## School of Computer Science

### List of Elective Courses

| CourseCode | Course Name | L | T | P | C |
|---|---|---|---|---|---|
| CS 570 | Model Checking and Software Verification | 3 | 0 | 0 | 3 |
| CS 572 | Modeling and Simulation of Systems | 3 | 0 | 0 | 3 |
| CS 462/CS 662 | Markov chains and their Applications | 3 | 0 | 0 | 3 |
| CS 611 | Computational Complexity | 3 | 0 | 0 | 3 |
| CS 520 | Combinatorial Optimization | 3 | 0 | 0 | 3 |
| CS 511 | Approaches to Software PerformanceEnhancement | 3 | 0 | 0 | 3 |
| CS 521 | Foundations of Functional Programming | 3 | 0 | 0 | 3 |
| CS 513 | Introduction to Information Security | 3 | 0 | 0 | 3 |
| CS 541 | System on Chip Design: Hardware-Software Perspectives | 3 | 0 | 2 | 4 |
| CS 620 | Algebraic Algorithms | 3 | 0 | 0 | 3 |
| CS 606 | Programming Language Paradigms | 3 | 0 | 0 | 3 |
| CS431 | Optimization: Theory and Algorithms | 3 | 0 | 0 | 3 |
| CS531 | High Dimensional Data Science | 3 | 0 | 0 | 3 |
| CS440 | Computer Graphics using OpenGL | 3 | 0 | 0 | 3 |
| CS441 | Foundations of Digital Transformation | 2 | 0 | 0 | 2 |
| CS442 | Geometric Modeling | 3 | 0 | 0 | 3 |

**CS570** **Model Checking and Software Verification** **(3-0-0-3)**
**Course Type: UG/PG**

**Description**

Arguing for program correctness is a challenging task. But it is non-optional, especially as the software has permeated our lives, in forms that are many times even safety- or business-critical.Formal Verification is an attractive and increasingly appealing alternative to simulation and testing.While the latter only explores a subset of possible behaviors, and leaves open the question of whether the unexplored behaviors may contain a fatal bug, formal verification explores all behaviors exhaustively. This course will focus on model checking as an approach to do formal verification.

**Syllabus**

Propositional and Predicate Logic (Overview); Modelling systems – Kripke Structure, Concurrent Systems, and NuSMV Model Checker; Linear-time properties – paths, traces, invariants, safety, liveness, fairness; Automata on finite and infinite words - Model checking omega-regular properties.

Temporal logics - Linear temporal logic (LTL), Computation tree logic (CTL)– Introduction toPromela and SPIN; Binary decision diagrams (BDDs) and Symbolic Model Checking.

Program Verification – Hoare Triples and Hoare Logic; Propositional satisfiability - SAT-based Model Checking (BMC, Inductive invariants), Introduction to CBMC; Predicate abstraction and Counterexample-Guided Abstraction Refinement (CEGAR).

**References**


1. Principles of Model Checking by *Christel Baier* and *Joost-Pieter Katoen*
2. Logic in Computer Science by *Michael Huth* and *Mark Ryan*
3. Model Checking (Second Edition, The Cyber-Physical Systems Series) by *Edmund Clark,Orna Grumberg, Daniel Kroening, Doron Peled,* and *Helmut Veith*

**CS572** **Modeling and Simulation of Systems** **(3-0-0-3)**
**Course Type: UG/PG**

### Description

This course offers an introduction to modeling and simulation of discrete-event systems with examples from several application areas such as computing systems, manufacturing, queues and computer networks. The approach of the course would be to get the student to start modeling and simulating simple representative systems as soon as possible and to explore the system behavior via simulations before delving into the theory to understand the observed behavior. The course will make use of Python's SimPy library.

### Syllabus

Introduction to models and simulation, the need for simulation, types of models.
Introduction to Python's SimPy library using examples.
Review of basic probability theory and Markov chains.
Random number generation.
Simulation of queues and their analysis.
Simulation of manufacturing systems.
Simulation of computer networks.
Parallel and distributed simulation: How to incorporate parallelism into the
Simulation execution while still obtaining the same results as a sequential execution.

### References


1. Discrete-Event Simulation: A First Course, by Leemis and Park
2. Probability, Markov Chains, Queues, and Simulation, by William J. Stewart
3. Parallel and Distributed Simulation Systems, by Richard Fujimoto
4. SimPy Documentation: https://simpy.readthedocs.io/en/latest/

**CS462/CS662** **Markov Chains and Their Applications** **(3-0-0-3)**
**Course Type: UG/PG**

**Description:**

The course deals with applications of Markov chains techniques in certain areas of computer science and of biology. The unifying aspect in these applications is the role played by mixing time analysis, which is the focus of the course.

**Syllabus**

In a typical combinatorial optimization problem, each instance $x$ is associated with a state space $S_X$, usually of size exponential in the size of $x$, where each element of $S_X$ has an associated cost (or a value). The problem is to find a state with minimum cost (or with maximum value). In the Markov chains approach to solving such a problem, a Markov chain is associated with each instance with the property that the goal state has the highest probability in the stationary distribution of the chain. Success of this approach crucially depends on the mixing time of the associated chain, that is, how quickly does the chain come close to its stationary distribution.

Markov chains have also been used to model evolution for the finite population case lending themselves to stochastic effects. For a population of size $N$ of $m$ types, a state of the chain is the labelling of the $N$ individuals each into one of the $m$ types. The population goes from one state to another through reproduction, mutation, and selection. The specific way in which each of these happens determine the the transition probability matrix of the chain. The result of the evolution modelled by the chain is given by the stationary distribution of the chain. In most cases however, it is not known how to determine the stationary distribution directly, we need to simulate the chain 'sufficiently long' and then sample from the resulting distribution to obtain statistical properties of the distribution. We need to determine the mixing time of the chain to find out how long is 'sufficiently long'. It has been established recently that the expected motion of such evolutionary chains turns out to be a dynamical system, the trajectory of which determines the mixing time properties like rapid mixing.

Approximate outline of the course: Ergodicity theorem of Markov chains. Mixing time of Markov chains. Combinatorial optimization using Markov chains. Metropolis algorithm. Notion of conductance and its relation to rapid mixing. Necessary and sufficient conditions for Markov chains approach to succeed for combinatorial optimization. Coupling of Markov chains. Markov chains modelling of molecular evolution. Quasispecies model. Viral evolution and notion of error threshold. Evolution for the finite population case. Expected motion of evolutionary Markov chains and corresponding dynamical system. Detailed analysis when the population is of two types. A qualitative understanding of the general $m$ types case.
Each student will be required to do a term paper.

**Textbooks and References**


1. Finite Markov Chains and Algorithmic Applications, Olle Haggstrom, Cambridge Univ. Press, 2002.
2. Markov Chains, JR Norris, Cambridge Univ. Press, 1997.
3. Algorithms for random Generation and Counting: A Markov Chain Approach, Alistair Sinclair, Birkhauser, 1993.
4. Markov chains and mixing times, David A. Levin, Yuval Peres, and Elizabeth Wilmer, American Mathematical Society, 2006.
5. Relevant papers.

**CS611** **Computational Complexity** **(3-0-0-3)**

**Course Type: UG/PG**

**Description**

In this course, we study various techniques used to classify computational problems according to their inherent complexity. How do we quantify and answer the following questions: “Is problem *A* more difficult than problem *B*? E.g. Is multiplication more difficult than addition?”, “Are there problems that are inherently difficult?”, “Are there problems that can never be solved, irrespective of the resources used?”, “Does randomization help?”

The more difficult a task, the more resources we use to solve it. The resources used by a computing machine are: time, space and randomness. We will study the usage of these resources and their interactions.

**Syllabus**

The computational model – brushing up algorithms, Big-o notation, Turing machines, P, NP and NP completeness – Reducibility, Cook-Levin theorem, Examples.
Diagonalization technique - Deterministic and non-deterministic time hierarchy, space hierarchy.
Space complexity – Configuration graphs, Savitch’s theorem, L, NL, Reachability, Immerman Szelepcsényi theorem.
Polynomial hierarchy (PH) – TQBF, other ways of defining PH, PH collapse.
Circuits – Equivalence with P/poly, counting argument to show the existence of hard functions, Karp-Lipton theorem.
Randomized Computation – Randomness as a resource, examples, complexity classes that use randomness.
Counting Classes (if time permits)- #P, permanent, Toda’s theorem.
Interactive proofs (if time permits)– Arthur-Merlin games, Interactive proofs, graph non-isomorphism,IP = PSPACE.

**Textbooks and References**


1. Sanjeev Arora, Boaz Barak: Computational Complexity - A Modern Approach. Cambridge University Press 2009, ISBN 978-0-521-42426-4, pp. I-XXIV, 1-579
2. Christos H. Papadimitriou: Computational complexity. Addison-Wesley 1994, ISBN 978-0-20153082-7, pp. I-XV, 1-523

**CS520** **Combinatorial Optimization** **(3-0-0-3)**

**Course Type: UG/PG**

Pre-requisites: Algorithms, Linear algebra

**Description**

The course introduces optimization and combinatorial techniques used to solve optimization problems. A special focus will be on linear programming.

Introduction to Optimization problem. Example problems like Travelling salesman problem, bipartite matching, max SAT etc.
Mathematics primer - Convex sets and functions, linear algebra

Linear programming introduction - Different forms of linear programming problems, Examples of problems reduced to linear programming.
The Simplex algorithm - Geometry of linear programming. Feasible solutions, basic feasible solution, the algorithm.
Duality - Dual of a linear program, Farkas' lemma, Primal-Dual algorithm schema.
Examples will be Ford-Fulkerson max-flow and Dijkstra's shortest path.
Linear programming in Polynomial time - Khachiyan's ellipsoid method and/or Karmarkar's Integer point method.

**Textbooks and References**


1. Combinatorial optimization by Papadimitriou and Steiglitz
2. Theory of Linear and Integer programming by Schrijver
3. Combinatorial Optimization by Schrijver

**CS511** **Approaches to Software Performance** **(3-0-0-3)**

**Course Type: UG/PG**

Pre-requisites: Basic programming and algorithms

**Description**

The course will introduce the students with some fundamental knowledge of software engineering, some algorithmic tricks to improve the code and some tools to improve performance of softwares

Topics: Following topics will be covered.

- Software development life cycle, process models, module design, testing, modelling
- Choosing variable names, function names, guidelines for creating designer's type, globals, organizing straight line code, choosing most effective ordering of case statements, selecting and exiting loops, length, nesting level of loop, etc, Unusual control structures - goto, recursion, multiple returns from a routine, table driven approach versus inheritance, taming deep nesting, McCabe measure of complexity.
- Different compiler optimizations e.g., code motion, common sub-expression elimination, constant propagation, etc and the optimizations which make the stage for parallelization e.g., loop skewing, interchange, software pipelining , etc which can make the code more efficient
- Algorithm design tricks : using binary search over linear search, tricks to reduce the space (e.g., using unsigned char if the integer is too small), dynamic allocation, verification using assertions, using hand-made quick-sort instead of system qsort or insertion sort, comparing sorting and searching algorithms in terms of time and space and use appropriately
- Code improvement techniques : Debugging, refactoring, caching, etc and tools e.g., coverage tool (gcov), profiling tool (gprof), memory leakage identification tool (valgrind), debugger (gdb)

**Textbooks and References**


1. Fundamentals of Software Engineering by Rajib Mal
2. Compilers: Principles, techniques and tools by Aho, Lam, Sethi, Ullman
3. Programming Pearls (second edition) by Jon Bentley
4. Code Complete by Steve McConnell

**CS521** **Foundations of Functional Programming** **(3-0-0-3)**

**Course Type: UG/PG**

**Description**

The course will introduce students to the functional programming paradigm and itsfoundations.

Topics covered

- What is functional programming, history and evolution of functional programming languages
- Types, declaring types, lists, operations
- Defining functions, composition
- Currying, lambdas, untyped and simply typed lambda calculus
- Defining functions -- conditionals, guards, pattern matching
- List comprehensions
- Higher order functions — map, filter, folds, Combinatory Logic
- Defining types, type classes
- Functional data structures — lists, trees, bst
- IO, hello world
- Monads and functors

**Textbooks and References**

Programming in Haskell - Graham Hutton

Learn you a haskell for great good - Miran Lipovaca

**CS513** **Introduction to Information Security** **(3-0-0-6)**

**Course Type: UG/PG**

**Description**

Computers and Internet have become an integral part of our lives. On daily basis, we use laptops, mobiles, LAN, Internet, Web and Cloud. As we can access computing and entertainments resources all over the world, cyber criminals have access to our personal data, which they can exploit in many ways. Hence, it is important that we make our systems and network secure.

The course covers basic concepts and principles of Information security and fundamental approaches to secure computers and networks.

The course assumes that students have basic understanding of computer organization, OS, and networks.

**Syllabus**

Module 1: Basics of Computer Security (threats, attacks, confidentiality, integrity, availability)
Module 2: Software Security – attacks and defences (buffer overflows, viruses, secure software development)
Module 3: Web Security
Module 4: : Internet security, DOS, DDOS, etc
Module 5: Basic concepts of Cryptography
Module 6: Human factors in security
Module 7: Network Security (Internet background, attacks, Firewall, NAT, VPN)
Module 8: Wireless network Security (Wireless network background, security mechanisms)
Module 9: Hacking defined
Module 10: OS and application vulnerabilities
Module 11: Blockchain Security concepts
Module 12: Invited talks from Industry professional
Module 13: Student Project discussions

**Textbooks and References**

1. *Security in Computing*, Pfleeger & Pfleeger, 4th ed.
2. *Introduction to Computer Security*, Goodrich & Tamassia, 1st ed.

**CS606** **Programming Language Paradigm** **3-0-0-3**

The purpose of the course is to obtain familiarity with different paradigms of computer programming, chiefly the object-oriented and the functional. The course will look at various aspects of programming languages -- types, memory, binding and scope, blocks, procedural abstraction, data abstraction, encapsulation, polymorphism, control flow, concurrency etc. -- and how they are present in various procedural, object-oriented and functional programming languages. As a case study we will take a closer look at select OOP and functional programming languages.

**Textbooks**

1. Findlay, William, Watt, David Anthony - Programming Language Design Concepts
2. Maurizio Gabbrielli, Simone Martini, Programming Languages: Principles and Paradigms

**Additional Readings**

1. Bjarne Stroustrup, The C++ Programming Language
2. Robert Harper, Programming in Standard ML
3. Harold Abelson, Gerald Jay Sussman, Julie Sussman, Structure and Interpretation of Computer Programs

---

**CS620** **Algebraic Algorithms** **3-0-0-3**

For B.Tech 3rd and 4th year, M.Tech, PhD.
Prerequisite: Algorithms course.

**Syllabus:**
GCD Algorithm, fast matrix multiplication, Quick introduction to groups: permutation groups, Normal subgroups, relation between group homomorphisms and kernels, rings, ideals, fields, irreducible polynomials, How to construct fields. Fast polynomial multiplication, Polynomial factorization, integer factorization, primality testing.

**Textbooks:**

1. Joseph Gallian: Contemporary Abstract Algebra

**References:**

1. Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein: Introduction to Algorithms
2. Lecture notes of 'Algebra and Computation' by Ramprasad Saptharishi(TIFR). https://www.tcs.tifr.res.in/~ramprasad/courses/2019algComp/
3. Lecture notes of 'Algebra and Computation' by Madhusudan (then in MIT) http://people.csail.mit.edu/madhu/ST12/
4. Joachim von zur Gathen: Modern Computer Algebra.
5. Rudolf Lidl and Harald Niederreiter: Introduction to Finite Fields and their applications.

---

**CS541 System on Chip Design: Hardware-Software Perspectives (3-0-2-4)**

Course Type: Open Elective (PG, UG- sem VII); Ph.D. (EE and CSE)

**Description:**
This course is focused on design and development aspects of system-on-chip design. Students will learn about designing a system that includes both microprocessors and hardware accelerators. Microprocessor programming, design of hardware accelerator, bus architecture, memory communication etc. will be the principal learning outcomes. This course does not cover backend design (fabrication).

**Syllabus:**
This course will take a deep dive in design of system on chip (SoC) involving hardware and software. Design and development of SoC comprising a microprocessor, hardware accelerators, external memory interfaces and the bus/interconnect architecture connecting them together. SoC FPGAs will be taken as the underlying platform to learn the concepts. Component design for SoC using High Level Synthesis (HLS) and Verilog Hardware Description Language, packaging components as IP cores, bus architectures, interconnect architecture to connect the components to a modern microprocessor like Arm, programming the microprocessor (embedded software development for SoC); Physical design flow (synthesis, placement and routing, timing verification), functional verification transaction level modeling; Architectural exploration of accelerators, C-to-RTL verification; Memory system hierarchy and architecture for SoC; SoC system performance analysis Lab sessions: high level synthesis, using Verilog to design hardware, programming of a modern microprocessor (embedded software development). The course will have a term design project (TDP) to be carried out in groups. TDP needs to be carried out during M.Tech. Lab hours as well as outside of it. TDPs would require teams to build hardware accelerators, program microprocessors and enable communication between various components. TDPs would be based on problems provided by the instructor or suggested by student teams. Real world SoCs are designed in groups and hence TDP is part of this course.

**Textbooks**
1. Michael Fingeroff. High Level Synthesis Blue BookPrinciples of Computer System Design, 1st Ed. Xlibris. 2010. ISBN: 978-1450097246 2. Michael Keating. The Simple Art of SoC Design - Closing the Gap between RTL and ESL, 1st Ed. Springer-Verlag New York 2011. ISBN: 978-1-4419-8585-9 3. Xilinx user manuals for Vivado HLS, Vivado, Vivado SDK, SDAccel 4. Arm DS-5 user manual

**References** 1. Transaction-Level Modeling with SystemC. F. Ghenassia, 1st Ed. Springer US 2005. ISBN:978-0387-26232-1 2. T. Grotker, S. Liao, G. Martin and S. Swan. System Design with SystemC, 1st Ed. Springer US 2002. ISBN: 978-1-4020-7072-3

**CS431** **Optimization: Theory and Algorithms** **(3-0-0-3)**

Course Type: UG (3rd, 4th year –CSE & MnC), PG (M.tech), Ph.D.

Prerequisite: Linear Algebra, Calculus, Programmingexperience (e.g MATLAB, Python)

**Course Objective:**
The objective is to be able to model problems as optimization problems, identify easy and hardinstances of optimization problems, learn theory and techniques for solving linear and nonlinearprogramming problems.

**Course Content:**
Modelling optimization problems, classes of problems- discrete, continuous, linear, quadratic, unconstrained and constrained
Unconstrained optimization – necessary and sufficient conditions, iterative algorithms: steepestdescent, Newton's method, conjugate gradient
Convex Sets, Convex functions, Convex Optimization, Farkas Lemma
Linear Programming- applications in transportation, network flow, Simplex Method, Duality inLPs
Constrained Optimization – KKT conditions, Duality, Conditions for Strong DualityApplications in Machine Learning

**Text book(s):**

1. David G. Luenberger and Yinyu Ye, Linear and Nonlinear Programming 3rd edition,Springer, ISBN: 978-0387745022

2. Edwin K.P. Chong and Stanislaw H. Zak, An Introduction to Optimization, 2nd edition,Wiley-Interscience Series in Discrete Mathematics and Optimization, ISBN: 0-471-39126-3
Stephen Boyd and Lieven Vandenberghe, Convex Optimization, Cambridge UniversityPress, ISBN: 0-521-83378-7

**Reference(s):** Jorge Nocedal and Stephen Wright, Numerical Optimization, 2nd edition,Springer, ISBN: 978-0-387-30303-1

R. Fletcher, Practical Methods of Optimization, Wiley, ISBN: 978-0471494638
Bertsimas and Tsitsikilis, Introduction to Linear Optimization, Athena Scientific, ISBN: 978-1886529199

**CS531** **High Dimensional Data Science** **(3-0-0-3)**

Course Type: UG (3rd, 4th year), PG (M.Tech), Ph.D.
Prerequisite: Algorithm Design, Probability, Linear algebra,Calculus

**Course Objective:**
The objective is to understand the theoretical foundations of high dimensional data science.

**Course Content:**
1. High Dimensional Space - The geometry of high dimension, properties of unit ball,Random projects and Johnson-Lindenstrauss Lemma, Separating gaussians
2. Singular Value Decomposition (SVD) - Introduction to SVD, best k-rank approximations,left singular vectors, power method for SVD, applications of SVD.
Compressed sensing

**Text book(s):** None

**Reference(s):**
1. Foundations of Data science by Blum, Hopcroft and Kannan
2. Understanding machine learning by Shai Shalev-Shwartz and Shai Ben-David

**CS440** **Computer Graphics using OpenGL** **(3-0-0-3)**

Course Type: UG ($3^{rd}$,$4^{th}$ year)

Prerequisite: CS 101 AND CS (113 OR 220 OR 400)

**Course Objective:** This is an introductory course on Computer Graphics. It is an open electivefor CS and Non-CS students with exposure to Data Structures and Programming. Students willbe graded based on Mid-Semester exam, Programming Assignments, a Mini-Project and an End-semester examination. There will be no separate lab associated with the course.

**Course Content:**
Introduction to Computer Graphics: Graphics Application and Software, Input Devices, Vectorand Raster Refresh Displays, LCD displays.
Two-Dimensional Transformations: Homogeneous Coordinates, Matrices and Point transformation Translation, Scaling and Rotation
Three-Dimensional Transformations: Rotation about an Arbitrary Axis in Space, Perspective Views, Camera models, Orthographic Projections, Axonometric Projections, Oblique Projections, View volumes for projections.
Viewing in 3D: 3D viewing pipeline, Specifying an Arbitrary 3D View, Combined transformation matrices for projections and viewing, Coordinate Systems and matrices, cameramodel and viewing frustrum.
Clipping, filling and Scan conversion: Scan conversion of lines, circles Filling polygons edge data structure, Line Clipping algorithms, Clipping Polygons, problem with multiplecomponents. Problems of Aliasing.
Graphics Programming using OpenGL: OpenGL, features in OpenGL, OpenGL operations, Abstractions in OpenGL – GL, GLU & GLUT, 3D viewing pipeline, viewing matrix specifications, a few examples and demos of OpenGL programs.
Solid Modelling: Representing Solids, Regularized Boolean Set Operations, Sweep Representations, Spatial-Partitioning Representations - Octree representation, B-Reps, Constructive Solid Geometry
Visible-Surface Determination: Techniques for efficient Visible-Surface Algorithms, Categories of algorithms, Back face removal, The z-Buffer Algorithm, Scan-line method, Painter's algorithms, Ray Tracing, BSP trees
Illumination and Shading: Reflectance properties of surfaces, Ambient, Specular and Diffuse reflections, Atmospheric attenuation, Phong's model, Gouraud shading
Plane Curves and Surfaces: Curve Representation, Nonparametric Curves, Parametric Curves, Cubic Splines, Bezier Curves, B-spline Curves, Quadric Surfaces. Bezier Surfaces.

**Text book(s):**

1. D. Hearn and M. Pauline Baker, Computer Graphics with OpenGL, 4th Ed. PearsonEducation

J. D. Foley, A. Van Dam, S. K. Feiner and J. F. Hughes, Computer Graphics - Principles,3rd Ed. Pearson Education

**Reference(s):**

1. Peter Shirley, Fundamentals of Computer Graphics, 4th Edition, CRC Press
2. Sumanta Guha, Computer Graphics Through OpenGL, 2nd Edition, CRC Press
3. D. F. Rogers and J. A. Adams, Mathematical Elements for Computer Graphics, 2ndEdition, McGraw-Hill International Edition, 1990.
4. F. S. Hill Jr., Computer Graphics using OpenGL, Pearson Education, 2003

**CS441** **Foundations of Digital Transformation** **(2-0-0-2)**

Course Type: UG (4th year), PG(M.Tech)

Prerequisite: CS101

**Course Objective:** Businesses around the globe are undergoing major transformations as their business model, business processes, underneath systems and platforms are continuously aligning with new and upcoming digital technologies. In this course we will study the basics of Digital Transformation, threats to existing businesses, and ways of transforming existing businesses using digital technologies. We will also study, in brief, the concept and applications of various digital technologies.
The course will be useful for graduating BTech and MTech students. The course will help them in understanding and applying digital technologies, not only for marginal productivity gains, but as an enabler for innovation and new business ideas.

Course Content:
Module I: What is Digital Transformation? [2 hours]
In this module we study how technologies change businesses. We will look at the exponential rateat which technologies are advancing and how this creates opportunities for existing and new businesses. We will also learn about how everything is becoming digital and its impact on our livesand businesses.
Module II: What is disruptive Innovation? [2 hours]
In this module, we will study how new technologies (eg. Cloud) can threaten established companies. We will look at supply and demand side disruptions.
Module III: The Internet, Web, Cloud, AI and Machine Learning [4 hours]
We will study how the Internet and Related technologies can be used for transforming businesses.We will start with basic concepts of the Internet and then we will study Web and Cloud Technologies.
Module IV: Digital Transformation Technologies [3 hours]
We will briefly look at concepts, and disruptive applications of existing and new digital technologies.
Module V: Managing Disruptive Changes [2 hours]
We will learn how companies can respond to disruptive changes and why it is so hard for companies to execute these theories. We will also study how development of platforms may help organizationsin the digital journey.
Module VI: Digital Transformation Case Studies [6 hours]
We will invite an expert from Industry to present a few case studies in digital transformation.
Module VII: Digital Transformation project Presentation [4 hrs]
A team of 3-4 students will study a business happening in Indian cities or villages. Team will come up with ideas on how to digitally transform the business, if possible. Students will present the ideain the class within 10 mins time period.
Module VIII: Management of Enterprise orientation towards Digital Technologies [4 hours] :Culture, Process Transformation, Alignment, Strategic Focus etc.

**Text book(s):**


1. George Westerman, Leading Digital: Turning Technology into Business Transformation
2. Ashish Pachory, Aligned to Win, Zorba books
3. David L Rogers, The Digital Transformation Playbook, Columbia Business School
4. Dominic M Mazzone, Digital or Death: Digital Transformation - The Only Choice for Businessto Survive, Smash, and Conquer

**Reference(s):**


1. Digital Transformation: A Roadmap For Billion-Dollar Organizations, by CapGemini
2. Aligning Technology with Business for Digital Transformation, Ashish Pachory, BEPPublication
3. Born to be digital: How leading CIOs are preparing for a digital transformation, by EY
4. Digital transformation: Creating new business models where digital meets physical, by IBM
5. Clayton M. Christensen, Michael Raynor, Rory McDonald. "What is Disruptive Innovation?"Clayton M. Christensen, Michael Raynor, Rory McDonald, Harvard Business Review, December2015. (HBS)
6. Joshua Gans, "The Other Disruption," Harvard Business Review, March 2016. (HBS)
7. G. Parker, Sangeet Paul Choudary, Platforms, Pipelines, and the New Rules of Strategy,"Marshall W. Van Alstyne, Geoffrey, Harvard Business Review, April 2016. (HBS)


---

**CS442** **Geometric Modeling** **(3-0-0-3)**

Course Type: UG (3rd, 4th year), M.tech, Ph.D.
Prerequisite: Programming, Calculus and Linear Algebra

Course Objective:
This is an introductory course in modelling techniques for 3D objects. It is an institute elective for CS and Non-CS students. It covers a wide range of different ways of representing the geometry of objects. The emphasis in this course will be on the theory and basic principles of constructing models and reasoning about the mathematics of models. Students will be graded based on Quizzes, Mid-Semester exam, Programming Assignments, and an End-semester examination.

Course Content:
1.Coordinate system: Local and global, Point and vectors, Linear maps and affine maps
2. Polygon: Convex and concave; polyhedron; convex hull
3. Parametric equation: Basic differential geometry of curves and surfaces
4. Curves: Explicit/Implicit equations of curves, Hermite curves
5. Bezier Curves: Bezier basis functions, Control points, Continuity, 6. Composite Bezier curves, Rational Bezier curves
6. B-Spline curves: B-Spline basis function, Closed B-Spline curve, NURBS
7. Surfaces: Bezier surfaces, Bicubic Bezier patch, Composite Bezier surfaces, B-Spline Surfaces, Matrix form
8. Solids: Parametric solids, Sweep solids, deformation of solids, Boundary models, Space partitioning models, Boolean models
9. Geometric modelers: Boundary representation modelling, Boundary representation operations
10. Voronoi diagram; Delaunay triangulation

Text book(s):


1. Geometric Modeling by Mortenson, 2nd Edition, Wiley Publishers
2. Curves and Surfaces for Computer Aided Geometric Design by Farin, Academic Press


Reference(s):


1. Applied Geometry for Computer Graphics and CAD, by Marsh 2nd Edition, Springer
2. Computational Geometry in C, by O'Rourke, 2nd Edition, Cambridge University Press


---