the program to run it.

27

❑ The input is to be mentioned.

❑ The syntax " $h=tf(num,den)$ " gives the transfer function and is represented as h .

❑ The syntax " $rlocus(h)$ " plots the rootlocus of the transfer function h .

❑ Generally the syntax is of the form

$rlocus(sys)$

$rlocus(sys,k)$

$rlocus(sys1, sys2,)$

$[r,k] = rlocus(sys)$

$r = rlocus(sys,k)$

❑ rlocus(sys) calculates and plots the root locus of the open loop SISO model sys.

❑ Now we have to solve it theoretically.

❑ Now we have to compare the practical and theoretical outputs to verify each other correctly.

THEORETICAL CALCULATIONS:

enter the numerator of the transfer function

num=

enter the denominator of the transfer function

den=

Transfer function :

RESULT:

7.BODE PLOT FROM A TRANSFER FUNCTION

AIM: To obtain bode plot for a given transfer function of the system using MATLAB.

APPARATUS:

Software: MATLAB

THEORY:

Bode computes the magnitude and phase of the frequency response of LTI models. When invoked without left-side arguments, bode produces a Bode plot on the screen. The magnitude is plotted in decibels (dB), and the phase in degrees. The decibel calculation for mag is computed as $20\log_{10}(|H(j\omega)|)$, where $H(j\omega)$ is the system's frequency response. Bode plots are used to analyze system properties such as the gain margin, phase margin, DC gain, bandwidth, disturbance rejection, and stability

bode(sys) plots the Bode response of an arbitrary LTI model sys. This model can be continuous or discrete, and SISO or MIMO. In the MIMO case, bode produces an array of Bode plots, each plot showing the Bode response of one particular I/O channel. The frequency range is determined automatically based on the system poles and zeros.

bode(sys,w) explicitly specifies the frequency range or frequency points to be used for the plot.

To focus on a particular frequency interval [wmin,wmax], set $w = \{wmin, wmax\}$. To use particular frequency points, set w to the vector of desired frequencies. Use logspace to generate logarithmically spaced frequency vectors. All frequencies should be specified in radians/sec.

bode(sys1,sys2,...,sysN) or bode(sys1,sys2,...,sysN,w) plots the Bode responses of several LTI models on a single figure. All systems must have the same number of inputs and outputs, but may otherwise be a mix of continuous and discrete systems. This syntax is useful to compare the Bode responses of multiple systems.

bode(sys1,'PlotStyle1',...,sysN,'PlotStyleN') specifies which color, linestyle, and/or marker should be used to plot each system. For example,

bode(sys1,'r--',sys2,'gx') uses red dashed lines for the first system sys1 and green 'x' markers for the second system sys2.

When invoked with left-side arguments

[mag,phase,w] = bode(sys)

[mag,phase] = bode(sys,w)

return the magnitude and phase (in degrees) of the frequency response at the frequencies w (in rad/sec). The outputs mag and phase are 3-D arrays with the frequency as the last dimension (see "Arguments" below for details). You can convert the magnitude to decibels by $\text{magdb} = 20*\log_{10}(\text{mag})$

UNIVERSITY OF ENGINEERING AND MANAGEMENT, JAIPUR

Lab Manual

MATLAB PROGRAM:

```
num=input('enter the numerator of the transfer function')
den=input('enter the denominator of the transfer function')
h=tf(num,den)
[gm pm wcpwgc]=margin(h)
bode(h)
```

PROCEDURE:

- ❑ Write the MATLAB program in the MATLAB editor.
- ❑ Then save and run the program.
- ❑ Give the required inputs.
- ❑ The syntax "bode(h)" solves the given input transfer function and gives the bode plot,
- ❑ wherenum,den are the numerator and denominator of the transfer function.
- ❑ Now plot the bode plot theoretically for the given transfer function and compare it with theplot obtained practically.

THEORETICAL CALCULATIONS:

enter the numerator of the transfer function
num =
enter the denominator of the transfer function
den =
Transfer function:
gm =
pm =
wcp =
wcg =

RESULT:

8.TRANSFER FUNCTION FROM STATE MODEL

AIM: To obtain the transfer function from the state model.

APPARATUS:

Software: MATLAB

THEORY:

The transfer function is defined as the ratio of Laplace transform of output to Laplace transform of input. A state space representation is a mathematical model of a physical system as a set of input, output and state variables related by first-order differential equations. The state space representation (also known as the "time-domain approach") provides a convenient and compact way to model and analyze systems with multiple inputs and outputs.

Unlike the frequency domain approach, the use of the state space representation is not limited to systems with linear components and zero initial conditions.

"State space" refers to the space whose axes are the state variables. The state of the system can be represented as a vector within that space.

MATLAB PROGRAM:

```
A=input('enter the matrix A')  
B=input('enter the matrix B')  
C=input('enter the matrix C')  
D=input('enter the matrix D')  
Sys=ss2tf(A,B,C,D)
```

EXAMPLE:

Obtain the transfer function from the State Model given below:

A=
B=
C=
D=

PROCEDURE:

- ❑ Type the program in the MATLAB editor that is in M-file.
- ❑ Save and run the program.
- ❑ Give the required inputs in the command window of MATLAB in matrix format.
- ❑ The command ss2tf(A,B,C,D)) converts the given transfer function into a state model.
- ❑ Note down the output obtained in MATLAB.
- ❑ The Transfer Function is also obtained theoretically.
- ❑ Both the state models are compared.

RESULT:

9.STATE MODEL FROM TRANSFER FUNCTION

AIM:

To obtain the state model from the given transfer function.

APPARATUS:

Software: MATLAB

THEORY:

There are three methods for obtaining state model from transfer function:

1. Phase variable method
2. Physical variable method
3. Canonical variable method

Out of three methods given above canonical form is probably the most straightforward method

for converting from the transfer function of a system to a state space model is to generate a model in "controllable canonical form." This term comes from Control Theory but its exact meaning is not important to us. To see how this method of generating a state space model works, consider the third order differential transfer function

MATLAB PROGRAM:

```
num=input('enter the numerator of the transfer function')  
den=input('enter the denominator of the transfer function')
```

UNIVERSITY OF ENGINEERING AND MANAGEMENT, JAIPUR

Lab Manual

`ss(tf(num,den))`

PROCEDURE:

- ☐ Type the program in the MATLAB editor that is in M-file.
- ☐ Save and run the program.
- ☐ Give the required inputs in the command window of MATLAB in matrix format.
- ☐ The command `ss(tf(num,den))` converts the given transfer function into a state model.
- ☐ Note down the output obtained in MATLAB.
- ☐ The state model is also obtained theoretically.
- ☐ Both the state models are compared.

RESULT:

10.STATE MODEL FROM ZEROS AND POLES

AIM: To obtain a state model from given poles and zeros using MATLAB.

APPARATUS:

Software: MATLAB

THEORY:

Let's say we have a transfer function defined as a ratio of two polynomials:

$H(s) =$

Where $N(s)$ and $D(s)$ are simple polynomials.

Zeros are the roots of $N(s)$ (the numerator of the transfer function) obtained by setting $N(s)=0$

and solving for s . Poles are the roots of $D(s)$ (the denominator of the transfer function), obtained by setting $D(s)=0$ and solving for s .

The state space model represents a physical system as n first order coupled differential equations.

This form is better suited for computer simulation than an n th

order input-output differential equation.

The general vector-matrix form of state space model is:

Where,

X = state vector

U = input vector

A = $n \times n$ matrix

B = $n \times 1$ matrix

The output equation for the above system is,

42

MATLAB PROGRAM:

`z=input('enter zeros')`

`p=input('enter poles')`

`k=input('enter gain')`

UNIVERSITY OF ENGINEERING AND MANAGEMENT, JAIPUR

Lab Manual

$[A,B,C,D]=zp2ss(z,p,k)$

PROCEDURE:

- ❑ Open the MATLAB window and open a new MATLAB editor.
- ❑ Write the MATLAB program in the MATLAB editor.
- ❑ Save and run the MATLAB program.
- ❑ Enter the given poles, zeros and gain as input in matrix format.
- ❑ The syntax “[A,B,C,D]=zp2ss(z,p,k)” solves zeroes, poles and gain given in the matrix format as input and gives the output in the form of a state model.
- ❑ This syntax transforms the given zeros, poles and gain into a state model.
- ❑ Note down the output state model obtained practically by using the syntax “[A,B,C,D]=zp2ss(z,p,k)” .
- ❑ Now find the state model theoretically for the given poles, zeros and gain.
- ❑ Compare the theoretically obtained state model from the given poles, zeros and gain with the one obtained practically. Write the result based on the comparison between theoretical and practical result.

EXAMPLE:

zeros are:

poles are:

gain=

RESULT:

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

Title of Course: Modeling & Simulation Lab

Course Code: CS795D

L-T-P scheme: 0-0-3

Course Credit: 2

Introduction:

Modelling and simulation are a vital part of many areas of engineering, allowing engineers to reason about the expected behaviour of a system without having to physically implement it. Simulation pervades much of electrical engineering, for example models of individual electronic devices, circuit simulation, network modeling, compression of speech/audio/image/video signals, design of biomedical devices, and modeling of physical systems for control purposes. Modelling allows an abstract representation of a signal or system in a (mathematically) compact and/or simplified form that is extremely useful in many fields, including analysis, design, compression, classification, and control. The main high-level aim of the course is to provide a thorough grounding in aspects of constructing and applying models and their simulation using well-known simulation tools (MATLAB and C). In particular, the course looks at how continuous-time systems can be represented and simulated using (discrete-time) computers. This also provides an interesting insight into the relationship between physical systems and computing algorithms. The course is intentionally designed to have a strong practical focus, with extensive laboratory work serving to develop key skills in computing and applications of mathematics.

Objectives:

This course aims to:

- a. Familiarise you with programming in MATLAB.
- b. Convey the analytical and practical details of a range of modelling techniques.
- c. Provide an understanding of finite difference approximation and numerical methods for differential equations, in the context of state-space representations of linear systems.
- d. Familiarise you with the modeling of dynamical systems and stochastic signals, including the choice of model, choice of model order, parameter estimation and goodness of fit.
- e. Provide a thorough grounding in parameter estimation techniques such as least squares (particularly) and maximum likelihood.
- f. Give you practical experience with simulating physical systems and modeling typical experimental data, for example second-order circuits, non-linear circuits, electrical machines and power systems, control systems, biomedical systems, and introductory network simulation2 .

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

Learning Outcomes:

On successful completion you should be able to:

1. Express a linear system in terms of its differential equation, transfer function, magnitude response, impulse response and step response, be able to convert between the different forms and explain the advantages of each;
2. Derive expressions that can be used to estimate parameters from different types of data, for different types of model structures;
3. Explain analytically how to simulate a continuous-time system by means of numerical integration;
4. Synthesise MATLAB code to simulate a given system or model;
5. Implement a suitable model for a given problem, making informed choices about the model type and model order, and calculate the model error.
6. Deduce the behaviour of previously unseen models or parameterisations and hypothesise about their merits

Course Contents:

Exercises that must be done in this course are listed below:

Experiment No.1: Introductory MATLAB

Experiment No.2: Circuit simulation

Experiment No.3 : Linear system

Experiment No.4: Numerical Des

Experiment No.5: Runge -Kutta

Experiment No.6: Least squares

Experiment No.7: System identification

Experiment No.8: Stochastic models

Experiment No.9: Parameter Estimation

Text Books:

1. Klee, H. (2007). Simulation of Dynamic Systems with MATLAB and Simulink, CRC Press, Boca Raton, FL. – This is a very detailed and comprehensive text, aimed slightly above the level of this course. For anyone with longer-term interests in dynamic systems, this text is highly recommended.
2. Woods, R. L., and Lawrence, K. L. (1997), Modeling and simulation of dynamic systems, Prentice-Hall, Upper Saddle River, NJ. – This is a more introductory level text that also deals with dynamic systems, across all areas of engineering. The coverage of the course is not very complete, but the style is fairly straightforward and there are many problems (with answers) given.

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Course Description

Title of Course: Seminar on Industrial Training

Course Code: CS781

L-T –P Scheme: 0-0-3

Course Credits: 2

Course Description & Objectives:

1. **Understand** the history of medical research and bioethics related to the HeLa cells. Understand the diverse social and economic, racial and gender contexts within which Henrietta Lacks lived and died. Understand the themes of this seminar. Appreciate the legacy and implications of these medical, ethical and social understandings on today's society.
2. **Identify**, understand and discuss current, real-world issues.
3. **Distinguish** and **integrate** differing forms of knowledge and academic disciplinary approaches (e.g., humanities and sciences) with that of the student's own academic discipline (e.g., in agriculture, architecture, art, business, economics, education, engineering, natural resources, etc.). And apply a **multidisciplinary strategy** to address current, real-world **issues**.
4. Improve oral and written **communication** skills.
5. Explore an appreciation of the **self** in relation to its larger diverse social and academic contexts.
6. Apply principles of **ethics** and **respect** in interaction with others.

Course Outcomes:

After the completion of this course, the student should be able to:

1. **Learn and integrate.** *Through independent learning and collaborative study, attain, use, and develop knowledge in the arts, humanities, sciences, and social sciences, with disciplinary specialization and the ability to integrate information across disciplines.*
2. *Use multiple thinking strategies to examine real-world issues, explore creative avenues of expression, solve problems, and make consequential decisions*
3. **Learn and integrate.** *Communicate. Acquire, articulate, create and convey intended meaning using verbal and non-verbal method of communication that demonstrates respect and understanding in a complex society.*
4. *Use multiple thinking strategies to examine real-world issues, explore creative avenues of expression, solve problems, and make consequential decisions.*

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Course Description

5. **Clarify purpose and perspective.** *Explore one's life purpose and meaning through transformational experiences that foster an understanding of self, relationships, and diverse global perspectives.*
6. **Practice citizenship.** *Apply principles of ethical leadership, collaborative engagement, socially responsible behavior, respect for diversity in an interdependent world, and a service-oriented commitment to advance and sustain local and global communities.*

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Course Description

Title of Course: Group Discussion

Course Code: HU781

L-T –P Scheme: 0-0-3

Course Credits: 2

A group discussion aims at a structured but informal exchange of knowledge, ideas, and perceptions among the participants on any issue, topic or sub-topic. Contributions are pooled together and examined in terms of their relevance and validity to the discussion objectives. If planned and organized in a structured way and certain essential conditions are met, it can provide a highly enriching and stimulating experience to the participants. Lets us see, the objectives, different steps involved in it and its limitations.

Objectives of a Group Discussion

-) Produce a range of options or solutions, addressing a particular problem or an issue.
-) Generate a pile of ideas by examining issues in greater depth, looking at different dimensions of these issues.
-) Broaden the outlook of the participants through cross-fertilization and exposure to new and different experiences and ideas and enrich their understanding of the issues under discussion.
-) Develop their skills in interpersonal communication and in expressing their views in a clear and succinct manner.
-) Effective means of changing attitudes through the influence of peers in the group
-) Valuable means of obtaining feedback for the training team on verbal skills, motivation level and personal traits of the participants and characteristics of the group

Steps in organizing a Group Discussion

-) Setting up the Groups
-) Planning a Group Discussion
-) Preparation of Group Reports
-) Presentation and Consolidation of Group Reports

Limitations

-) If the group is large, not all the members may get the opportunity to participate and contribute to the discussion.
-) If the task is not clearly defined, the discussion may lack focus and, as a result, it may be unproductive.
-) Difficulties can arise if the leader is unskilled in guiding the discussion and/or not familiar with the topic or the issues.
-) Some members may dominate and, in a way, hijack the discussion.

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Course Description

-) As this is a group task, some members may take it easy and not feel constrained to participate.

Learning outcomes

After studying this course, you should be able to:

-) understand the key skills and behaviours required to facilitate a group discussion
-) prepare effectively before facilitating a meeting
-) consider some of the difficult behaviours that can occur in meetings
-) think of some possible strategies for dealing with these.

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

Title of Course: Project Part- I

Course Code: CS782

L-T –P Scheme: 6P

Course Credits: 4

Project: an activity where the participants have some degree of *choice* in the outcome. The result is complete and functional, that is, it has a beginning, middle and end. Usually, it spans multiple lab periods and requires work outside scheduled lab periods. Since there are choices in implementation, *design* is inherently a component of a project. A project is inherently different from an *analysis* or *exercise*, in which the solution has a predictable form. Projects span a wide variety of possibilities: design and build, identify a system, do a forensic analysis, evaluate a product or assess some environmental situation.

Program Objective 1

Graduates shall make their way to the society with proper scientific and technical knowledge in mechanical engineering.

Program Objective 2

Graduates shall work in design and analysis of mechanical systems with strong fundamentals and methods of synthesis.

Program Objective 3

Graduates shall adapt to the rapidly changing environment in the areas of mechanical engineering and scale new heights in their profession through lifelong learning.

Program Objective 4

Graduates shall excel in career by their ability to work and communicate effectively as a team member and/or leader to complete the task with minimal resources, meeting deadlines.

Program Outcomes:

1. Ability to apply knowledge of mathematics, science and mechanical engineering fundamentals for solving problems.
2. Ability to Identify, formulate and analyze mechanical engineering problems arriving at meaningful conclusions involving mathematical inferences.
3. Ability to design and develop mechanical components and processes to meet desired needs considering public health, safety, cultural, social, and environmental aspects.
4. Ability to understand and investigate complex mechanical engineering problems experimentally.
5. Ability to apply modern engineering tools, techniques and resources to solve complex mechanical engineering activities with an understanding of the limitations.
6. Ability to understand the effect of mechanical engineering solutions on legal, cultural, social, public health and safety aspects./li>

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

7. Ability to develop sustainable solutions and understand their impact on society and environment.
8. Ability to apply ethical principles to engineering practices and professional responsibilities.
9. Ability to function effectively as an individual and as a member or leader in diverse teams and in multidisciplinary settings.
10. Ability to comprehend, design documentation, write effective reports, make effective presentations to the engineering community and society at large.
11. Ability to apply knowledge of engineering and management principles to lead teams and manage projects in multidisciplinary environments.
12. Ability to engage in independent and life-long learning in the broad context of technological changes and advancements.