Computational Optimization & Applications

Siju Swamy

2025-11-17

Table of contents

Course Vision & Context

The Optimization Revolution in Computational Sciences

In an era dominated by Artificial Intelligence, Machine Learning, and Data Science, *optimization forms the fundamental backbone* that powers intelligent decision-making systems. From recommending your next movie to orchestrating global supply chains, from training deep neural networks to scheduling autonomous vehicles—optimization algorithms are the invisible engines driving technological progress.

This course positions you at the intersection of *mathematical theory* and *computational practice*, equipping you with both the conceptual understanding and hands-on skills to design, implement, and deploy optimization solutions for real-world challenges.

Course Syllabus (48 Hours / 12 Weeks)

Course Overview

Course Code: 20MAT382

Course Name: Computational Optimization and Applications

Duration: 12 Weeks (4 hours/week)

Credits: 4

Total Marks: 150 (Internal: 70 + External: 80)

Intensive Learning Approach

Accelerated project-based curriculum focusing on core optimization concepts with immediate practical application through integrated micro-projects.

Course Objectives

S.No	COURSE OBJECTIVES
1	To gain a comprehensive understanding of optimization concepts and their real-world relevance, emphasizing Python as a practical tool for optimization.
2	To develop proficiency in Python for optimization, including formulating and solving linear programming problems, implementing nonlinear optimization and analysing optimization solutions.
3	To acquire skills in project planning and optimization techniques using Python.
4	To master optimization techniques in the context of machine learning, including Gradient Descent, Stochastic Gradient Descent, and various optimization algorithms, all implemented in Python.
5	To apply optimization knowledge and Python skills to solve combinatorial and graph-based problems, while also considering the ethical aspects of optimization in engineering, logistics, and decision-making.

Course Outcomes

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

CO Code	COURSE OUTCOMES	REVISED BLOOM'S TAXONOMY LEVEL
CO1	Demonstrate a thorough understanding of optimization concepts, problem types, and their real-world applications.	3
CO2	Formulate and solve linear programming problems using Python.	3
CO3	Implement nonlinear optimization algorithms, and analyse optimization solutions using Python.	2
CO4	Develop practical project planning skills and proficiency in applying heuristic algorithms to real-world scenarios.	2
CO5	Demonstrate mastery of optimization in Machine Learning, with the ability to apply Gradient Descent, Stochastic Gradient Descent.	3

Core Competencies

- Formulate real-world problems as mathematical optimization models

- Implement optimization algorithms in Python using industry tools
- Analyze and validate optimization solutions
- Develop end-to-end optimization systems for practical applications

Assessment Plan (70 Marks Internal)

Continuous Evaluation (70 Marks)

A. Theory Components (20 Marks)

- **Internal Exam 1**: 10 marks (Week 6)
- **Internal Exam 2**: 10 marks (Week 12)

B. Practical Components (50 Marks)

- **Assignments/Micro Projects**: 15 marks (3 projects × 5 marks each)
- **Lab Exams**: 10 marks (2 exams × 5 marks each)
- **Day-to-Day Lab Work**: 15 marks
- **Attendance**: 10 marks

External Examination (80 Marks)

- **End Semester Theory Exam**: 80 marks

12-Week Delivery Plan (4 Hours/Week)

Phase 1: Foundation & Linear Methods (Weeks 1-4)

Week 1: Optimization Fundamentals & Python Setup (4 hours)

- **Theory (2h)**: Optimization concepts, problem classification, LP formulation
- **Lab (2h)**: `Python` environment setup, `PuLP` introduction
- **Lab Work**: Basic LP implementation (1 mark)
- **Micro-Project 1 Launch**: Campus facility location problem

Week 2: Linear Programming & Solution Methods (4 hours)

- **Theory (2h)**: Graphical method, Simplex algorithm
- **Lab (2h)**: `PuLP` implementation, constraint handling
- **Lab Work**: Complex constraint implementation (1 mark)
- **Attendance**: Week 1-2 (2 marks)

Week 3: Advanced LP & Real Applications (4 hours)

- **Theory (1h)**: Sensitivity analysis, duality
- **Lab (3h)**: Transportation problems, case studies
- **Lab Work**: Transportation problem solution (1 mark)
- **Micro-Project 1 Due**: Submission (5 marks)

Week 4: Nonlinear Optimization Foundations (4 hours)

- **Theory (2h)**: Unconstrained optimization, Golden Section
- **Lab (2h)**: `SciPy` optimization, function minimization
- **Lab Work**: Nonlinear solver implementation (1 mark)
- **Attendance**: Week 3-4 (2 marks)

Phase 2: Constrained & Combinatorial Methods (Weeks 5-8)

Week 5: Constrained Optimization (4 hours)

- **Theory (2h)**: KKT conditions, constraint handling
- **Lab (2h)**: Constrained NLP implementation
- **Lab Work**: KKT condition implementation (1 mark)
- **Micro-Project 2 Launch**: Nonlinear cost optimization
- **Lab Exam 1**: Basic LP/NLP implementation (5 marks)

Week 6: Project Planning & Heuristics (4 hours)

- **Theory (1h)**: CPM/PERT fundamentals
- **Lab (3h)**: Project scheduling, greedy algorithms
- **Internal Exam 1**: Theory assessment (10 marks)
- **Lab Work**: Project scheduling implementation (1 mark)

Week 7: Graph Algorithms I (4 hours)

- **Theory (1h)**: Graph theory, shortest path concepts
- **Lab (3h)**: NetworkX implementation, Dijkstra's algorithm
- **Lab Work**: Shortest path implementation (1 mark)
- **Micro-Project 2 Due**: Submission (5 marks)
- **Attendance**: Week 5-7 (2 marks)

Week 8: Graph Algorithms II (4 hours)

- **Theory (1h)**: MST, network flows, TSP overview
- **Lab (3h)**: Advanced graph algorithms
- **Lab Work**: MST implementation (1 mark)
- **Micro-Project 3 Launch**: Routing optimization

Phase 3: Advanced Applications & Integration (Weeks 9-12)

Week 9: Machine Learning Optimization I (4 hours)

- **Theory (2h)**: Gradient Descent, SGD, optimization in ML
- **Lab (2h)**: Basic GD implementation
- **Lab Work**: Gradient descent implementation (1 mark)
- **Attendance**: Week 8-9 (2 marks)

Week 10: Machine Learning Optimization II (4 hours)

- **Theory (1h)**: Advanced optimizers, neural networks
- **Lab (3h)**: TensorFlow/PyTorch optimization
- **Lab Work**: Advanced optimizer implementation (1 mark)
- **Lab Exam 2**: Graph and ML optimization (5 marks)

Week 11: Integrated Applications (4 hours)

- **Workshop (4h)**: Comprehensive system implementation
- **Lab Work**: Integrated system development (2 marks)
- **Micro-Project 3 Due**: Submission (5 marks)

Week 12: Review & Final Assessment (4 hours)

- **Internal Exam 2**: Theory assessment (10 marks)
- **Course Review**: Comprehensive concepts revision
- **Lab Work**: Final implementation polish (1 mark)
- **Attendance**: Week 10-12 (2 marks)

Thematic Project: Campus City Supply Chain

Micro-Projects (15 Marks Total)

Micro-Project 1: Basic LP Implementation (5 marks)

- **Timeline**: Week 1-3
- **Scope**: 6 facilities, 3 warehouses, linear costs
- **Assessment**: Model correctness (2), Code quality (2), Documentation (1)

Micro-Project 2: Nonlinear Optimization (5 marks)

- **Timeline**: Week 4-7
- **Scope**: Enhanced cost models, KKT conditions
- **Assessment**: Algorithm implementation (2), Analysis (2), Validation (1)

Micro-Project 3: Graph & Network Optimization (5 marks)

- **Timeline**: Week 8-11
- **Scope**: Routing, shortest paths, resource allocation
- **Assessment**: System design (2), Performance (2), Documentation (1)

Detailed Mark Distribution

Day-to-Day Lab Work (15 Marks)

- **Weekly Implementation Tasks**: 12 marks (1 mark × 12 weeks)
- **Integrated System Development**: 3 marks (Week 11)

Attendance (10 Marks)

- **Weekly Attendance**: 2 marks per 3-week block
- **Full Attendance Bonus**: 2 marks for 100% attendance

Lab Exams (10 Marks)

- **Lab Exam 1** (Week 5): Basic LP/NLP implementation (5 marks)
- **Lab Exam 2** (Week 10): Graph and ML optimization (5 marks)

Internal Exams (20 Marks)

- **Internal Exam 1** (Week 6): Modules 1-2 theory (10 marks)
- **Internal Exam 2** (Week 11): Modules 3-4 theory (10 marks)

Learning Outcomes Mapping

Theory Outcomes (Internal Exams + External)

- Formulate optimization problems mathematically
- Understand algorithm properties and convergence
- Analyze problem structures and solution methods

Practical Outcomes (Lab Work + Projects)

- Implement optimization algorithms in Python
- Develop end-to-end optimization systems
- Validate and analyze optimization results
- Create professional documentation and visualizations

Grading Rubrics

Micro-Projects (5 marks each)

- **Excellent (5)**: Flawless implementation with advanced features
- **Very Good (4)**: Correct implementation with good documentation
- **Good (3)**: Basic functionality with minor issues
- **Satisfactory (2)**: Meets minimum requirements
- **Poor (1)**: Significant functionality missing

Lab Work (Weekly 1 mark)

- **Complete (1)**: Task fully implemented and demonstrated
- **Partial (0.5)**: Basic implementation with issues
- **Incomplete (0)**: Task not attempted or completely non-functional

Lab Exams (5 marks each)

- **Algorithm Implementation**: 2 marks
- **Problem Solving**: 2 marks
- **Code Quality**: 1 mark

Success Strategy

Maximizing Internal Marks

- **Consistent Attendance**: 10 marks easily achievable
- **Regular Lab Work**: 15 marks through weekly completion
- **Quality Projects**: 15 marks with careful implementation
- **Lab Exam Preparation**: 10 marks with practice
- **Internal Exam Focus**: 20 marks through concept mastery

External Exam Preparation (80 Marks)

- Comprehensive theory coverage from all modules
- Problem-solving practice with various optimization types
- Mathematical formulation skills
- Algorithm analysis and comparison

Weekly Preparation Guide

Before Each Week

- Review weekly objectives and deliverables
- Prepare development environment
- Read theoretical concepts in advance

During Each Week

- Attend all sessions (critical for attendance marks)
- Complete lab work during sessions
- Start micro-projects early
- Seek clarification immediately

After Each Week

- Submit all lab work promptly
- Review concepts for internal exams
- Prepare for upcoming assessments
- Maintain code repository

This assessment-focused syllabus ensures students can maximize their 70 internal marks through consistent performance while preparing comprehensively for the 80-mark external examination. The structured approach balances theoretical understanding with practical implementation skills.

1 Introduction to Computational Optimization and Applications

```
Welcome to Computational Optimization & Applications
Where Mathematics Meets Computational Intelligence
Bridging Theory and Practice in the AI Revolution
```

1.1 Why Optimization Matters Now More Than Ever

The exponential growth in data complexity and computational requirements has transformed optimization from a theoretical discipline to an essential toolkit for every computer scientist and data professional. Consider these real-world contexts:

- **AI Systems**: Training neural networks is essentially an optimization process (Gradient Descent)
- **Operations Research**: Logistics, scheduling, and resource allocation drive billion-dollar efficiencies

- **Data Science**: Model selection, hyperparameter tuning, and feature engineering are optimization problems
- **Autonomous Systems**: Path planning, control systems, and decision-making rely on optimization algorithms
- **Quantum Computing**: Many quantum algorithms are designed to solve optimization problems more efficiently

1.2 Programme Objectives & Learning Outcomes

1.2.1 Core Educational Mission

This minor programme is designed to bridge the critical gap between theoretical optimization mathematics and practical computational implementation. Upon successful completion, you will be able to:

Domain	Learning Outcomes
Theoretical Foundation	• Formulate real-world problems as mathematical optimization models • Understand optimality conditions and convergence properties • Analyze problem structures to select appropriate solution methods
Computational Skills	• Implement classical and modern optimization algorithms in Python • Utilize industry-standard optimization libraries and frameworks • Develop end-to-end optimization pipelines for practical applications
AI/ML Integration	• Understand optimization's role in training machine learning models • Implement gradient-based methods for neural network optimization • Apply optimization to hyperparameter tuning and model selection
Problem-Solving	• Design optimization solutions for complex, multi-objective problems • Evaluate solution quality and algorithm performance • Communicate optimization insights to technical and non-technical stakeholders

1.2.2 The "Smart City Logistics" Experiential Thread

Throughout this course, we'll employ a *continuous practical thread*: optimizing logistics and operations for a smart city ecosystem. This narrative provides:

- **Real-world context** for theoretical concepts
- **Progressive complexity** as we advance through modules
- **Portfolio-building** implementation experience
- **Industry-relevant** problem-solving skills

Smart City Optimization Journey:

- **Module 1**: Warehouse Location (Linear Programming)
- **Module 2**: Fuel Cost Optimization (Nonlinear Programming)
- **Module 3**: Delivery Routing (Graph Algorithms)
- **Module 4**: Dynamic Scheduling (Heuristic Methods)
- **Module 5**: Demand Prediction (ML Integration)

1.3 Course Roadmap & Syllabus Integration

1.3.1 Module Progression: From Foundations to Frontiers

Our journey through computational optimization is strategically sequenced to build from fundamental principles to advanced applications:

1.3.1.1 Foundation Phase: Mathematical Underpinnings

- **Module I**: Linear Programming & Formulation Skills
- **Module II**: Nonlinear Optimization & Constraint Handling

1.3.1.2 Advanced Phase: Algorithmic Thinking

- **Module III**: Project Planning & Heuristic Methods
- **Module IV**: Combinatorial & Graph Optimization

1.3.1.3 Integration Phase: AI/ML Applications

- **Module V**: Gradient Methods & Machine Learning Optimization

1.3.2 Assessment Strategy: Theory Meets Practice

To ensure comprehensive understanding and skill development, assessment integrates both theoretical knowledge and practical implementation:

- **Series Examinations**: Test conceptual understanding and problem formulation
- **Practical Assignments/ Micro Project**: Evaluate implementation skills and computational thinking

- **Final Project**: Assess integrated problem-solving and solution design
- **Continuous Evaluation**: Monitor progress through micro-projects and code reviews

1.4 Optimization in the AI/ML Ecosystem

1.4.1 The Central Role in Machine Learning

Optimization isn't just adjacent to machine learning—**it is machine learning**. The entire process of training machine learning models revolves around optimization principles:

- **Loss Minimization**: Finding model parameters that minimize prediction error
- **Convergence Analysis**: Understanding when and how algorithms reach optimal solutions
- **Regularization**: Balancing model complexity with performance through constrained optimization
- **Hyperparameter Tuning**: Optimizing the optimization process itself

1.4.2 Emerging Trends & Future Directions

The field of optimization is rapidly evolving, driven by advances in:

- **Large-Scale Optimization**: Methods for billion-parameter models in deep learning
- **Automated Optimization**: AutoML and neural architecture search
- **Quantum Optimization**: Quantum annealing and hybrid quantum-classical algorithms

- **Federated Optimization**: Privacy-preserving distributed learning
- **Multi-Objective Optimization**: Pareto optimization for conflicting objectives
- **Explainable Optimization**: Interpretable and transparent optimization processes

1.5 Technical Ecosystem & Tools

1.5.1 Why Python for Optimization?

Python has emerged as the **lingua franca** for computational optimization due to:

- **Rich Ecosystem**: Comprehensive libraries for every optimization paradigm
- **AI/ML Integration**: Seamless connection with machine learning frameworks
- **Performance**: C/Fortran-backed numerical computing with Python simplicity
- **Community**: Vibrant ecosystem with continuous algorithm development
- **Industry Adoption**: Widely used in both academia and industry

1.5.2 Core Toolchain

Throughout this course, we'll work with industry-standard tools:

Our Computational Optimization Stack:

Category	Tools
Numerical Computing	NumPy, SciPy
Linear Programming	PuLP, CVXPY
Machine Learning	Scikit-learn, TensorFlow, PyTorch
Graph Algorithms	NetworkX
Visualization	Matplotlib, Plotly, Seaborn
Development	Jupyter, VSCode, Git

1.6 Getting Started: Your Learning Journey

1.6.1 Prerequisites & Preparation

To succeed in this course, you should have:

- **Programming Fundamentals**: Basic Python proficiency
- **Mathematical Background**: Linear algebra and calculus foundations

- **Computational Mindset**: Willingness to experiment and debug
- **Problem-Solving Attitude**: Persistence through challenging concepts
- **Curiosity and Creativity**: Interest in exploring multiple solution approaches

1.6.2 How to Maximize Your Learning

1. **Engage Actively**: Don't just read—implement every concept in code
2. **Think Critically**: Question why certain methods work better for specific problems
3. **Experiment Freely**: Modify parameters, break code, and learn from failures
4. **Connect Concepts**: Relate theoretical principles to practical implementations
5. **Build Portfolio**: Document your work for future career opportunities
6. **Collaborate Effectively**: Learn from peers through code reviews and discussions
7. **Stay Updated**: Follow recent developments in optimization research

1.6.3 Course Structure and Expectations

This course is designed as a **blended learning experience** combining:

- **Theoretical Foundations**: Mathematical principles and algorithm concepts
- **Practical Implementation**: Hands-on coding exercises and projects
- **Real-World Applications**: Industry-relevant case studies and problems
- **Assessment and Feedback**: Continuous evaluation and improvement

1.7 Welcome to the Journey

You are beginning a journey into one of the most fundamental and powerful domains of computer science and artificial intelligence. The skills you develop here will serve as a foundation for advanced work in machine learning, operations research, data science, and algorithmic design.

As we progress through the modules, remember that each concept builds toward a comprehensive understanding of how to make computers not just compute, but **optimize**—transforming them from calculators into intelligent decision-makers.

The journey through computational optimization is challenging but immensely rewarding. You'll gain not just technical skills, but a new way of thinking about problem-solving that will serve you throughout your career in technology.

> **"Optimization is the science of better. In a world of limited resources and unlimited wants, optimization provides the mathematical foundation for making the best possible decisions."**

---

2 Module 1: Basics of Optimization and Linear Programming

2.1 Module Overview

2.1.1 Learning Objectives

- Understand fundamental optimization concepts and problem classification
- Formulate real-world problems as Linear Programming models
- Solve LP problems using graphical and computational methods
- Implement LP solutions in Python for practical applications
- Apply optimization thinking to facility location problems

2.1.2 Smart City Context: Warehouse Location Challenge

In our Smart City Logistics project, we face a critical business decision: *where to locate new distribution warehouses* to minimize transportation costs while serving all city facilities efficiently. This module provides the mathematical foundation to solve this strategic problem.

2.2 Foundations of Optimization

2.2.1 What is Optimization?

Optimization is the science of finding the *best possible solution* from all feasible alternatives under given constraints. In computational terms, it involves:

- **Decision Variables**: Quantities we can control (e.g., warehouse locations, shipment quantities)
- **Objective Function**: What we want to maximize or minimize (e.g., total transportation cost)
- **Constraints**: Limitations and requirements (e.g., budget, capacity, demand)

2.2.2 Optimization Problem Classification

Problem Type	Characteristics	Smart City Example
Linear Programming	Linear objective and constraints	Warehouse location with fixed costs
Nonlinear Programming	Nonlinear relationships	Fuel costs that increase with distance
Constrained Optimization	With limitations	Budget constraints on construction
Unconstrained Optimization	No limitations	Theoretical ideal locations
Discrete Optimization	Integer decisions	Yes/no decisions for locations
Continuous Optimization	Real-valued decisions	Precise coordinates for facilities

2.2.3 Real-World Applications in CSE

- **Resource Allocation**: CPU time, memory allocation in operating systems
- **Network Optimization**: Internet routing, data flow optimization
- **Machine Learning**: Model training via loss function minimization
- **Database Systems**: Query optimization and indexing
- **Computer Graphics**: Rendering optimization and path tracing

2.3 Basic Python for Optimization

2.3.1 Essential Python Libraries Setup

```
# Core optimization stack installation
# pip install numpy scipy matplotlib pulp pandas jupyter
```

Key `Python` Libraries Required for Computational Part

- `NumPy`: Numerical computing and array operations
- `SciPy`: Scientific computing and optimization algorithms
- `Matplotlib`: Data visualization and result plotting

- `PuLP`: Linear programming interface
- `Pandas`: Data manipulation and analysis

2.3.2 Python Fundamentals for Optimization

```python
# Essential operations we'll use frequently
import numpy as np
import matplotlib.pyplot as plt

# Array operations for constraint matrices
coefficients = np.array([[2, 1], [1, 3], [4, 2]])

# Function definitions for objectives
def transportation_cost(warehouses, facilities):
    return np.sum(warehouses * facilities)

# Data handling for problem parameters
demand_data = {'Hospital': 50, 'School': 30, 'Mall': 80}
```

2.4 Linear Programming Fundamentals

2.4.1 Mathematical Definition of Linear Programming

A Linear Programming (LP) problem can be formally defined as:

Objective Function:

$$\text{Optimize } Z = c_1x_1 + c_2x_2 + \cdots + c_nx_n$$

Subject to Constraints:

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n &\leq b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n &\leq b_2 \\ &\vdots \\ a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n &\leq b_m \end{aligned}$$

Non-negativity Conditions: