CE/CZ 2001 – Algorithms

Course Code	CE/CZ 2001							
Course Title	Algorithms							
Pre-requisites	CE/CZ 1007: Data Structures							
No of AUs	3							
Contact Hours	Lectures	25	TEL	2	Tutorials	11	Example classes	8

Course Aims

This core engineering course aims to develop your knowledge, understanding and skills about algorithms, including: (1) methods and techniques to design and implement algorithms; (2) methods and techniques to analyse the correctness and resource requirements (mainly the time and space complexities) of algorithms. Because algorithms are essential to both Computer Engineering (CE) and Computer Science (CS), this course has vital importance for learning other courses in CE or CS, and shall prepare you for future careers in the science and technology of computing.

Intended Learning Outcomes (ILO)

By the end of this course, you should be able to:

1. Conduct complexity analysis of basic algorithms.
2. Design and analyse algorithms using the suitable strategies (e.g. incremental, divide and conquer, data-structure, and greedy approaches) to solve a problem.
3. Compare the efficiencies of different algorithms for the same problem (e.g. searching, sorting or graph traversal).
4. Describe various heuristic problem-solving methods.
5. Implement algorithms from pseudo code into real code.

Course Content

	Topics	Lectures (Hours)	Tutorials (Hours)	Example Classes (Hours)
1	**Introduction to algorithms** What is an algorithm? Algorithm design and analysis. Examples of different design: Power function, Sine Function, Maclaurin Series vs CODIC. Mathematical Foundation: Set, Functions, Floor and Ceiling, Power and Logarithm, Summations and Series, Limits, Proof Method.	**2**	**1**	**0**
2	**Analysis of Algorithms** Time and space complexities of algorithms. Analyzing basic program constructs. Best case, worst case and average case time complexity analysis. Deducing recurrence relations for time complexity of recursive algorithms. Solving elementary recurrence relations. Asymptotic time complexity analysis. Big-Oh, big-Omega, and big-Theta notations. Common Complexity Classes. Basic techniques for proving asymptotic bounds. Space Complexity. Faster computer or faster algorithms.	**5**	**2**	**2**
3	**Searching** Iterative and recursive sequential search algorithms. Binary search, its invariance, and complexity. Open and closed address hashing using linear probing and double hashing collision resolution techniques. All algorithms covered are accompanied by asymptotic complexity analysis.	**2**	**1**	**2**
4	**Sorting** Insertion-sort, Heap-sort, Quick-sort, Merge-sort, which contrast three approaches: Incremental, Data Structures and Divide-and-Conquer. All algorithms covered are accompanied by asymptotic complexity analysis.	**6**	**3**	**2**

5	**Graphs** Basic graph representation methods, adjacency lists, and adjacency matrix, Systematic graph traversals with breadth-first search (BFS) and depth-first search (DFS) algorithms. A generic backtracking algorithm and its complexity. Maze-Search. Introduction to Greedy algorithms; Dijkstra's Single-source Shortest Paths algorithm, Correctness proof and complexity of Dijkstra's algorithm. Minimum Spanning trees and their properties. Greedy algorithm for finding Minimum Spanning Trees (MSTs) using Prim's algorithm.	**6**	**3**	**2**
6	**Basic Computability and Complexity Theory** Tractable and intractable problems. Un-computable functions. The Halting Problem. Non-determinism as a computational model. The Classes NP and P. Brute-force and heuristic algorithms Nearest-link-first and shortest-link-first for Travelling Sales Person (TSP) problem.	**2**	**1**	**0**
7	Revision	**2**	**0**	**0**
	Check for Hours	**=25**	**=11**	**=8**

Assessment (includes both continuous and summative assessment)

a) Final Examination: 60%
b) Mid-Term Quiz: 20%
c) Example Classes: 20%

CE/CZ 2002 – Object Oriented Design & Programming

Course Code	CE/CZ 2002							
Course Title	Object Oriented Design & Programming							
Pre-requisites	CE/CZ 1007: Data Structure							
No of AUs	3							
Contact Hours	Lectures	24	TEL	0	Tutorials	10	Laboratories	10

Course Aims

The object-oriented paradigm to the design of software is one major successful approach to address complexity and maintainability issues in software systems. We want students to establish an object-oriented mindset and to gain valuable insights into how software can be developed using the object-oriented approach. This course should not be interpreted to be a pure programming language course. Rather, the programming language serves to illustrate, via practical examples, the concepts learnt in the course.

As a student of this course, you will learn essential object-oriented concepts such as encapsulation, the separation of design from implementation; the use of inheritance and polymorphism. You will discover how to describe these concepts using appropriate UML diagrams. Finally, you will also learn good design principles for reuse, and to realise these principles using object-oriented programming languages such as Java and/or C++.

Intended Learning Outcomes (ILO)

Upon completion of the course, you should be able to:

1. Explain the concepts of object-oriented methodology and demonstrate it in developing software programs.
2. Design simple software programmes using good design principles with consideration for reuse and maintainability, to solve problems.
3. Implement a given design in Java and/or C++.

Course Content

	Topics	Lectures (Hours)	Tutorials (Hours)	Presentations (Hours)
1	**Introduction to Object Orientated Programming** Procedural vs OO programming; Object and Class; Four basic elements and four features of OO programming.	**2**	**0**	
2	**Class and Object** Attributes; Class Definition; Message Sending; Copying Objects; The Keyword 'this'; Accessors and Mutators; The Keyword 'static'; Static vs. Instance methods; Object Composition.	**4**	**2**	
3	**Inheritance in Java** Generalization and Specialization; Method Overloading & Overriding; Liskov Substitution Principle; Visibility Modifiers; Final Classes and Methods; Abstract Classes and Methods; Multiple Inheritance and Interfaces.	**3**	**2**	
4	**Exception Handling & Persistent Objects (E-Learning)** Error Handling; Java's Exception Handling; Java's Exception Hierarchy; Exception Classes; Object serialization; Saving objects, e.g. to file, to RDBMS.	**2**	**1**	
5	**Polymorphism in Java** Polymorphism; Binding; Object Typecasting; Benefits of Polymorphism; Three ways of Method Overriding.	**3**	**2**	
6	**Object Relationships** Aggregation. Composition. Associations. Delegation. Cardinality.	**2**	**1**	
7	**Object Collaboration** Messages. Object Interactions. Sequence Flow. UML Sequence Diagram.	**2**	**1**	
8	Modelling OO Application Identifying Class and Objects, Defining Classing, Use of Class and Sequence	**2**	**1**	
9	**Designing for Reuse** Good design principles e.g. Single Responsibility Principle (SRP). Don't Repeat Yourself (DRY) Principle, Open-Closed Principle (OCP). Interfaces and abstract classes. Design by contract. Inheritance versus Delegation. Loose coupling. Design pattern e.g. Singleton, Façade.	**2**	**1**	

10	**C++ Programming Language** Types and declarations. Pointers and arrays. Expressions and statements. Functions/Methods. Standard libraries. Transforming class specification into code.	**4**	**2**	
	Check for Hours	**=26**	**=13**	

Assessment (includes both continuous and summative assessment)

a) Final Examination: 60%

b) Clicker/Java Quiz (Participation): 5%

c) Laboratory Assessment: 5%

d) Assignment: 30%

CE2003 - Digital Systems Design

Course Code	CE 2003									
Course Title	Digital Systems Design									
Pre-requisites	CE/CZ 1005: Digital Logic									
No of AUs	3									
Contact Hours	Lecture	4	TEL	22	Tutorials	13	Laboratories	10	Example Class	5

Course Aims

This course aims to develop skills in digital hardware design to meet basic industry expectations. Here, the principles established in CE/CZ1005-Digital Logic will be extended to encompass circuits of greater complexity. These circuits are no longer implemented using multiple discrete logic elements, as presented in CE/CZ1005, but rather use Application Specific Gate Array (ASIC) or Field Programmable Gate Array (FPGA) technologies. Both of these technologies can accommodate circuits of complexity equivalent to many millions of gates.

This course will focus on FPGA technology as it is gaining commercial popularity (due to the cost of ASIC Implementations). The necessary logic and connectivity for both technologies is synthesised by software tools from code written in a Hardware Description Language. As such, this course presents you with circuit design techniques for more advanced systems in a methodical manner based on modern design techniques.

Intended Learning Outcomes (ILO)

To gain the theoretical and practical knowledge and experience needed for the design and implementation of real digital systems on modern programmable devices.

By the end of this course, you (as a student) would be expected to be able to:

1. Use electronic design automation (EDA) tools in digital circuit modelling, design, simulation, verification and prototyping with FPGA, including:
    a. Designing digital systems with moderate datapath and control complexity,
    b. Integrating designs with external IP blocks,
    c. Using a ‘testbench’ to verify a design’s functionality
2. Analyse how the use of FPGA resources can impact the performance of a hardware implementation
3. Effectively present designs and analyses in written and oral form

Course Content

	Topics	Lectures (Hours)	TEL (Hours)	Tutorials (Hours)
1	**Review of Verilog and the Digital Design Flow** Verilog code structure, modules, ports, continuous assignments, behavioural modelling for combinational and synchronous circuits. Basic outline of the digital design flow including synthesis, mapping, placement and routing. Differences between ASIC and FPGA flows.		2	1
2	**Verification and Testing** Importance of testing, causes of design errors. Verilog testbench design, self-checking testbenches, using simulators, generating signals. Testing strategies, hierarchical testing. Testing methodology.	2		1
3	**Arithmetic Design** Number representations, signed numbers, basic arithmetic circuits, ripple and carry-lookahead adders, binary multiplication. Numbers in Verilog, signed arithmetic in Verilog. Converting to fixed-point and analysing conversion error. Arithmetic on FPGAs.		3	2
4	**FPGA Architecture and Synthesis** Understanding basic building blocks in FPGAs: configurable logic blocks and look-up tables, flexible routing, I/O blocks, embedded memories, digital signal processing blocks. Understanding the basic tasks that are automated by synthesis tools.	2		1
5	**Timing, Pipelining, and Scheduling** Combinational delay. Critical path in synchronous circuits, clocking. Pipelining simple circuits. Drawing and analysing schedules. Designing datapaths from schedules. Resource-constrained scheduling. Fundamentals of pipelined datapath design.		4	2
6	**Subsystem Design** Datapath controllers, simple CPU components: register file, memories, ALU, program counter, control, simple 5 stage pipeline. Microprocessor case study.		5	2
7	**Busses and Interfacing** Serial and parallel busses. AMBA AXI basic, burst transfers, wait states. SERDES Multi-Gigabit Transceivers. PCI-e. Bus scalability. Introduction to network-on-chip.		2	1
8	**Asynchronous Circuits** Advantages/disadvantages. Flow tables, flow diagrams, state reduction, implementation.		4	2
9	**Revision**	1		
	Check for Hours	= 6	= 20	= 12

Assessment (includes both continuous and summative assessment)

a) Final Examination: 60%
b) Quiz: 15%
c) Laboratory: 15%
d) Project: 10%

Note: Final examination and quiz are closed book.

CE 2004 – Circuits and Systems

Course Code	CE 2004							
Course Title	Circuits and Signal Analysis							
Pre-requisites	NIL							
No of AUs	3							
Contact Hours	Lectures	26	TEL	0	Tutorials	12	Laboratories	8

Course Aims

The first part of this course aims to develop the students' ability to identify and solve circuit analysis problems associated with simple resistor circuits and RL & RC circuits. With the various circuit analysis techniques covered in this subject, the students will be able to identify and choose the most effective methods to solve the given circuit design. The second part of this course aims to enable the students to analyse and design circuits with active circuit elements (Op-Amps), and perform analysis for signals and systems in general, with emphasis on Linear time-invariant (LTI) systems.

Intended Learning Outcomes (ILO)

This course introduces Circuits Analysis and Signal Analysis. Upon the successful completion of this course, you shall be able to:

1. Explain the need for Circuit Analysis for simple resistor circuits and RL & RC circuits;
2. Identify and choose the most effective Circuit Analysis Methods for a given Circuit Design;
3. Solve both resistive circuits (DC) as well as RL & RC circuits (AC – Step Input);
4. Analyse and design active circuits (with Op-Amps);
5. Use a more general methodology to tackle different signals and LTI systems.

Course Content

	Topics	Lectures (Hours)	Tutorials (Hours)
1	**Introduction to Circuits** Concepts and terminology. Drivers and motivations for resistor circuit analysis. Introduction to Ohm Laws, combination of resistors – series and parallel.	**2**	**1**
2	**Nodal, Mesh & Loop Analysis Techniques** Circuit Analysis - Nodal, Mesh & Loop Analysis Techniques.	**3**	**1**
3	**Superposition, Thevenin/Norton Circuits** Additional Circuit Analysis - Superposition, Thevenin Theorem, Norton Theorem, Maximum Power Transfer.	**3**	**1**
4	Capacitance & Inductance, Series/Parallel Capacitance & Inductance, Series/Parallel.	**2**	**1**
5	**1st/2nd Order Transient Cct Analysis Techniques** First & Second-Order Transient Circults. Differential Equation & Step-by-Step Analysis Techniques.	**3**	**2**
6	Op-Amps, Negative feedback, Non-inverting and Inverting op-amps Ideal requirements of amplifiers, Op-Amps, Op-Amp characteristics, Loading effect analysis, Negative feedback and its significance, Non-inverting Op-Amps, Inverting Op-Amps, Op-Amp specifications, Various devices derived from Op-Amps.	**4**	**1**
7	**Continuous-time signals, manipulations, properties** Basic test signals and their properties and manipulations, Linear time-invariant (LTI) systems.	**2**	**1**
8	**LTI systems, Convolution** Linear time-invariant (LTI) systems (cont'd), System representations, System analysis and design.	**2**	**1**
9	**Laplace Transforms** Laplace transforms and Inverse Laplace transforms, Analysis of LTI systems using Laplace transforms, System transfer functions.	**2**	**1.5**

10	**Frequency based System Analysis** Eigen functions, System's response to test signals, Convolution, Impulse and Step responses, Finding LTI system output.	**2**	**1.5**
	Check for Hours	**=26**	**=12**

Assessment (includes both continuous and summative assessment)

a) Final Examination: 60%

b) Laboratory written report (Part 1): 20%

c) Laboratory written report (Part 2): 20%

CE/CZ 2005 – Operating Systems

Course Code	CE/CZ 2005							
Course Title	Operating Systems							
Pre-requisites	CE/CZ 1006: Computer Organisation and Structure and CE/CZ 1007: Data Structures							
No of AUs	3							
Contact Hours	Lectures	26	TEL	0	Tutorials	12	Laboratories	8

Course Aims

This course aims to develop your understanding of fundamental concepts and principles of modern operating systems, and to build your knowledge on the design and implementation of main operating system components.

Intended Learning Outcomes (ILO)

The course covers the following main components of an operating system: process management, memory management, file systems, and I/O. Upon the successful completion of this course, you shall be able to:

1. Explain fundamental concepts and principles of modern operating systems;
2. Identify the functions and services provided by each component of an operating system;
3. Describe how processes, memory, files, and I/O devices are managed in an operating system;
4. Analyze and evaluate algorithms that implement core functions of major operating system components; and
5. Implement functions for operating system components in accordance to design requirements using the concepts and principles learnt in this course.

Course Content

	Topics	Lectures (Hours)	Tutorials (Hours)
1	**Overview of Operating Systems (OS)** Functions, Types, Services, Evolution, Concepts, Structures. OS vs kernel vs scheduler. Batch vs real-time. From 'large' to 'small'. Influential legacy OS. UNIX, Linux, Windows and others.	**2**	**1**
2	**Processes and Threads** Process concept, operations on processes, inter-process communication. Concept of Thread, multi-threading models, relationship between processes and threads.	**2**	**1**
3	**Process Scheduling** Central Processing Unit (CPU) scheduling concepts, criteria and algorithms.	**2**	**1**
4	**Process Synchronization** The critical-section problem, mutual exclusion and condition synchronization, software/hardware solutions. Semaphores, Monitors, and classical synchronization problems.	**4**	**2**
5	**Deadlock and Starvation** Deadlock vs starvation, Characterization of deadlock prevention, avoidance, detection and recovery	**2**	**1**
6	**Memory Organization** Local vs physical address space, fixed-partitioning vs variable partitioning, Paging vs segmentation vs swapping.	**2**	**1**
7	**Virtual Memory Management** Virtual memory concepts, Page replacement algorithms, Frame allocation, and Issues related to demand paging.	**4**	**2**
8	**File System Organization and Implementation** File concept, access methods, structure and implementation. File allocation methods and disk space management. Directory structure and implementation.	**4**	**1**

9	**Input/Output (I/O) Management and Disk Scheduling** I/O software structure. Kernel I/O sub-system; Device drivers, Disk performance parameters. Disk scheduling policies.	**2**	**1**
10	**Issues in Real-time Operating Systems** Features of real-time kernels. Real-time CPU scheduling and real-time performance issues.	**1**	**1**
11	**Protection and Security** Goals of protection; Domain of protection; Protection models; Security problems and threats; Authentication; and Encryption	**1**	**1**
	Check for Hours	**=26**	**=13**

Assessment

a) Final Written Examination: 60%
b) Continuous Assessments: 40%

CE/CZ 2006 – Software Engineering

Course Code	CE/CZ 2006							
Course Title	Software Engineering							
Pre-requisites	CE/CZ 2002 (can be taken concurrently): Object Oriented Design and Programming							
No of AUs	3							
Contact Hours	Lectures	26	TEL	0	Tutorials	10	Student presentations	2

Course Aims

This course aims to equip SCSE students with foundation knowledge on issues and techniques required for the design and implementation of quality software.
You will have the necessary knowledge and skills to delve deeper into systems analysis (CZ3003) and advanced topics in Advanced Software Engineering (CZ3002).

Intended Learning Outcomes (ILO)

Upon completion of the course, you will be able to understand the roles and purposes of various activities in software engineering process. Specifically, you will be able to

1. Participate in all stages of the Software Development Life Cycle for a medium-size software project to deliver the required work products.
2. Elicit and specify requirements clearly and correctly.
3. Use good software design concepts and considerations.
4. Design and carry out test activities to verify that requirements have been met.
5. Perform simple project management of a medium-size software project.

Course Content

	Topics	Lectures (Hours)	Tutorials (Hours)	Presentations (Hours)
1	**Introduction to Software Engineering** Overview. Software Engineering Development Activities: stages and work products. Life Cycle Models: Waterfall, V, Spiral, RAD, Iterative, Agile. Case Study. UML Activity Diagram.	**3**	**0**	**0**
2	**Requirement Specification** Concepts: functional and non-functional requirements, software requirements. Elicitation: techniques, use case model. Specification: software requirement specification document, traceability. Validation: reviews, prototyping.	**4**	**1**	**0**
3	**Requirement Analysis** Conceptual Modelling: control, entity, interface objects. From Use Cases to Objects.	**2**	**1**	**0**
4	**Project Management** Overview. Concepts: work breakdown structure, task scheduling, and resource estimation. Activities: planning, organising, controlling, terminating. Tools.	**4**	**2**	**0**
5	**Design** Design Process: architectural design, detailed design, and software design document. Design Fundamentals Recap: Abstraction, Coupling and Cohesion, Decomposition and Modularisation, Encapsulation, Separation of Interface and Implementation. Key Issues: Concurrency, Event Handling, Error and Exception Handling, Separation of Business Logic from Interface, Data Persistence, and related Design Patterns (note that a few simple ones have been introduced in CE2002/CZ2002).	**6**	**2**	**0**
6	**Implementation and Testing** Mapping design models to code. Test case design: selection criteria, effectiveness. Levels: unit, integration, system. Strategies: object- oriented testing, white-box, black box. Formal Method: Finite State Machine. UML State Machine Diagram.	**5**	**3**	**2**
7	**Maintenance** Maintainability: reuse, version control.	**2**	**1**	**0**
	Check for Hours	**=26**	**=10**	**=2**

Assessment

a) Final Examination: 60%
b) Case study: 40%

CE 2007 – Microprocessor based Systems Design

Course Code	CE 2007							
Course Title	Microprocessor based Systems Design							
Pre-requisites	CE/CZ 1006: Computer Organization and Architecture							
No of AUs	3							
Contact Hours	Lectures	19	TEL	5	Tutorials	9	Student presentations	0

Course Aims

Microprocessors may be integrated within self-contained single-chip microcontrollers and include all the resources to execute programs. However, in many designs, resources such as read-only memory, random access memory, peripheral controllers, analogue-to-digital converters etc. need to be selected for specific reasons and connected to a microprocessor. This is often necessary to satisfy space, cost, performance, and data acquisition constraints. This course aims to provide you with a fundamental understanding of designing microprocessor-based systems. This includes understanding the range of components available in a microprocessor-based system, how they are connected and used for reliable operation.

Intended Learning Outcomes (ILO)

Upon completion of the course, you should be able to:

1. Compare alternative microprocessor and microcontroller features and select appropriate features for a given application task.
2. Interface any selected memory or peripheral devices to a microprocessor with due regard to signal, voltage levels, timing requirements and protocol.
3. Derive and explain how signals associated with sensors are processed via analogue circuits and acquired as quantized digital numbers using analogue-to-digital converters (ADC).
4. Write real-time programs to read data from switches, sensors or ADC readings, manipulate displays, and control actuators via amplifiers or digital-to-analogue converters (DACs).
5. Write real-time programs that make use of timers or timer interrupts.
6. Explain the working principles of semiconductor memories and how they are interfaced to the microprocessor.

Course Content

	Topics	Lectures (Hours)	Tutorials (Hours)	TEL (Hours)
1	**Microprocessor Landscape** Generic types of microprocessors and their applications; from 4-bit to 64-bit. Introduction to Digital Signal Processors, Digital Signal Controllers and SmartCard processors. List selected manufacturers and their processors; by, name, architecture, application and market share. Including major manufacturers Freescale, IBM, Intel, NXP, Renesas and Texas Instruments.	**1**	**1**	**1**
2	Microprocessor Packages Signals and Interfacing – Part 1 What do processors look like? Comprehensive coverage of packages including DIL, TSSOP, QFP and BGA etc. Interfacing to real memory and parallel peripherals with critical timing analysis and design. Unidirectional vs bi-directional vs tri-state. Functional signal classifications for a) busses: address, data. b) control signals including: CE, RW, OE, WE, IOR, IOW, BE, RAS, CAS, Wait, Ready, Operation Acknowledge, Byte vs Word selects. Interpreting datasheets. Timing parameters: data setup and hold times, signal voltages, rise and fall times. Address decoders: discrete logic vs CPLD.	**2**	**1**	**0**
3	Introduction to ARM Cortex-M Architecture and Programming ARM processors as intellectual property. ARM-Cortex-M1 FPGA-based soft-core processor. Cortex-M1 processor architecture and stare-up sequence. Memory map and isolated I/O. Cortex-M1 programming model, assembly language and embedded C programming.	**2**	**1**	**3**

4	**Peripherals, Interfaces and Applications – Part 1** Single-bit switch with mechanical 'bounce' and 'debounce' in hardware or software. Extension to scanned keyboards. Schmitt triggers. Counters and timers. Serial interfaces, their timing and applications: Review of Universal Asynchronous Receiver Transmitter (UART), then in detail a) Inter-Integrated Circuit (I2C) with clock stretching, and b) Serial Peripheral Interface (SPI), both with multiple examples. Tone-burst communications. E.g. audio Frequency Shifted key (FSK), Dual-Tone Multi-Frequency (DTMF) and infra-red 38 kHz modulated NEC protocol for TV remote controls.	**3**	**1**	**0**
5	**Analog Signal Conditioning and Interfacing** Analogue sensors for temperature, sound, light and displacement. Operational amplifiers for signal conditioning and low pass filter. Voltage analogue-to-digital conversion: Successive approximation, integrating, Sigma-Delta, Flash and pipe-lined. Conversion time, resolution, linearity, monotonicity, stability, equivalent number of bits. Sample and hold. Voltage-to-frequency converters. Digital-to-analogue converters (DACs). Pulse width modulation. Motors.	**3**	**1**	**0**
6	**Displays** Driving single light emitting diodes (LEDs). Multiplexed and non-multiplexed 7-segment LEDs. Dot matrix LEDs and Liquid Crystal Displays (LCD).	**1**	**1**	**0**
7	**Signals and Interfacing – Part 2** Interrupt and timer interrupt. Direct memory access (DMA). Asynchronous interrupts: maskable and non-maskable including Reset. Memory fault exception.	**2**	**1**	**0**
8	**Peripherals, Interface and Applications – Part 2** Characteristics of Inter-IC Sound (I2S), Controller Area network (CAN) and Universal Serial Bus (USB).	**1**	**1**	**0**
9	**ARM Debug, System Design Issues**	**1**	**0**	**0**
10	**Semiconductor Memory Technology and Characteristics, Fabrication**	**3**	**1**	**1**
	Check for Hours	**=19**	**=9**	**=5**

Assessment

a) Final Examination: 60%
b) Assignment and Report: 20%
c) Laboratory Quiz: 20%

CE9010 – Introduction to Data Science

Course Code	CE9010					
Course Title	Introduction to Data Science					
Pre-requisites	CE/CZ1003, CE/CZ1011 (or equivalent - which is basics in statistics)					
No of AUs	3					
Contact Hours	Lecture	26	Tutorials	13	Laboratories	10

Course Aims

This course aims to develop skills in data science. In 2012, the Harvard Business School called data scientist the sexiest job of the 21st century. Why has data scientist become a most in-demand job? Today massive amounts of data are available in all areas of science, government and industry. Exploited sensibly, these raw data can significantly improve the efficiency of research, services and industries in as many fields as healthcare, engineering, finance, telecommunications or urban development just to name a few.

This course is designed for students who recognize the value of data science, but are not majoring from the School of Computer Science and Engineering (SCSE) and certain other programmes (see Appendix 3). The course provides you with an understanding of the fundamentals of data science and the practical skills set required to be a data scientist, including the design and implementation of basic techniques.

Intended Learning Outcomes (ILO)

By the end of this course, you (as a student) would be expected to be able to:

1. Identify and apply data analysis techniques to different data problems.
2. Implement data science techniques with Python.
3. Analyze and solve real-world data science projects.
4. Engage more professionally in written and oral scientific communication,
5. Work cohesively and effectively in teams.

Course Content

	Topics	Lectures (Hours)	Tutorials (Hours)	Laboratories (Hours)
1	**Introduction to Data Science** Why data science is essential? Big data era, powerful computational infrastructure, large IT companies invest in DS.	2	0	0
2	**Review of Linear Algebra and Coding Principles** Revise system of linear equations, standard solver, eigenvalue decomposition, Tikhonov regularization, and gradient descent technique. Review coding principles such as libraries, data structures, execution in Python.	2	2	0
3	**Linear Regression** Linear regression uses linear system of equations to solve real-valued prediction problem. Apply gradient descent technique to solve linear regression problems.	2	2	2
4	**Logistic Classification** Classification technique based on linear representation score and logistic loss function. Application to classification of images as cats and dogs.	2	1	0
5	**Support Vector Machine** Introduce SVM classification techniques from hard to soft class representations. Kernel trick for non-linear classification.	2	1	2
6	**Unsupervised Clustering** Present k-means technique for Gaussian data distribution. Discuss non-linear data grouping techniques such as Shi-Malik.	2	1	0
7	**Feature Extraction** Design Principal Component Analysis and Non-Negative Matrix Factorization. Application to face generative models.	2	1	2
8	**Recommender Systems** Introduce the two main classes of recommender systems; collaborative- and content-based filtering techniques.	2	1	0
9	**Neural Networks** Presentation of artificial neural networks, a.k.a. deep learning, with fully connected neural networks, learning process, convolutional neural networks and recurrent neural networks.	8	4	4
10	**Revision**	2		
	Check for Hours	= 26	= 13	= 10

Assessment (includes both continuous and summative assessment)
a) Final Examination: 40% b) Laboratory: 40% c) Project: 20% Note: Final examination is closed book.