Republic of Iraq
Ministry of Higher Education & Scientific Research
University of Babylon
College of Education for Pure Sciences
Department of Mathematics

Employing Control Mathematical Models of Numerical Differential Optimization

A Dissertation

Submitted to the Council of College of Education for Pure Sciences University of Babylon in Partial Fulfillment of the Requirements for the Degree of Doctor of Philosophy in Education - Mathematics.

By

Mazin Kareem Kadhim Marjan

Supervised by

Assist. Prof. Dr. Ahmed Sabah Al-Jilawi

2023 A.D. **1444 A.H.**


قالُوا سُبْحَانَكَ
لاَ عِلْمَ لَنَا
إلاَّ مَا عَلَّمْتَنَا
إنَّكَ أنتَ
الْعَلِيمُ الْحَكِيمُ
صدق الله العظيم
سورة البقرة الآية (٣٢)

The Supervisor Certificate

I certify that the dissertation entitled " Employing Control Mathematical Models of Numerical Differential Optimization " by " Mazin Kareem Kadhim Marjan " has been prepared under my supervision in Babylon University - College of Education for Pure Sciences as a partial for requirement the Degree of Doctor of Philosophy in Education - Mathematics.

Signature:

Name: Dr. Ahmed Sabah Al-Jilawi

Title: Assistant Professor

Date: / / 2023

According to the available recommendation, I forward this dissertation for debate by the examining committee.

Signature:

Name: Dr. Azal Jaafer Musa

Head of Mathematics Department

Title: Assistant Professor

Date: / / 2023

Scientific Supervisor's Certification

I certify that I have read the scientific content of this dissertation " Employing Control Mathematical Models of Numerical Differential Optimization " and I have approved this dissertation is qualified for debate.

Signature:

Name: Dr. Bassem Abbas Hassan

Title: professor

Address: College of Computer Science and Mathematics, University of Mosul

Date: / / 2023

Scientific Supervisor's Certification

I certify that I have read the scientific content of this dissertation " Employing Control Mathematical Models of Numerical Differential Optimization " and I have approved this dissertation is qualified for debate.

Signature:

Name: Dr. Hazim Ghdhaib Kalt

Title: Assistant Professor

Address: College of Education for Pure Sciences , University of Karbala

Date: / / 2023

Linguistic Supervisor's Certification

I certify that I have read this dissertation entitled " Employing Control Mathematical Models of Numerical Differential Optimization " and corrected its grammatical mistakes; therefore, it has qualified for debate.

Signature:

Name: Dr. Maamoon S. Salih

Title: Assistant Professor

Address: English Department, University of Babylon

Date: / / 2023

Examining Committee Certificate

We certify that we have read this dissertation entitled " Employing Control Mathematical Models of Numerical Differential Optimization " as an examination committee, examined the student " Mazin Kareem Kadhim Marjan " in its contents and that in our opinion it meets the standard of a dissertation for the Degree of PhD in Education/ Mathematics.

Signature:
Name: Dr. Mushtak A.K Shiker Al-Jenabi
Title: Professor
Date : 17 / 5 / 2023
Chairman

Signature:
Name : Dr. Bushra Hussien Aliwi
Title: Assist. Professor
Date : 17 / 5 / 2023
Member

Signature:
Name: Dr. Janan Hamza Farhoud
Title: Assist. Professor
Date : 17 / 5 / 2023
Member

Signature:
Name : Dr. Kawthar Fawzi Hamza
Title: Assist. Professor
Date : 17 / 5 / 2023
Member

Signature:
Name: Dr. Basim Karim Kadhir
Title: Assist. Professor
Date : 17 / 5 / 2023
Member

Signature:
Name : Dr. Ahmed Sabah Al-Jilawi
Title: Assist. Professor
Date : 17 / 5 / 2023
Member / Supervisor

Approved by the Dean of College
Signature:
Name: Prof. Dr. Bahaa Hussein Salih Rabee
Date : 17 / 5 / 2023
Address: Dean of the College of Education for Pure Sciences

ACKNOWLEDGEMENTS

God Almighty in His Noble Book: (And whoever is grateful, he is thankful for himself). At the beginning of my speech, I must first thank God Almighty, who enabled me to reach this high scientific stage, and paved the way for me to be among you today. I extend my thanks and gratitude to Prof. Dr. Ahmed Sabah Al-Jilawi, who was and still is a true brother and friend, when I remember all that you have done for me in order to reach what I have reached on this day, my tongue is unable to say anything, expressions of thanks are few, and words of praise are not You can fulfill your right, you are everything at all times.

To all my teachers.

To my family.

To my loyal friends.

DEDICATION

To the soul of my dear mother and to the soul of my dear father who never failed me in anything, I dedicate this work to my beloved wife and children who have endured so much for me.

Mazin Kareem Kadhim Marjan

CONTENTS

LIST OF FIGURES

LIST OF TABLES

Table 1: **Symbols List**

$f(x)$	Objective Function
$Minf(x)$	The Minimum of Objective Function
$Maxf(x)$	The Maximum of Objective Function
$g(x)$	The Inequality Constraint
$h(x)$	The Equality Constraint
C	The Convex Set
$\mathcal{L}$	The Lagrangian Function
$\mathbb{R}$	Real Numbers
$\nabla f(x)$	The Gradient of f
$\nabla^2 f(x)$	The Hessian of f
x^*	The Minimizer (Local, Global)
x^{k+1}	The Next Iterate
x^k	The Current Iterate
α_k	The Step Length
$\mathbb{R}^n$	The collection of order lists of n real numbers

Table 2: **Abbreviations List**

$DOPs$	Discrete Optimization Problems
LP	Linear Programming Problems
TSP	Traveling Salesman Problem
MST	Minimum Spanning Tree
MPC	Model Predictive Control
QP	Quadratic Programming
SDP	Semidefinite Programming
HJB	Hamilton-Jacobi-Bellman
IT	Information Technology
PEN	Penalty
$fcalls$	function calls
AUG	Augmented
NLP	Non-Linear Programming
$MILP$	Mixed-Integer Linear Programming
KKT	Karush-Kuhn-Tucker Conditions
ABC	Programming Language
MV/LV	Transformer Electrical
NP	Non Deterministic Polynomial Time
GP	General Problem
CEi	Energy generation cost for producer i
CRi	Reserve capacity price for producer
DEt	Energy demand at time t
DRt	Reserve capacity requirement at i
TC	total operation cost

ABSTRACT

Our goal in this dissertation is to find a new technique for optimal control that require level control complexity in terms of saving time, speed and accuracy. In fact, a group of mathematical systems, all of which depend on differential equations, have been studied, as these systems represent a close link with optimal control. Most of the problems of these systems lie in the control of finding the optimal solution. For this, the numerical optimization method is adopted as a means to find solutions to the problems of these mathematical systems by finding an approximate solution. In the first model, we deal with movement the least sliding friction block with the least control effort can be reversed by restriction the amount of displacement the mass is allowed to move. The second embodiment the control of a dynamic system over a period of time is optimized to be an objective function. In this regard, an effective algorithm has been applied about the dynamic system, which is a spacecraft with controls similar to rocket thrusters, the goal of which to reach the moon with the least amount of spent fuel. The third and fourth model are subject to differential constraints. Many areas including production transportation and energy systems seek to reduce the total time required for a task. The optimum control problem seeks to reduce the final time. And the objective of the optimizer is to achieve the desired end state in the least amount of time possible. For network systems, the fifth model problem is a representation of the very common following load scenario.

CHAPTER 1

INTRODUCTION

Optimization problems that aim to maximize profit, minimize costs, or minimize the usage of resources employ objective functions. Problems with Optimization are finding the best way to do something is a key part of many important applied problems. Often, this means determining whether a function has a maximum or minimum value, like the least amount of time it takes to make a certain trip, the least amount of money it costs to do a task, the most power a device can make, and so on. Many of these problems can be solved by finding the right function and then using calculus to find the maximum or minimum value needed [28]. The main objective is to optimize a large number of crucial combinatorial problems. Problems involving optimization can be thought of as generalizations of decision problems in which the solutions are assessed by an objective function in addition to the decision criteria. Candidate solutions and solutions are commonly used to define the objective function, and the objective function value of a specific candidate solution (or solution) is sometimes referred to as its quality. Any combinatorial optimization problem can be framed as a minimization or a maximization question, depending on whether the specified objective function needs to be minimized or maximized [20]. Often, one of the two formulations is more intuitive, yet algorithmic problems of minimization and maximizing are treated a like. In addition,

many decision-making algorithms can be applied to related optimization problems in a relatively straightforward manner [86]. The simple extensions of algorithms that work well on particular decision problems fall short. The evolutionary community, is quite familiar with numerical optimization problems. Many of these technologies, on the other hand, are designed to address real-world problem that go beyond simple restrictions (as well as other traditional optimization methods). These kinds of difficulties can be solved inside the boundaries of a territory that is a reasonable distance from the rest of the world. Therefore, it is essential to research some of the problem-specific aspects that can successfully explore these constraints [75]. Numerous optimization models and algorithms depend on systems of linear and nonlinear equations. The solution of systems of linear equations is fundamental to both the simplex method for linear programming and Newton's method for nonlinear optimization, which is used to find dual variables (Lagrange multipliers) in the two settings. The nonlinear control system is a form of control that violates the homogeneity principle [44]. In the real world, there are no truly linear control systems (linear control systems only exist in theory). It may take the form of a set of differential equations, a set of partial differential equations, or a set of discrete equations [69]. In the calculus of variations, optimal control theory has been formulated as an extension. Mathematicians have laid the theoretical groundwork for optimal control in numerous scientific domains, including mathematics, engineering, biomedical, and management sciences. One of the most important developments in optimum control is the maximum principle, developed in the late 1950 by Pontryagin and his colleagues [65]. One of the most important principles of optimal control is that it may be applied to any problem in calculus of variations (and gives equivalent results, as one would expect). This extension can be understood most clearly by looking at the usual problem of the calculus of variations, which is to select a function that is continuously differentiable function $y(t),\ t_1 \leq t \leq t_2$, to solve the following problem :

$$\begin{cases} \textit{Minimize :} & \displaystyle\int_{t_1}^{t_2} \eta(t, y(t), \dot{y}(t))\ dt \\ \textit{Subject to :} & y(t_1) = y_1 \end{cases} \tag{1.1}$$

In fact, the aforementioned issue can be conveniently recast as the search for a continuously differentiable function $u(t),\ t_1 \leq t \leq t_2$.

$$\begin{cases} \textit{Minimize :} & \varphi(u) = \displaystyle\int_{t_1}^{t_2} \eta(t, y(t), u(t))\ dt \\ \textit{Subject to :} & \dot{y}(t) = u(t); \quad y(t_1) = y_1 \end{cases} \tag{1.2}$$

Control optimization refers to this type of issue. State (or phase) and control variables are the two kinds of variables in optimal control problems [70]. A system of differential equations dictates the evolution of the former. Many optimum control problems are non-classical because the control as well as the state variables are subject to constraints, making many problems in optimal control non-classical. For example: 'Determine the optimal control signals that will meet the physical restrictions and, at the same time, minimize (or maximize) some performance objective.

1.1 Related Works

There are nine primary areas to our study endeavor that will all help each other in order to reach our goal:

1. Optimization History
2. Linear and Nonlinear Optimization Problems
3. Constrained and Unconstrained Optimization Problems
4. Differential Techniques
5. Dynamic Optimization Fields
6. Optimal Control Systems
7. Optimization Methods for Energy System
8. Python Software in Optimization
9. Optimization Algorithm Design

1.1.1 Optimization History

Many theoretical and computational advances in the field of nonlinear optimization have occurred since 1947. But it is important to remember that nonlinear optimization was first studied in the 1600s. This subject is actually studied by both Fermat (1638) and Newton (1670) :

$$\min_{u} \quad f(u) \qquad wher\ u \in \mathbb{R}$$

The classical optimality condition is developed by both Newton and others : $\frac{df(u)}{du} = 0$. This is generalized by Euler (1755) to multivariate nonlinear optimization :

$$\min \quad f(u_1, \ldots, u_n)$$

with the optimality condition : $(\nabla f(u) = 0)$.

Euler and Lagrange worked on infinite-dimensional problems in the late 1700s and developed the calculus of variations [5]. After that, until Dantzig's development of contemporary linear optimization in 1947, optimization received very little attention. General optimality requirements for nonlinear constrained optimization are proposed in the late 1940s and early 1950s by Kuhn and Tucker (preceded by Karush). Many decision-making problems in management and industrial operations at the time are being solved using nonlinear optimization techniques. Larger-scale nonlinear optimization was developed in the 1960s as computers became more powerful. Internal point approach for linear optimization pioneered by Karmarkar in 1984 and expanded to nonlinear convex optimization by Nemirovskii in 1994 opened up a new field of study in optimization [49]. Since 2000, "huge-scale" nonlinear optimization, which considers problems involving hundreds or even millions of decision variables and constraints, has gained a lot of attention and advancement. The mathematical optimization can be defined by the following diagram see figure (1.1) :

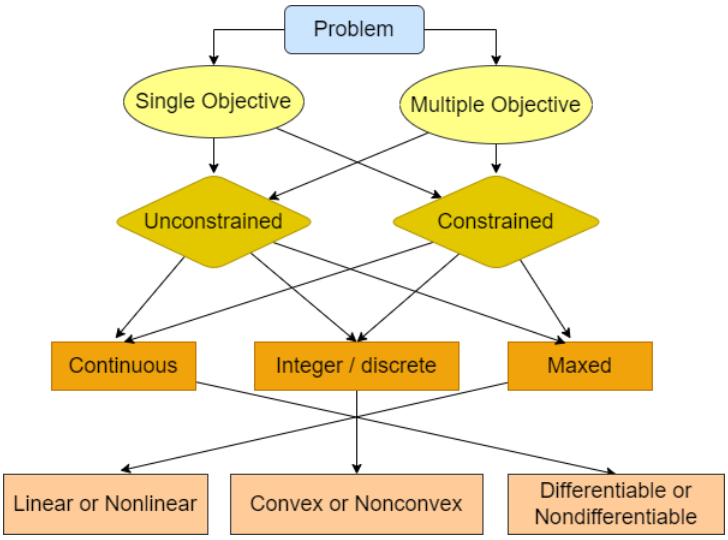


Figure 1.1: Optimization Modelling

1.1.2 Linear and Nonlinear Optimization Problems

In 1827, Fourier published a method for solving systems of linear inequalities that would later be known as the Fourier-Motzkin elimination method. In 1939, Soviet mathematician and economist Leonid Kantorovich formulated the general linear programming problem as a linear programming problem. Not only did he find a solution, but he invented one as well. During World War II, he devised this method of budgeting and revenue generation to reduce military expenditures and increase enemy losses. The Dutch-American economist T. C. Koopmans did similar work to that of Kantorovich around the same time [19]. In 1975, the Nobel Prize in Economics was shared by Kantorovich and Koopmans. Another person who used linear programs to solve transportation problems and came up with a solution very similar to the simplex method is Frank Lauren Hitchcock in 1941. It is impossible for Hitchcock to win the Nobel Prize now that he has passed away (1957). Significant theoretical and computational progress in the field of nonlinear optimization can be traced back to 1947. However, keep in mind that research into nonlinear optimization dates back to at least the 1600. It is investigated by Fermat (1638) and Newton (1670).

$$\min_x f(x) \; where \;\; x \in R \; , \tag{1.3}$$

and Newton developed the classical optimality condition: $\dfrac{df(x)}{dx} = 0$. This is generalized by Euler (1755) to multivariate nonlinear optimization:

$$\min \quad f(x_1, \ldots, x_n) \tag{1.4}$$

with the optimality condition: $\nabla f(x) = 0$

Both Euler and Lagrange worked on problems with infinite dimensions and created the calculus of variations in the late 1700. After that, not much was done to study optimization until 1947, when Dantzig came up with modern linear optimization. In the

late 1940 and early 1950, Kuhn and Tucker (Karush came before them) came up with general conditions for optimality in nonlinear constrained optimization [26]. At that time, nonlinear optimization is used to solve problems in chemical engineering, in portfolio management, and in a wide range of management and industrial operations decision problems. In the 1960, larger-scale nonlinear optimization is created at the same time that computers got more powerful. In 1984, Karmarkar's interior-point method for linear optimization opened up new ways to do optimization. In 1994, Nesterov and Nemirovskii extended this method to nonlinear convex optimization. In the 1990. Since 2000, "huge-scale" nonlinear optimization, which looks at problems with thousands or even millions of decision variables and constraints, has been getting a lot of attention and making progress. see figure (1.2):

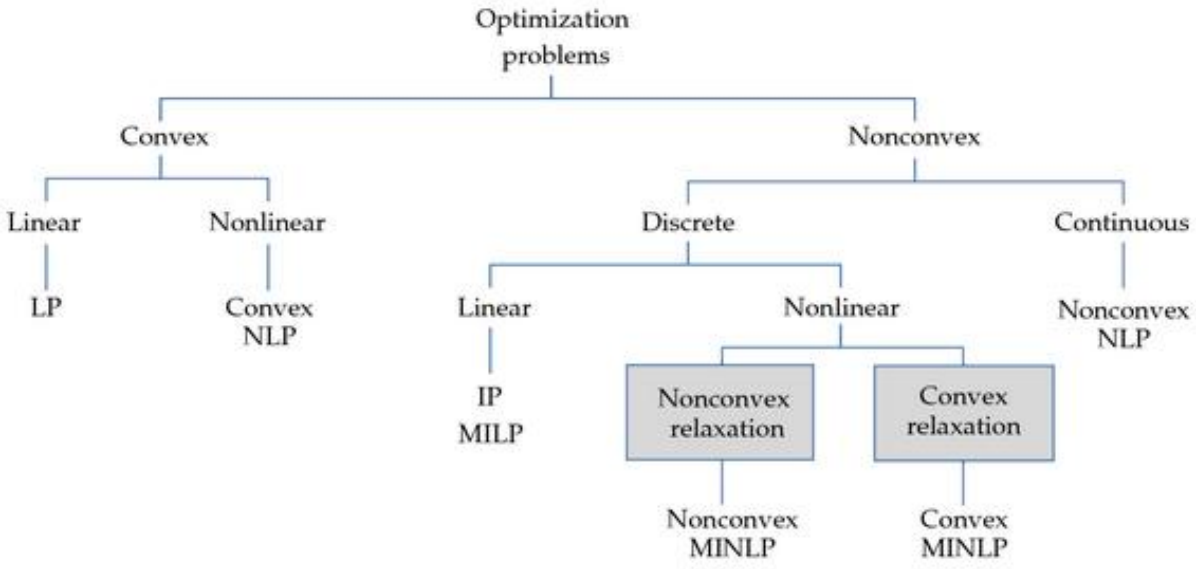


Figure 1.2: Linear and Nonlinear Optimization

1.1.3 Constrained and Unconstrained Optimization Problems

To say that a choice variable is unconstrained is to say that there are no limitations on what the choice variable can be. When a choice variable is constrained, it can only take on a limited range of values. The fact that we are only concerned with positive values for variables like output and pricing means that practically all economic problems are confined in this way. Due to the fact that this constraint is so prevalent, the tend to overlook it and consider constrained issues to be ones that include limitations in addition to non-negativity [67]. A good example of this is the budgetary restraint faced by the average consumer the amount of items you can purchase based on your income and price range is constrained if your income is less than or equal to your expenditure. In other words, we don't want you to conceive of limited problems that just limit the choice variable to non-negative numbers. An unconstrained optimization problem is one without any constraints. Constraints can be added to an unconstrained optimization problem with appropriate penalties in order to improve machine learning and deep learning algorithms. Due to the lack of restrictions, it is simple to formulate unconstrained optimization problems. We'll look at an objective function in this article. $f : R \to R$ that we wish to minimize. We express this as,

$$\min_{x} f(x)$$

If we are interested in the value of x that achieves the minimum, we shall write this as

$$x^* = \underset{x}{argmin}\ f(x)$$

where $\underset{x \in S}{argmin}\ f(x) = \{x \in S : f(x^*) \leq f(x)\ for\ all\ x^* \in S\}$

Example 1.1.1.

If $f : \mathbb{R} \to \mathbb{R}$ is given by $f(x) = (x+1)^2 + 3$, determine the argmin $f(x)$.

Solution :

$x = -1$ the argmin $f(-1) = 3$.

For the most part, in the field of machine learning, the more concerned with the second formulation. There are x signifies the model parameters that are being fit to the training data, and $f(x)$ denotes the loss that we want to minimize as a function of those parameters [48]. A constrained optimization problem consists of its objective function, its choice variables, and its constraint . It is a function of the decision variables that indicates the quantitative measure that the decision maker seeks to minimize or maximize. To increase the objective function's value, decisions are made using mathematical representations of the system's components, which are called "decision variables" Constraints refer to the limitations placed on the variables used in making decisions, frequently relating to available resources. It is necessary to translate the optimization problem from language into mathematical notation. Decision variables and parameters are used to specify the objective function and constraints in any optimization task. This also includes selecting whether the objective function should be maximized or minimized. Assuming the objective is to be maximized, the typical notation for any optimization issue is as follows:

$$\begin{cases} \textit{Minimize :} & z = f(x_1, x_2, \ldots, x_n, g_1, g_2, \ldots, g_k) \\ \textit{Subject to :} & c_j(x_1, x_2, \ldots, x_n, g_1, g_2, \ldots, g_k) \leq C_j \\ & for j = 1, 2, \ldots, m \end{cases} \tag{1.5}$$

where, $x_1, x_2, \ldots, x_n$ are the decision variables, $f(x_1, x_2, \ldots, x_n)$ is the objective function, and $c_j(x_1, x_2, \ldots, x_n, g_1, g_2, \ldots, _k) \leq C_j$ represent the constraints. Note that the constraints can include both inequality and equality constraints and that the objective function and the constraints also include parameters $g_1, g_2, \ldots, g_k$, which are not varied in the optimization problem. In the following is a simplified diagram of the structure of Constrained and Unconstrained ,see figure (1.3):

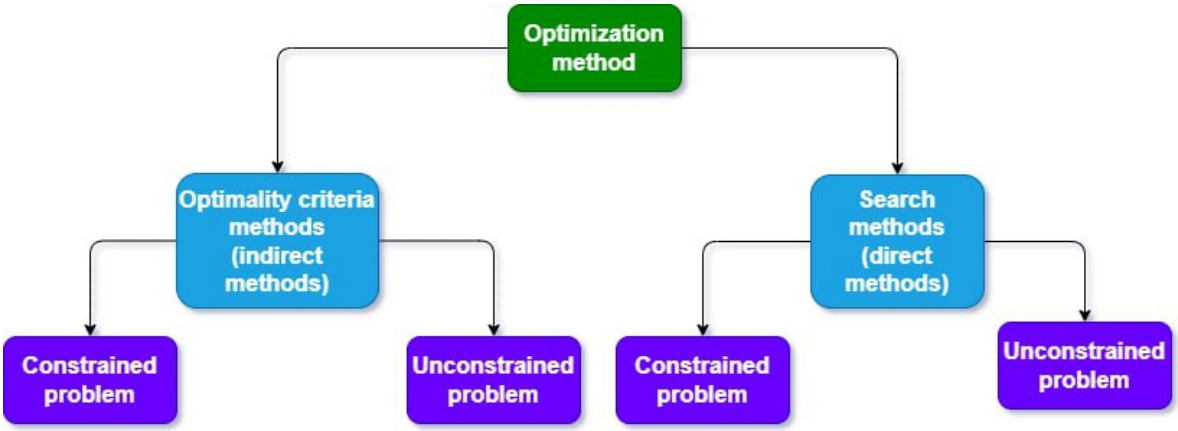


Figure 1.3: Constrained and Unconstrained

1.1.4 Differential Techniques

Newton and Leibniz developed calculus in the mid-seventeenth century, differential equations are born. After Newton published his Principia in 1642, the foundation of modern mathematics is laid. Calculus and his three main principles are taught in a book published in 1687. Mathematics has been made feasible because of these rules of motion [75]. Theory of differential equations has been influenced by the work of mathematicians. When working with specific physical models and gaining insights. In spite of this, after years of gradual growth, the subject is now clearly defined in mathematics. 'Differential Equation' is a name that says it all. It's a formula. That requires differentials of functions to be taken into consideration. Before getting into the groove of our main discussion. The first explain the term 'differential' of a function. If $f(x)$ be a function of the independent variable x and $f'(x)$ be its derivative,then the differential of $f(x)$, denoted by $df(x)$, is defined by the relation $df(x) = f(x).\triangle x$, where $\triangle x$ stands for the increment in x. In particular if $f(x) = x$, $f(x) = 1$ and so $dx = \triangle x$, signifying the truth that for an independent variable x, the differential dx of x and its increment are the same. Hence if $y = f(x)$, $dy = df(x) = f'(x) \triangle x$. For dependent variable $y = f(x)$, the differential dy is proportional to the differential dx, the proportionality constant being $f'(x) = \frac{dy}{dx}$

. This is one of the reasons why differential coefficients are used in elementary calculus. As a result, in a differential equation, one or more dependent variables' derivatives with respect to one or more independent variables may be included [43]. There are only one or more ordinary derivatives of functions in a differential equation when the number of independent variables equals unity, and these equations are referred to as ordinary differential equations, or ODEs. They are known as partial differential equations because they have more than one independent variable with respect to which partial derivatives of a single dependent variable are included in the equation.

ODE :

$$\frac{d^2y}{dx^2} - 8x\frac{dy}{dx} = 5, \tag{1.6}$$

$$(\frac{d^2x}{dt^2})^2 + w^2x^3 = 0. \tag{1.7}$$

PDE :

$$\frac{\partial u}{\partial t} = \lambda\frac{\partial^2 u}{\partial x^2}, \tag{1.8}$$

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = 4\pi\rho(x, y). \tag{1.9}$$

Up to the second order, derivatives of y with respect to x can be found in eq (1.6). In eq (1.7), the derivatives and the exponents of the independent variables are equal. As it turns out from the pde itself, u is a function of two independent variables, x and t, hence, λ is a constant in eq (1.8). In eq (1.9), x and y are considered independent variables, while u and ρ are considered dependent variables. The concept of differential equations is depicted in the following diagram. see figure (1.4):

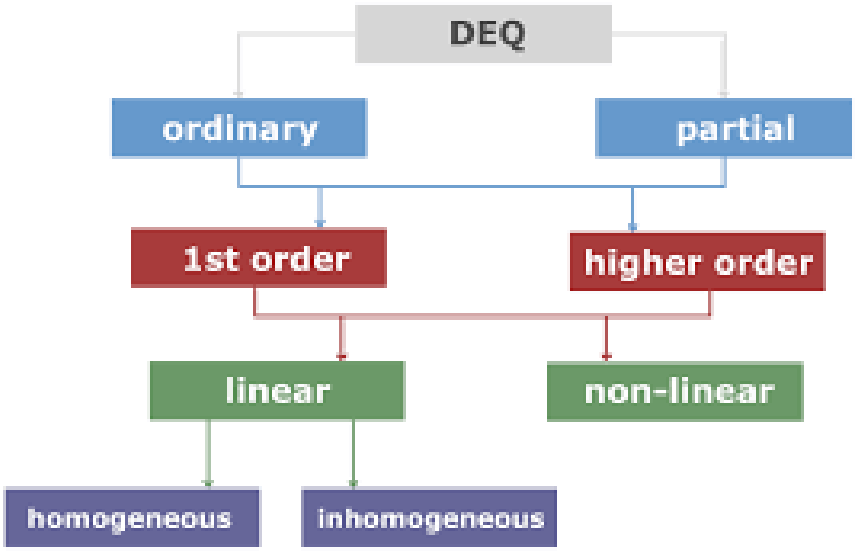


Figure 1.4: Differential Techniques

1.1.5 Dynamic Optimization Fields

Dynamic optimization models. They capture many relevant real-life problems: scheduling, route planning, solving optimal strategies for games, inventory control, investment problems, machine repair, text processing, dna sequence matching, stopping problems, airplane rocket flight path optimization, minimum time effort problems, optimal fishery management, saving consumption optimization, optimal feedback controllers for plants and regulator problems and so on [60].

A wide variety of dynamic test cases have been used in the literature. The following criteria can be used to categorize these dynamic test challenges into distinct subcategories:

1- Whether or not an algorithm's current answer has any effect on the problem's future behavior.

2- As to whether or not the resultant changes may be accurately predicted.

3- Whether or if the optimization algorithm is able to see the changes, and if so, how many detectors are needed to detect them. Optimizing Dynamically.

4- Constraints on the problem: Whether the problem has any constraints.

5- If the problem has one or more goals, the number of objectives.

6- Changes of the following kinds: Explanation in great detail of how the search space changes over time.

7- If the changes are cyclic recurrent in the search space.

8- Continuity: Whether the changes are regular or not.

9- Objective function parameters, variable domain, variable number, constraint, and other parameters can all change. The following is a list of the most typical properties of general-purpose dynamic benchmark problems in the literature [88].

10- The vast majority of dynamic test issues are non-time-related. In terms of time-linkage tests, there are only a few general-purpose and problem-specific variations.

11- In the continuous domain, the vast majority of dynamic test generators and challenges are either unconstrained or domain-limited.

12- Changes can be detected with as little as a few detectors in the default settings of most general-purpose dynamic test issues. In order to make changes more difficult to notice, generators can be programmed to produce various scenarios.

13- The objective functions are usually the ones that change.

14- The default parameters of many generators problems can change without warning. At the very least, the frequency and periodicity of change can be predetermined for some generators problems.

15- Circular recurrent changes characterize the majority of benchmark generators and challenges

16- In the majority of cases, benchmark generators and issues anticipate that changes occur after a set number of generations or fitness assessments.

17- The vast majority of generators and problems have a single goal. Most research on dynamic multi-objective issues is limited to a few papers. see figure (1.5):

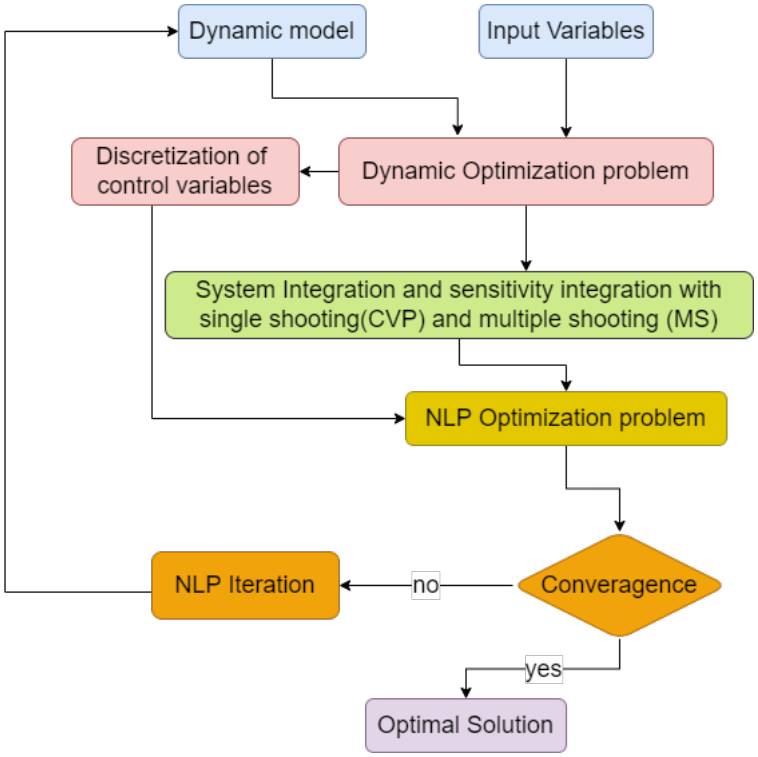


Figure 1.5: Dynamic Optimization Algorithm

1.1.6 Optimal Control Systems

Here is a long history of optimum control theory stemming from the calculus of variations, but it isn't until the early 1960s, ln aircraft applications when impressive advancements in trajectory prediction generated a boom of interest [39]. The "isoperimetric problem" the shape of the plane curve with a given length containing the maximum area is a well-known and solved geometrical optimization problem in ancient times. Yet Galileo is where the story really starts. He submitted two shape challenges to the world back in 1638: the design of a heavy chain suspended between two points (the catenary) and the design of an iron wire that could allow a bead to slide across its length in the shortest time possible (the brachistochrone). A few years later, in 1662, Fermat proposed the idea that light always seeks the shortest path through a series of optical medium. In a wide range of fields, optimum control has been applied and its effects studied. These include aerospace, process control, robotics, bioengineering and economics, finance and management science, among others. To now, digital computers have been able to solve only a limited number of complex control problems, with the introduction of digital computers, optimal control theory and methodologies may now be applied to a wider range of circumstances [16]. Depending on the performance index, the kind of time domain (continuous or discrete), the presence of different kinds of restrictions, and the variables that can be freely chosen, there are several types of optimum control issues. The following actions must be followed in order to formulate an optimal control problem :

1- Model of the system that needs to be regulated mathematically,

2- Specification of the performance metric,

3- State boundary conditions and restrictions that must be met by both states and controls, as well as their definitions,

4- Determine which variables can be used without constraint. see figure (1.6):

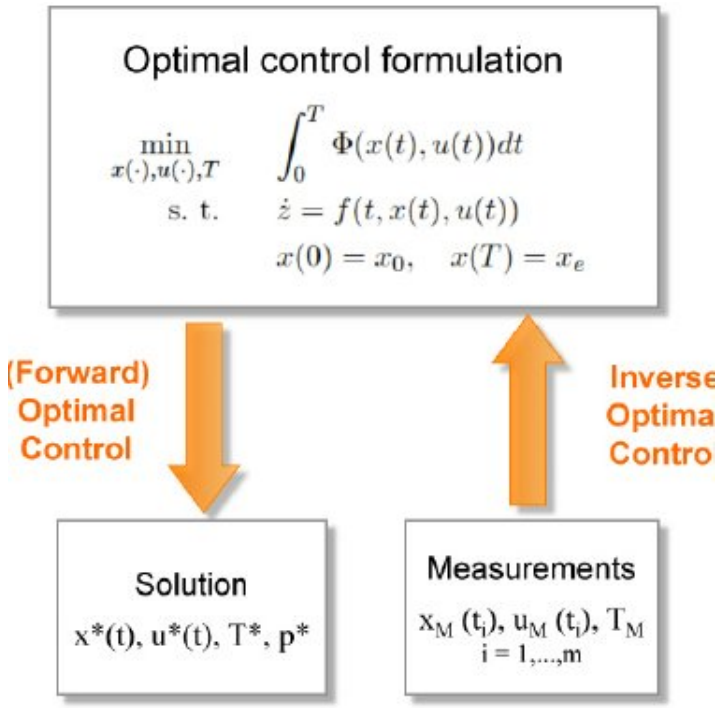


Figure 1.6: Formulation of Optimal Control Problem

1.1.7 Optimization Methods for Energy System

Scientists and engineers have been working for decades to improve and optimize energy systems. As a preliminary stage, a model of the system's behavior should be developed. The system's performance is then analyzed using the model(s) in use, and if changes need to be made, the system is given the tools it needs. As a result, the first stage in every improvement or optimization project is to construct a model that describes the energy system from all relevant perspectives. Various types of models, including thermal, economic, electric, environmental, thermoenvironmental, and empirical, are typically produced in this context. Power supply. The choice variables must be chosen so that they can be used independently. govern a specific set of goals. There must be only one decision variable for a two-phase flow, for example, if the

temperature and pressure in the system are dependent. As an alternative, parameters refer to the formulaic variables that are therefore, it can't be changed during optimization. Parameters that can be used to identify a problem physical qualities, fuel costs, and the heating value of a substance are all considered in the design of energy systems [89]. Thermal and climatic conditions. For example, different factors in governing equations. In the majority of optimization problems, the decision over all the genuine variables, variables cannot be swapped. They've perfected it. There should be no value that goes against the rules. Physical limitations are the most common source of restrictions. Engineered and regulating norms establish constraints or restrictions. As an illustration, consider the when performing optimization, it is impossible for the decision variables to have values that could result in there is a breach of the rules of thermodynamics. Some engineering may reveal additional constraints. limits imposed by local and governmental rules [78]. Examples include the temperature and pressure of an object. Technical or environmental constraints may limit the energy system's capabilities. local codes may limit the impact of the energy system. There are instances when it's necessary to make both lower and higher limits on decision variables exist. The underlying physics constraint expressions exist in the problem formulation by way of equalities and inequalities in mathematics. Governing bodies often place restrictions mathematical formulations of the problem include physical equations in the form of equalities. Inequalities are a common method of expression for engineering constraints. Constraints may be present in an optimization problem, or they may not. Unconstraint optimization problems are those in which there are no constraints. A constrained optimization problem is one that has a set of constraints. An unconstraint problem in engineering, especially in the optimization of energy systems, is extremely difficult to assume. Constraint optimization is a more difficult problem to solve using optimization methods than the unconstrained problem. see figure (1.7):

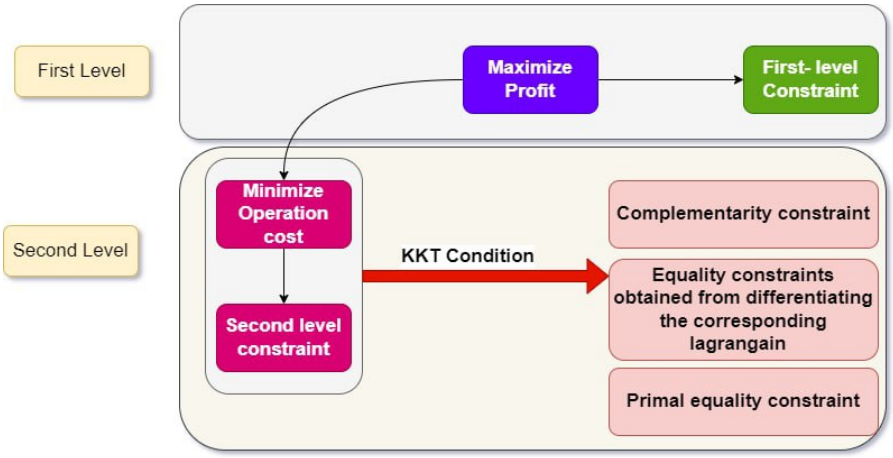


Figure 1.7: Energy Systems

1.1.8 Python Programming Language

Python is a high-level, all-purpose programming language. Its design philosophy places a strong emphasis on code readability through the usage of substantial indentation. [12] Python uses dynamic typing and trash collection. It supports a variety of programming paradigms, including structured (especially procedural), object-oriented, and functional programming. Due to its extensive standard library, it is frequently called a (batteries included) language. Python is a well-known programming language due to its simplicity, ease of use, open source licencing, accessibility, well-known community, strong support and help, and abundance of packages, tutorials, and sample programmes that make it simple for a newcomer to learn and code with Python. Python can be used to create a wide range of applications, including those for the web, desktop GUI programmes, science and mathematics programmes, machine learning programmes, and other types of applications. Python is a high-level programming language. As a replacement for the ABC programming language, Guido van Rossum started developing Python in the late 1980s. Python 0.9.0, a high-level, general-purpose programming language, is first released in 1991. [57]. see figure (1.8) :

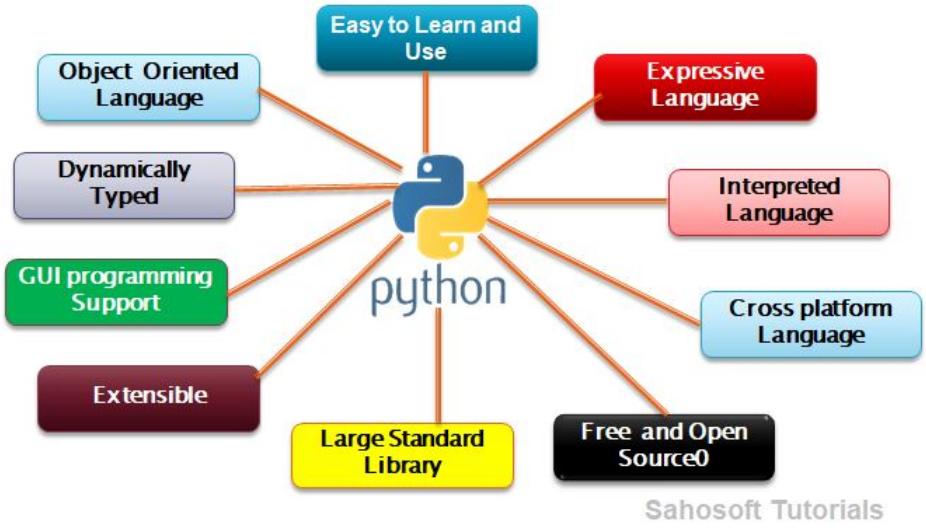


Figure 1.8: Python Programming Language

1.1.9 Optimization Algorithm Design

Algorithms are a set of step-by-step instructions for solving a problem. Algorithm design and analysis is crucial in the field of applied mathematics and information technology when developing algorithms to solve various problems. Despite the fact that algorithm design has been around for a while, it is always evolving and improving. Randomized algorithms and data structures, particularly those based on hashing, are becoming increasingly significant [50]. The complexity of the best algorithms known for fundamental problems such as finding optimal solution, and network flows has been decreased by recent advancements. What is an algorithm if the date the beginnings of contemporary algorithm creation and analysis to the 1970s? An algorithm is a step-by-step process for achieving a specified goal. task. Every sane computer program begins with an algorithm [82]. An algorithm must answer a general, well-specified issue in order to be interesting. In order to specify an algorithmic problem, one must provide

a list of all possible instances. If the run it in one of these environments, than need to work on and with the results you get. This An important distinction is between a problem and an individual instance of a problem. Among other things, sorting is an algorithmic problem.

Problem: Sorting

Input : A sequence of n keys $a_1, ..., a_n$

Output : The permutation (reordering) of the input sequence such that

$a_1^{'} \leq a_2^{'} \leq .... \leq a_{n-1}^{'} \leq a_n^{'}$.

should can sort names like "Mike," "Bob," "Sally," "Jill," and "Jan" into an array like "Mike Bob Sally Jill Jan." The first step in resolving a problem is to recognize that it is a general issue rather than a specific one. An algorithm is a technique that accepts any input and changes it into the desired output. Sorting can be accomplished using a plethora of different algorithms. It's possible to start with just one element and then slowly add the rest of the elements so that the list is maintained in a sorted order. On a specific instance (the letters of the phrase "insertionsort"), this algorithm's logical flow is shown in animation. see figure (1.9):

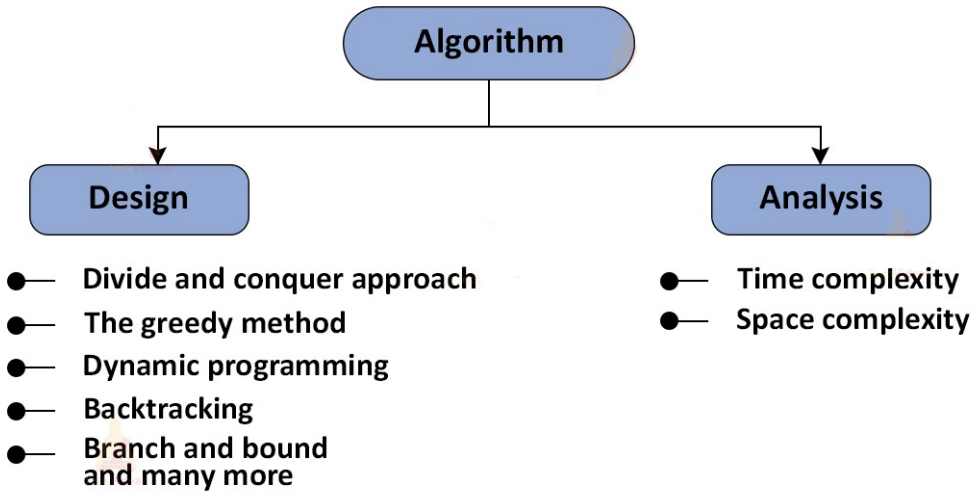


Figure 1.9: Algorithm Design

CHAPTER 2

MATHEMATICAL BACKGROUND

In this chapter, we provide definitions, notation, and necessary results of optimization problem. Also, we present some basic concepts and facts of real mathematical analysis.

2.1 Basic Concepts of Optimization Algorithms

The optimization Algorithms includes a discussion of the fundamental definitions and concepts that are relevant to our work.

2.1.1 Concepts and Their Relationships

Definition 2.1.1. [8] Optimization problem is a computational problem where the goal is to discover the best possible answer. In other words, finding a solution that falls inside the range of the goal function's minimum (or maximum) value.

Definition 2.1.2. [85] Optimal Solution is a solution to an optimization problem that minimizes (or maximizes) the objective function is referred to as a "optimal solution" . As can be seen in the accompanying figure and the approach taken to arrive at the best possible outcome see figure (2.1):

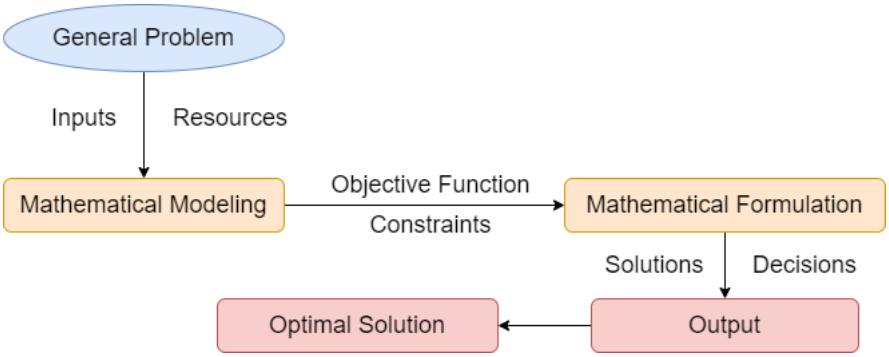


Figure 2.1: Optimal Solution

Definition 2.1.3. [51] The objective function is a tool for maximizing (or minimizing) a specific goal. It's a numerical value here. On a more practical level, it may be a project's cost, output, profit margin, or even materials saved through a simplified work flow. Based on a set of constraints and the relationship between one or more decision variables, an objective function aims to maximize profits or reduce losses. For example, capacity, availability, resources, and technology are all examples of restrictions.

Definition 2.1.4. [62] A feasible region is a region that may be described by a set of coordinates that meet a set of inequality conditions. The region meets all the constraints of a linear programming scenario . The idea is to use it as a method of improving things. see figure (2.2):

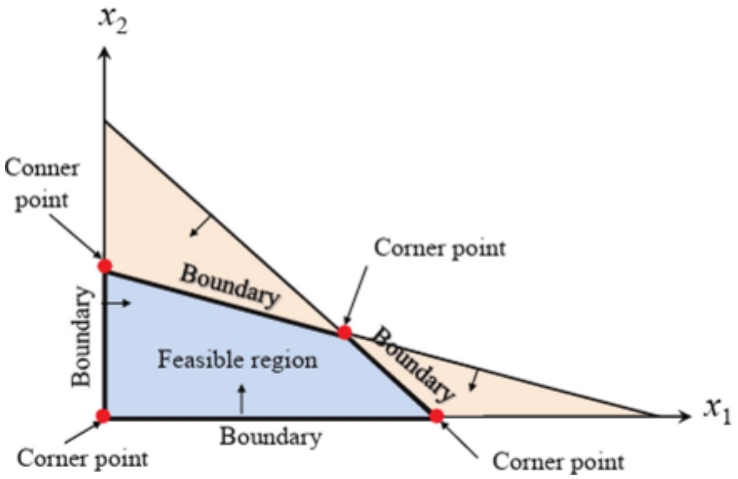


Figure 2.2: Feasible Region

Definition 2.1.5. [13] Let $S \subseteq R^n$ be a nonempty set. If $f : S \to R$, the argument of the minimum is the set of elements in S that achieve the global minimum in S, which is defined by :

1. $x^* \in S$ is called a global minimum point of f over S if $f(x) \geq f(x^*)$ for any $x \in S$.
2. $x^* \in S$ is called a strict global minimum point of f over S if $f(x) > f(x^*)$ for any $x^* \neq x \in S$.
3. $x^* \in S$ is called a global maximum point of f over S if $f(x) \leq f(x^*)$ for any $x \in S$.
4. $x^* \in S$ is called a strict global maximum point of f over S if $f(x) < f(x^*)$ for any $x^* \neq x \in S$.

Definition 2.1.6. [2] Let $f : R^n \to R$ and $x^* \in R^n$. Then x^* is called a local minimizer of f if there is a scalar $t > 0$ such that $f(x^*) \leq f(x)$ for all $x \in B(x^*, t) = \{x \in R^n \mid \| x - x^* \| \leq t\}$, see figure (2.3):

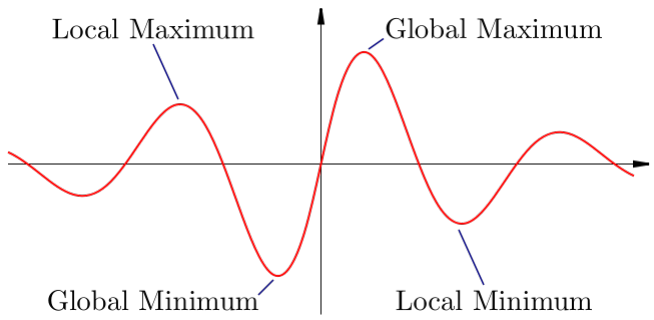


Figure 2.3: Local and Global Solutions of Optimization Problem

2.2 Optimization Problems and Their Classifications

Optimization problems can be characterized as either continuous or discrete based on the nature of the objective and constraint functions. Functions must be continuously optimized in continuous optimization [76] . Discrete has integral values at the optimal

solution for some or all of the n choice variables, while the solution value for each real number in real can be any of the n choice variables. Unconstrained or constrained, problems that have the set $S = R^n$ (the domain S is all of space) are called unconstrained. The problem is said to as constrained if S is a proper subset of R^n, see figure (2.4):

Definition 2.2.1. [63] A real-valued function $f(x)$ defined on a subset S of R^n is said to be Continuous at x if $x_k \to x$ implies $f(x_k) \to f(x)$. Equivalently, $f(x)$ is continuous at x if for given an $\varepsilon > 0$, there is $\mu > 0$ such that $\|y - x\| < \mu$ implies $|f(y) - f(x)| < \varepsilon$ for all $y \in S$.

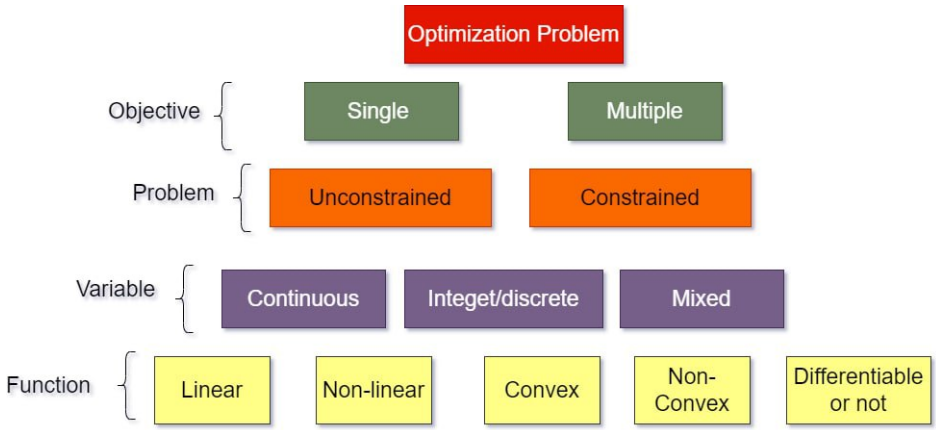


Figure 2.4: Optimization Problems and Their Classifications

2.2.1 Constrained Optimization

Constrained minimization can be expressed as follows:

$$\begin{cases} Minimize: & f(x) \\ Subject\ to: & j_i(x) = v_i \quad for\ i = 1, ..., n \quad Equality\ \ constraints \\ & k_i(x) \geq b_i \quad for\ j = 1, ..., m \quad Inequality\ \ constraints \end{cases} \tag{2.1}$$

where $j_i(x) = v_i\ for\ i = 1, ..., n$ and $k_i(x) \geq b_i\ for\ j = 1, ..., m$. The objective function $f(x)$ must be optimized within the set of constraints, which are defined as the requirements

for satisfying the constraint [11].

Apply the Lagrangian multipliers approach on an equation with only equality constraints to turn it into an unconstrained issue with the number of variables equal to the number of variables originally present.

The objective function can be omitted from consideration if the constraints are all equality constraints and all linear, leaving an unconstrained problem involving a reduced number of variables with a smaller number of variables to be considered [72].

When solving optimization problems, Lagrange multipliers can be used:

$$\begin{cases} \textit{Minimize :} & f(x,y) \\ \textit{Subject to :} & h(x,y) = 0 \end{cases} \tag{2.2}$$

First partial derivatives of both g and h must be continuous in order to answer these problems clearly. As a result of the introduction of the Lagrange multiplier (λ), we are able to solve the Lagrange function as follows:

$$\varphi(x,y,\lambda) = f(x,y) - \lambda h(x,y).$$

To solve a Lagrange function find:

$$\bigtriangledown_{x,y,\lambda}\varphi(x,y,\lambda) = 0$$
$$\bigtriangledown_{x,y} f(x,y) = -\lambda \bigtriangledown_{x,y} h(x,y).$$

This defines a system of three equations with three unknowns. The Lagrange method generalizes to functions with n variables. $\bigtriangledown_{x_1,\ldots,x_n,\lambda}\varphi(x_1,\ldots,x_n,\lambda) = 0$
which requires solving $(n+1)$ equations for $(n+1)$ unknowns, see figure (2.5):

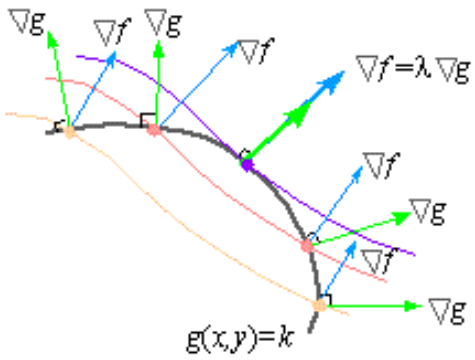


Figure 2.5: Lagrange Function

2.2.2 Unconstrained Optimization

The minimization of an objective function that is dependent on actual variables but does not have any constraints on their values is considered an unconstrained optimization issue. Mathematically [67].

let $x \in R^n$ be a real vector with $n \geq 1$ components and let $\Psi : R^n \to R$ be a smooth function. Then the unconstrained optimization problem is

$$min_x \Psi(x)$$

Constrained optimization problems can be reformulated into unconstrained optimization problems and vice versa, resulting in both types of problems. In many cases, penalized terms in the objective function can be used to substitute the constraints of optimization problem and the problem can be solved as an unconstrained problem.

2.3 Basic Facts and Definitions

2.3.1 Convex Sets

Definition 2.3.1. [47] Let the set $S \subset R^n$. if, for any$z_1, z_2 \in S$, we have

$$\lambda z_1 + (1-\lambda) z_2 \in S, \forall \lambda \in [1, 0], \tag{2.3}$$

then S is said to be a convex set.

Based on this definition, the line segment connecting any two points $z_1, z_2 \in S$ is completely contained in S for any two points $z_1, z_2 \in S$. It also asserts that two random points in S can be connected by a continuous path, which is to say that S is path-connected. It is possible to inductively prove that the set $S \subset R^n$ is convex if and only if, for any $z_1, z_2, ..., z_m \in S$, the set is convex.

$$\sum_{i=1}^{m} \lambda_i z_i \in S, \tag{2.4}$$

where $\sum_{i=1}^{m} \lambda_i = 1, \quad \lambda_i \geq 0, \quad i = 1, ..., m.$

Definition 2.3.2. [29] A path connected domain is a domain where every pair of points in the domain can be connected by a path going through the domain.

In eq (2.3), $z = \lambda z_1 + (1-\lambda) z_2, \quad where \ \lambda \in [0, 1], \quad$ is called a convex combination of z_1 and z_2. In eq (2.4), $z = \sum_{i=1}^{m} \lambda_i z_i \quad$ is called a convex combination of $z_1, ..., z_m \quad$ where $\sum_{i=1}^{m} \lambda_i = 1, \quad \lambda_i \geq 0, \quad i = 1, ..., m.$

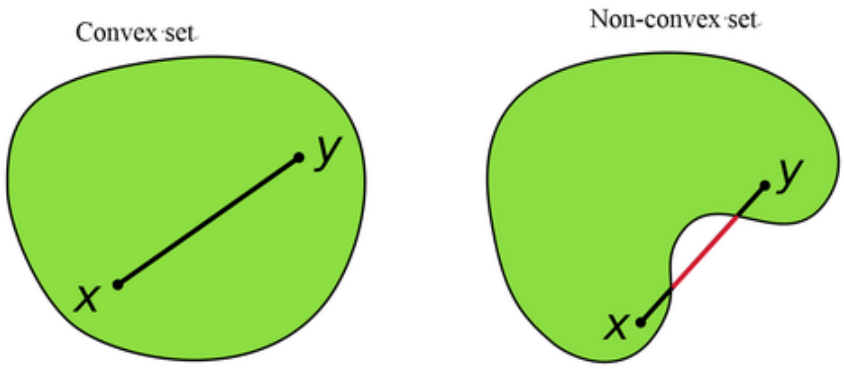


Figure 2.6: Convex set and non Convex set

2.3.2 Convex Functions

Definition 2.3.3. [38] Let the set $S \subset R^n$ be a nonempty convex set. Let $f : S \subset R^n \to R$. If, for any $z_1, z_2 \in S$, and all $\lambda \in [0, 1]$, we have

$$f(\lambda z_1 + (1-\lambda)z_2) \leq \lambda f(z_1) + (1-\lambda)f(z_2), \tag{2.5}$$

then f is said to be convex on S. If the above inequality is true as a strict inequality for all $z_1 = z_2$, i.e.,

$$f(\lambda z_1 + (1-\lambda)z_2) < \lambda f(z_1) + (1-\lambda)f(z_2), \tag{2.6}$$

then f is called a strict convex function on S. If there is a constant $c > 0$ such that for any $z_1 = z_2$,

$$f(\lambda z_1 + (1-\lambda)z_2) \leq \lambda f(z_1) + (1-\lambda)f(z_2) - \frac{1}{2}c\,\lambda(1-\lambda)\parallel z_1 - z_2 \parallel^2, \tag{2.7}$$

then f is called a uniformly (or strongly) convex function on S,(see Figure 2.7)

Definition 2.3.4. [75] A normed vector space $X = (X, \|\cdot\|_\chi)$ is said to be uniformly convex if for sequences $\{x_n\} = \{x_n\}_{n=1}^{\infty}, \{y_n\} = \{y_n\}_{n=1}^{\infty}$, the assumptions $\|x_n\|_\chi \leq 1$, $\|y_n\|_\chi \leq 1$, and $\|x_n + y_n\|_\chi \leq 2$ together imply that $\|x_n - y_n\|_\chi \to 0$.

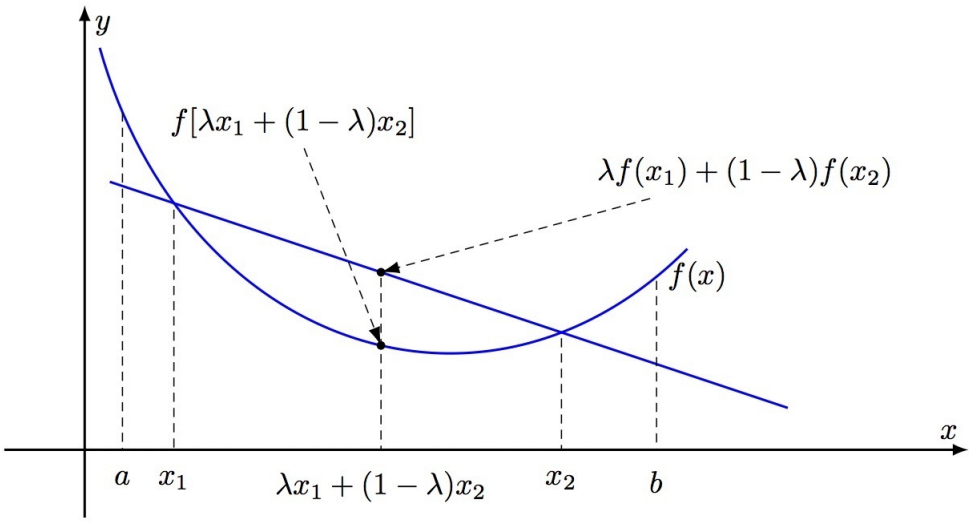


Figure 2.7: Convex Functions

A function f is said to be concave on S if and only if it is convex (strictly convex, uniformly convex) on S.

Example 2.3.1.

1-The functions $h(x) = -x^2$ and $j(x) = \sqrt{x}$ are concave on their domains, as their second derivatives $h''(x) = -2$ and $j''(x) = \dfrac{-1}{4x^{3/2}}$ are always negative.

2-Any affine function $h(x) = ax + b$ is both concave and convex, but neither strictly-concave nor strictly-convex.

3-The sine function is concave on the interval $[0, \pi]$.

2.3.3 Epigraph and Hypograph of a Function

Function $\Psi : R^n \to R$ is epigraph or supergraph is the collection of points that are either on or above its graph, and denoted by $epi(\Psi)$ [3] :

$$epi\Psi = \{(x, \eta) : x \in R^n, \eta \in R, \eta \geq \Psi(x)\} \subseteq R^{n+1} \tag{2.8}$$

The hypograph of $\Psi : R^n \to R$ is the set of points lying on our below its graph, defined by:

$$hyp\Psi = \{(x, \eta) : x \in R^n, \eta \in R, \eta \leq \Psi(x)\} \subseteq R^{n+1} \tag{2.9}$$

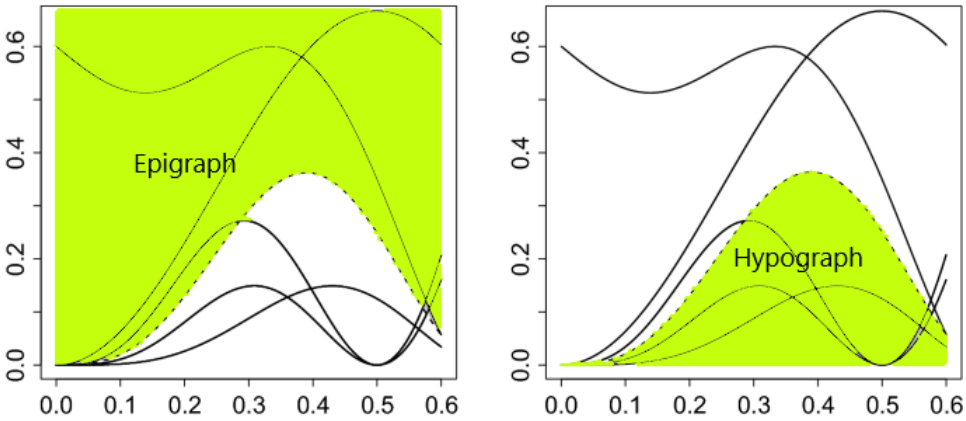


Figure 2.8: Epigraph and Hypograph for Function Ψ

2.3.4 Norm

Definition 2.3.5. [41] A mapping $\| \cdot \|$ is called a norm if and only if it satisfies 6 the following properties :

1- $\| z \| \geq 0, \ \forall z \in R^n; \ \| z \| = 0$ if and only if $z = 0$;

2- $\| \lambda z \| = | \lambda | \| z \|, \ \forall \lambda \in R; \ z \in R^n$,

3- $\| z + c \| \leq \| z \| + \| c \|, \ \forall z, \ c \in R^n$.

The following are well-known applications of vector norm:

$$\| z \|_\infty = \max_{1 \leqslant i \leqslant n} | z_i |, \quad (l_\infty - norm) \tag{2.10}$$

$$\| z \|_1 = \sum_{i=1}^{n} | z_i |, \quad (l_1 - norm) \tag{2.11}$$

$$\| z \|_2 = (\sum_{i=1}^{n} | z_i |^2)^{1/2}, \quad (l_2 - norm) \tag{2.12}$$

The above examples are particular cases of l_k-norm which is defined as

$$\| z \|_k = (\sum_{i=1}^{n} | z_i |^k)^{1/k}, \quad (l_k - norm). \tag{2.13}$$

The ellipsoid norm, another commonly used vector norm, is defined as

$$\| z \|_S = (z^T S z)^{1/2}, \tag{2.14}$$

It is defined by the Euclidean norm

$$\| z \|_2 = \sqrt{z^T z} = \sqrt{\sum_{i=1}^{n} z_i^2}. \tag{2.15}$$

Definition 2.3.6. [73] A function $\langle \cdot, \cdot \rangle : R^n \times R^n \to R$ is an inner product if :

1- $\langle x, x \rangle > 0$, $\langle x, x \rangle = 0 \Leftrightarrow x = 0$ (positivity).

2- $\langle x, y \rangle = \langle y, x \rangle$ (symmetric).

3- $\langle \alpha x, y \rangle = \alpha \langle x, y \rangle \; for \; all \; \alpha \in R$ (homogeneity).

4- $\langle x + y, z \rangle = \langle x, z \rangle + \langle y, z \rangle$ (additivity).

2.3.5 Convergence Rate

Order and rate of convergence in numerical analysis are parameters that describe the speed at which a convergent sequence is approaching its limit [27]. It is said that a sequence x_n that converges to x^* has order of convergence $q \geqslant 1$ and rate of convergence μ if it does.

$$\lim_{n\to\infty} \frac{\mid x_{n+1} - x^* \mid}{\mid x_n - x^* \mid^q} = \mu. \tag{2.16}$$

It's also known as the asymptotic error constant μ. Some authors will use rate, which is not standardized e.q.

When calculating numerical approximations using iterative methods, the rate and order of convergence might provide useful information. Typically, fewer repetitions are required to get a meaningful approximation when the order of convergence is higher. If you take a look at a sequence's asymptotic behavior, to find anything conclusive about any one part of the sequence. Discretization approaches employ similar notions. A measure of the method's efficiency is its speed at which the discretized issue solution converges to the continuous problem solution when the grid size decreases to zero. It is, however, distinct from the language for iterative approaches in this regard.

2.3.6 Adjacency Matrix

An adjacency matrix is a square matrix used to represent a finite graph in graph theory and computer science. The vertices in the graph are either adjacent or not according to the values in the matrices [79]. The adjacency matrix is a (0,1)-matrix with zeros on its diagonal in the special situation of a finite simple graph. Since all of the edges of an undirected graph are bidirectional, the adjacency matrix will be symmetrical.

Definition 2.3.7. [83] The adjacency matrix of an undirected graph G with n nodes is $(n \times n)$ matrix A = $[a_{ij}]$ and is defined by:

$a_{ij} = 1$ { if there is a path exists from Vi to Vj }

$a_{ij} = 0$ { otherwise }

Let's take a look at some of the most crucial aspects of the adjacency matrix.

- In the case of a row and a column, if there is an edge connecting the vertices Vi and Vj, the value of $a_{ij} = 1$.
- The value of $a_{ij} = 0$ if there is no edge between vertex Vi and Vj.
- The diagonal of the vertex matrix (or adjacency matrix) should be zero if the simple graph does not contain any self loops.
- For an undirected graph, the adjacency matrix is symmetric. Values in both rows and columns must be equal to each other, according to this rule.
- This means that, when the adjacency matrix is multiplied by itself, the route from Vi to Vj has the equivalent length of 2 if a non-zero value is present at the i th row and j th column. The adjacency matrix has a non-zero value that indicates the number of unique pathways.

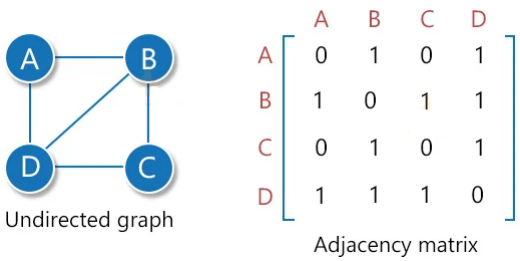


Figure 2.9: Adjacency Matrix

2.3.7 Symmetric Matrices

Definition 2.3.8. [6] A matrix is symmetric if $B = B^T$, is equal to itself. When the indices are swapped, the matrix B is symmetric if and only if its components are unchanged. i.e., $b_{ij} = b_{ji}$.

$$B = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 2 & 5 & 6 & 7 \\ 3 & 6 & 8 & 9 \\ 4 & 7 & 9 & 10 \end{bmatrix} \tag{2.17}$$

is symmetric while the matrix

$$B = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 12 & 5 & 6 & 7 \\ 3 & 6 & 8 & 9 \\ 4 & 7 & 9 & 10 \end{bmatrix} \tag{2.18}$$

is not symmetric as $2 = b_{12} \neq b_{21} = 12$.

An $(n \times n)$ symmetric real matrix B is said to be positive-definite if $u^T Bu > 0$ for all non-zero u in $\mathbb{R}^n$.Formally,

B positive-definite $\Leftrightarrow u^T Bu > 0$ for all u $\in \mathbb{R}^n \setminus \{0\}$.

An $(n \times n)$ symmetric real matrix B is said to be positive-semidefinite or non-negative-definite if $u^T Bu \geq 0$ for all u in $\mathbb{R}^n$.Formally,

B positive-semidefinite $\Leftrightarrow u^T Bu \geq 0$ for all u $\in \mathbb{R}^n \setminus \{0\}$.

An $(n \times n)$ symmetric real matrix W is said to be negative-definite if $x^T Wx < 0$ for all non-zero x in $\mathbb{R}^n$.Formally,

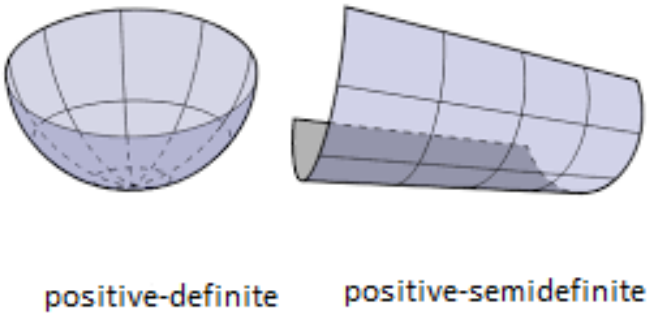


Figure 2.10: Positive-definite and Positive-semidefinite

W negative-definite $\Leftrightarrow x^T W x < 0$ for all x $\in \mathbb{R}^n \setminus \{0\}$.

An $(n \times n)$ symmetric real matrix W is said to be negative-semidefinite or non-positive-definite if $x^T W x \leq 0$ for all x in $\mathbb{R}^n$.Formally,

W negativ-semidefinite $\Leftrightarrow x^T W x \leq 0$ for all x $\in \mathbb{R}^n \setminus \{0\}$.

2.3.8 Gradient and Hessian

Definition 2.3.9. [6] Let M be an open subset of $\mathbb{R}^n$ and $f : M \to \mathbb{R}$ a function . The partial derivative of f at the point $b = (b_1, ..., b_n) \in$ M with respect to the i-th variable x_i is defined as

$$\begin{cases} \dfrac{\partial}{\partial x_i} f(a) = \lim\limits_{h \to 0} \dfrac{f(b_1, ..., b_{i-1}, b_i + h, a_{i+1}, ..., a_n) - f(b_1, ..., a_i, ..., a_n)}{h} \\ \qquad\qquad = \lim\limits_{h \to 0} \dfrac{f(b + he_i) - f(b)}{h} \end{cases} \tag{2.19}$$

Even if all partial derivatives $\dfrac{\partial f}{\partial x_i}(b)$ exist at a given point b,where e_i is i th unit vector.

Definition 2.3.10. [86] Let $f : \mathbb{R}^n \to \mathbb{R}$, the Gradient is a vector that comprises the partial derivatives of the function with respect to the design variables $(x_1, x_2, ..., x_n)$.

$$\bigtriangledown f(x) = \begin{bmatrix} \dfrac{\partial f}{\partial x_1} \\ \dfrac{\partial f}{\partial x_2} \\ .... \\ \dfrac{\partial f}{\partial x_n} \end{bmatrix} \tag{2.20}$$

Definition 2.3.11. [22] The $(n \times n)$ symmetric matrix is known as the Hessian matrix.

$$\bigtriangledown^2 f(x) = \begin{bmatrix} \dfrac{\partial^2 f}{\partial x_1 \partial x_1} & \dfrac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \dfrac{\partial^2 f}{\partial x_1 \partial x_n} \\ \dfrac{\partial^2 f}{\partial x_2 \partial x_1} & \dfrac{\partial^2 f}{\partial x_2 \partial x_2} & \cdots & \dfrac{\partial^2 f}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \dfrac{\partial^2 f}{\partial x_n \partial x_1} & \dfrac{\partial^2 f}{\partial x_n \partial x_2} & \cdots & \dfrac{\partial^2 f}{\partial x_n \partial x_n} \end{bmatrix} \tag{2.21}$$

The Hessian is symmetric if the second partials are continuous.

Definition 2.3.12. [90] The Jacobian of a function $f : \mathbb{R}^n \to \mathbb{R}^m$ is the matrix of its first partial derivatives .

$$Jf(x) = \begin{bmatrix} \frac{\partial f_1}{\partial x_1}(a) & \frac{\partial f_1}{\partial x_2}(a) & \dots & \frac{\partial f_1}{\partial x_n}(a) \\ \frac{\partial f_2}{\partial x_1}(a) & \frac{\partial f_2}{\partial x_2}(a) & \dots & \frac{\partial f_2}{\partial x_n}(a) \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1}(a) & \frac{\partial f_m}{\partial x_2}(a) & \dots & \frac{\partial f_m}{\partial x_n}(a) \end{bmatrix} \tag{2.22}$$

Definition 2.3.13. [52] The trace of the Hessian matrix is known as the Laplacian operator denoted by: $\bigtriangledown^2$,

$$\bigtriangledown^2 f = \ trace(H) = \ \frac{\partial^2 f}{\partial x_1^2} + \frac{\partial^2 f}{\partial x_2^2} + \cdots + \frac{\partial^2 f}{\partial x_n^2}. \tag{2.23}$$

Definition 2.3.14. [77] The sum of the diagonal elements of a matrices is known as the trace :

$$\text{trace (A)} = a_{11} + a_{22} + \cdots + a_{mn}$$

$$tr\ (A) = \sum a_{ii} \tag{2.24}$$

2.4 Nonlinear Optimization

When solving optimization problems that have nonlinear constraints or nonlinear objective functions, nonlinear programming is the technique of solving them. Subject to restrictions like as linear constraints and nonlinear constraints, an objective function can be minimized or maximized in this manner. Constrictions can be based on inequalities or equalities, depending on the context [4]. It also aids in understanding design trade offs, selecting optimal designs and calculating optimal trajectories, as well as model calibration in computation finance, using nonlinear programming techniques. A

nonlinear constrained optimization problem can be formulated as follows in mathematics:

$$(Q)\quad \begin{cases} \textit{Minimize}: & f(n_1, n_2, \ldots, n_t; r_1, r_2, \ldots, r_s) \\ \textit{Subject to}: & g_j(n_1, n_2, \ldots, n_t; r_1, r_2, \ldots, r_s) \geq b_j, \\ & j = 1, 2, \ldots, M \qquad a_i \leq n_i \leq b_i,\ l_k \leq r_k \leq n_k, \end{cases} \tag{2.25}$$

n_i are integers or discrete variables, are real (continuous) variable,
$1 \leq i \leq t,\ 1 \leq k \leq s$. Real, integer, and discrete variables are all involved in the problem (Q), which is a very general optimization problem.

Definition 2.4.1. [10] Integer variables are variables that must take an integer value (0, 1, 2, ...). Binary variables are a kind of integer variable that is different from others. Binary variables can only take the values 0 or 1. They are integer variables with a maximum of 1 and an implicit minimum of 0 on each one.

Definition 2.4.2. [18] A variable is discrete variable if and only if it has a one-to-one relationship with the set of natural numbers, N. In other words, a variable that is discrete over a certain range of real values is one for which, for any value in the range of values that the variable is permitted to take, there is a space Positive minimum of the next other value that the variable is allowed to take. Either there are a finite number of possible values or the number is infinite. Common instances are variables that must be integers, non-negative integers, positive integers, or just 0 and 1.

Definition 2.4.3. [63] A real-valued function $f(x)$ defined on a subset S of $\mathbb{R}^n$ is said to be continuous at x if $x_k \to x$ implies $f(x_k) \to f(x)$. Equivalently, $f(x)$ is continuous at x if for given an $\varepsilon > 0$, there is $\mu > 0$ such that $\| y - x \| < \mu$ implies $| f(y) - f(x) | < \varepsilon$ for all $y \in S$.

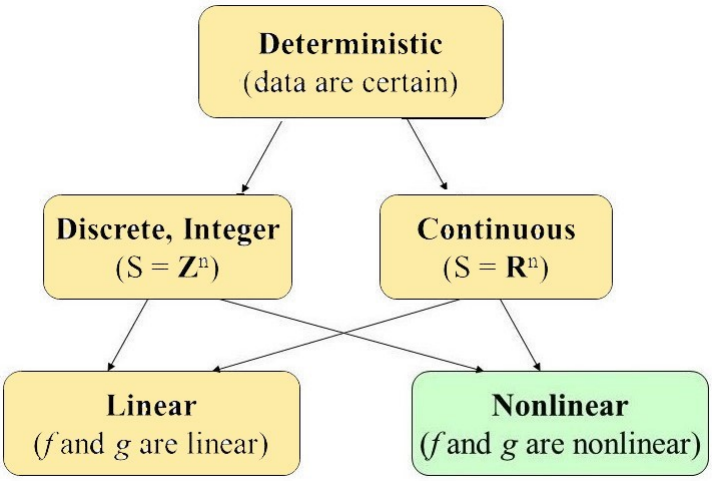


Figure 2.11: Nonlinear Optimization

2.5 KKT Conditions for Nonlinear Problems

The standard form of the nonlinear optimization problem is given by

$$(P)\begin{cases} \textit{Minimize :} & f_0(z), \\ \textit{Subject to :} & f_i(z) \leq 0, \quad i = 1, \ldots, I, \\ & g_i(z) = 0, \quad i = 1, \ldots, E \\ & z \in Z \end{cases} \tag{2.26}$$

We define the Lagrangian function as

$$\zeta(z, \lambda, \upsilon) = f_0(z) + \sum_{i=1}^{I} \lambda_i f_i(z) + \sum_{i=1}^{E} \upsilon_i g_i(z), \tag{2.27}$$

where $\lambda \in \mathbb{R}_+^I$ and $\upsilon \in \mathbb{R}^E$ are the dual variables (Lagrange multipliers). Then we define the Lagrangian dual function to be

$$\psi(\lambda, \upsilon) = \min_{z \in Z} \zeta(z, \lambda, \upsilon) \tag{2.28}$$

We denote the optimal value of the Lagranian dual problem by d^*, the optimal value of

primal problem by p^*, and z^* is an optimal solution of (P). The Lagrange dual problem with dual variables (λ, υ) is given by

$$(D)\begin{cases} \textit{maximize :} & \psi(\lambda, \upsilon) \\ \textit{Subject to :} & \lambda \geq 0. \end{cases} \tag{2.29}$$

The dual function provide lower bounds on the optimal value p^* of the problem (2.26), that is, for any $\lambda \geq 0$ and any we have

$$\psi(\lambda, \upsilon) \leq p^* \tag{2.30}$$

2.5.1 KKT Conditions

The foundation of linear and nonlinear programming is Karush-Kuhn-Tucker (KKT) conditions (for a problem with differentiable $f_i,\ g_i$) [35] :

- Primal constraints: $f_i(z) \leq 0,\ i = 1, \ldots, I$, and $g_i(z) = 0,\ i = 1, \ldots, E$.
- Complementary slackness: $\lambda_i f_i(z) = 0,\ i = 1, \ldots, I$.
- Dual constraints: $\lambda \geq 0$.
- Gradient of Lagrangian with respect to z vanishes:

$$\nabla f_0(z) + \sum_{i=1}^{I} \lambda_i \nabla f_i(z) + \sum_{i=1}^{E} \upsilon_i \nabla g_i(z) = 0. \tag{2.31}$$

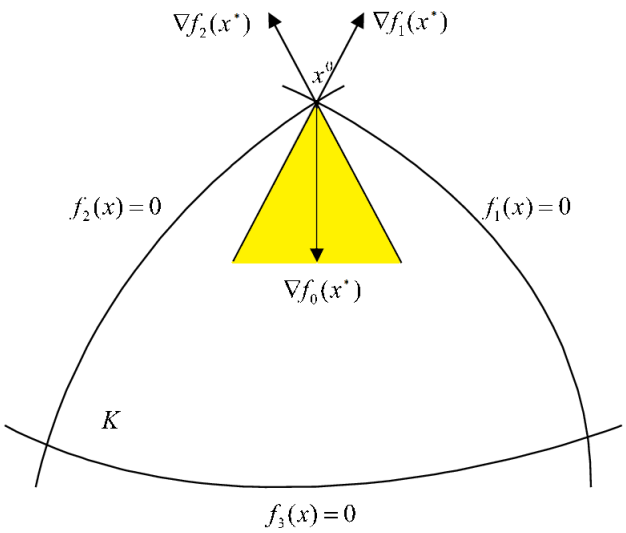


Figure 2.12: KKT Conditions

Example 2.5.1. Consider the general optimization problem :

$$\begin{cases} \textit{Minimize :} & z_1 z_4(z_1+z_2+z_3)+z_3 \\ \textit{Subject to :} & z_1 z_2 z_3 z_4 \geq 25 \\ & z_1^2+z_2^2+z_3^2+z_4^2=40 \\ & 1 \leq z_1, z_2, z_3, z_4 \leq 5 \\ & z_0=(1,5,5,1). \end{cases} \tag{2.32}$$

In order to find the best solution, the optimizer must first minimize a nonlinear objective function. At the optimal solution, the values of the variables are constrained by both equality constraints ($= 40$) and inequality constraints (> 25). Both the sum of the squares of the variables and the product of the four variables must be greater than 40. Also, all variables must be between 1 and 5, and the starting guesses are $z_1 = 1, z_2 = 5, z_3 = 5$, and $z_4 = 1$.

Code 1:Use KKT Conditions for Nonlinear Problems

```
import numpy as np
from scipy.optimize import minimize
def objective(z):
    return z[0]*z[3]*(z[0]+z[1]+z[2])+z[2]
def constraint1(z):
    return z[0]*z[1]*z[2]*z[3]-25.0
def constraint2(z):
    sum_eq = 40.0
    for i in range(4):
        sum_eq = sum_eq - z[i]**2
    return sum_eq
# initial guesses
n = 4
z0 = np.zeros(n)
z0[0] = 1.0
z0[1] = 5.0
z0[2] = 5.0
z0[3] = 1.0
# show initial objective
print('Initial SSE Objective: ' + str(objective(z0)))
# optimize
b = (1.0,5.0)
bnds = (b, b, b, b)
con1 = {'type': 'ineq', 'fun': constraint1}
con2 = {'type': 'eq', 'fun': constraint2}
cons = ([con1,con2])
solution = minimize(objective,z0,method='SLSQP',
bounds=bnds,constraints=cons)
z = solution.z
# show final objective
print('Final SSE Objective: ' + str(objective(z)))
# print solution
print('Solution')
print('z1 = ' + str(z[0]))
print('z2 = ' + str(z[1]))
print('z3 = ' + str(z[2]))
print('z4 = ' + str(z[3]))
```

Initial Objective: 16.0

Final Objective: 17.01401724563517

Solution :

$z_1 = 1.0$, $z_2 = 4.742996096883977$, $z_3 = 3.8211546234095715$, $z_4 = 1.379407645075325$.

2.6 The Duality

There are two ways to look at an optimization problem: as a primal problem or as a dual problem. In some cases, an issue can be solved more effectively in its dual form than in its primary version [15]. In the case of a minimization issue, any possible solution in its dual version serves as a lower bound on its primary version solution. Hence, the duality gap is the gap between the dual and primal optimal solutions. The duality gap is 0 for linear and convex nonlinear problems. These problems necessitate a solution that is best for both the primary and the secondary. As a result, looking for optimality is the same as looking for a solution that works in both primal and dual contexts.

2.6.1 Linear Programming

The problem of minimizing a linear function with linear constraints is known as linear programming. In standard form, a linear programming problem (LP) is written as

$$(LP)\begin{cases} \text{Minimize} & \langle c, x\rangle \\ \text{Subject to} & \langle a_i, x\rangle = b_i, \quad i = 1, \ldots, m \\ & x \geq 0, x \in \mathbb{R}^n. \end{cases}$$

The problem data are given by the vectors $a_i \in \mathbb{R}^n$, for $i = 1, \ldots, m$, the numbers $b_i \in \mathbb{R}$, for $i = 1, \ldots, m$, and the vector $c \in \mathbb{R}^n$. Consider the primal (P) and the dual (D) standard linear programming problem: and

$$(D)\begin{cases} \text{Maximize} & \langle b, y\rangle \\ \text{Subject to} & A^T y \geq c \end{cases}$$

One of the most surprising and valuable characteristics of linear programming is that we may correlate a matching dual problem with every LP problem. It's worth noting that the

optimizing a linear function over a polyhedron once more. The primal problem (LP-P) and its dual problem (LP-D) have extremely natural and direct algebraic connections, as we will demonstrate (LP-D) [33].

2.6.2 The Relationships Between Primal and Dual Problem

If there is a solution to the primal problem, there is likewise a solution to the dual problem. Furthermore, the solution value for both problems is the same, i.e., the primal objective function's maximum value is exactly equal to the dual function's minimum value [58].

Example 2.6.1. Consider the following simple linear programming problem:

$$(P)\begin{cases}\text{Maximize} & x_1+2x_2+4x_3\\ \text{Subject to} & x_1+x_2+x_3=9,\\ & x\geq 0.\end{cases} \tag{2.33}$$

The dual of problem (2.33) is given by

$$(D)\begin{cases}\text{Minimize} & 9y_1\\ \text{Subject to} & y_1\geq 1,\\ & y_1\geq 2,\\ & y_1\geq 4.\end{cases} \tag{2.34}$$

The optimal solution of problem eq(2.33) is $x_1=x_2=0$, $x_3=9$ and the optimal value is $x_1+2x_2+4x_3=36$; an optimal solution of problem eq(2.34) is $y_1=4$, and the optimal value is $9y_1=36\Rightarrow y_1=\dfrac{36}{9}=4$.

The following table Eq(2.1) briefly shows the relationship between primal and dual Problem :

Primal	**Dual**
$\leq$ constraint	constraint ≥ 0
$\geq$ constraint	constraint ≤ 0
$=$ constraint	constraint ≥ 0 or ≤ 0
max	min
right-hand side	objective coefficient
variable ≥ 0	$\geq$ constraint
variable ≤ 0	$\leq$ constraint
variable ≥ 0 or ≤ 0	$=$ constraint
objective coefficient	right-hand side
number of constraints	number of variables
number of variables	number of constraints

Table 2.1: The Relationships between Primal and Dual Problem.

2.6.3 Nonlinear Problem

Revisiting problem (2.35)

$$\begin{cases} \textit{Minimize :} & f(z) \\ \textit{Subject to :} & g_i(z) \leq 0 \\ & h_i(z) = 0. \end{cases} \tag{2.35}$$

The dual problem of eq(2.35) is given

$$\begin{cases} \textit{Minimize :} & f(z) + \sum_{i=1}^{m} \lambda_i g_i(z) + \sum_{i=1}^{p} \mu_i h_i(z) \\ \textit{Subject to :} & \lambda_i g_i(z) = 0 \\ & \lambda_i \geq 0. \end{cases} \tag{2.36}$$

It is also known as Lagrangian Function, the objective function of a dual-problem situation [42]. The dual feasibility and complementary slackness criteria for the nonlinear situation are the first order derivative of Lagrangian Function, coupled with the two sets of constraints.

2.6.4 Primal Dual Relationship

Definition 2.6.1. [45] Weak duality If v is feasible for (P) and u is feasible for (D), then $b^T u \geq c^T v$.

Definition 2.6.2. [30] Strong duality If v is feasible for (P) and u is feasible for (D), then $b^T u = c^T v$.

Very Linear programming known as the primary is linked to a second Linear programming known as the dual. Both of the issues are primary, and the other is a secondary one. The best solution to any problem indicates the best solution for the other.
Let the primal problem be

$$(P) \begin{cases} \textit{Maximize :} & K_y = c_1 y_1 + c_2 y_2 + \ldots + c_n y_n \\ \textit{Subject to :} & \textit{restrictions} \\ & a_{11} y_1 + a_{12} y_2 + \ldots + a_{1n} y_n \leq b_1 \\ & a_{21} y_1 + a_{22} y_2 + \ldots + a_{2n} y_n \leq b_2 \\ & \vdots \\ & a_{m1} y_1 + a_{m2} y_2 + \ldots + a_{mn} y_n \leq b_n \end{cases} \tag{2.37}$$

and

$$y_1 \geq 0, y_2 \geq 0, \ldots, y_n \geq 0$$

The corresponding dual is defined as

$$(D) \begin{cases} \textit{Minimize :} & K_w = b_1 w_1 + b_2 w_2 + \ldots + b_m w_m \\ \textit{Subject to :} & \textit{restrictions} \\ & a_{11} w_1 + a_{21} w_2 + \ldots + a_{m1} w_m \geq c_1 \\ & a_{12} w_1 + a_{22} w_2 + \ldots + a_{m2} y_m \geq c_2 \\ & \vdots \\ & a_{1n} w_1 + a_{2n} w_2 + \ldots + a_{nm} y_m \geq c_n \end{cases} \tag{2.38}$$

and

$$w_1, w_2, \ldots, w_m \geq 0$$

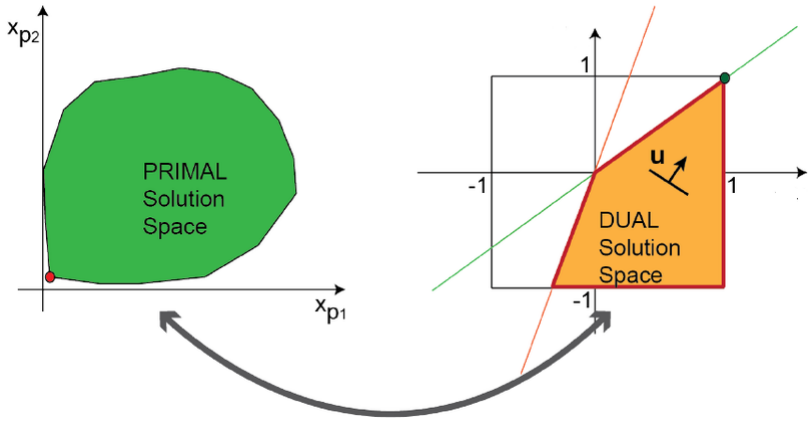


Figure 2.13: Primal Dual Relationship

2.7 Differential Equations

This equation contains one or more derivatives of a function. Derivatives of a dependent variable with respect to a set of independent variables characterize differential equations [32]. One independent variable is enough to create an ordinary differential equation.

Example 2.7.1. Differential Equations:

1- $\frac{d^2y}{dz^2} - \frac{dy}{dz} - 5z = 0$

2- $(\frac{d^2y}{dz^2}) - 7\frac{dy}{dz} + 8y = 0$

3- $[2 + (\frac{dy}{dz})] = (\frac{d^2y}{dz^2})^{\frac{2}{3}}$

Note :(Partial Differential Equation) describes an equation with many partial derivatives of its original function.

Differential equations can be of any order and degree

- Order : greatest differential coefficient
- Degree :When a differential equation is written as [Polynomial in the derivatives], the highest order derivative is raised to that power.

$$f(zy)\frac{d^m y}{dz^m} + g(zm)\frac{d^{m-1}y}{dz^{m-1}} + \ldots = 0. \tag{2.39}$$

Note :

The degree of a differential equation can be specified only if it is written as a polynomial.

- $y' + 4y^2 + y = 0$ at point (1,1)
- $(y')^2 + y^2 - 2 = 0$ at point (1,2)
- $y'' + (2y^1)^5 + \sin y = 0$ at point (2,1)

2.7.1 Formation of Differential Equation

A constant k can be used to represent the variables y and x in an equation where they are the dependent and independent variables. A differential equation can now be formed. Count the number of arbitrary constants (let n be the number of constants). The order of the differential equation will determine the number of arbitrary constants. (n) To remove an arbitrary constant, differentiate the above equation n times [84]. The differential equation derived from the above elimination equation is needed. For the curve-loving family :

$$f(x, y, a, a_2 \dots a_n) = 0 \tag{2.40}$$

There are n distinct parameters represented by $a, a_2 \dots a_n$. The differential equation of the original equation eq (2.40) can be obtained by differentiating it n times.

2.7.2 Classification of Differential Equations

1- Ordinary or partial.

2- First order... and higher orders.

3- linear and non-linear.

4- Homogeneous and heterogeneous.

5- In terms of coefficients (fixed or variable) [21].

2.7.3 Methods for Solving First-Order Differential Equations

2.7.2.1 Exact [74]

1- Separable Equations

A first-order differential equation of the form

$$\frac{dy}{dx} = g(x)h(y). \tag{2.41}$$

2- Linear Equations

A first-order differential equation of the form

$$a_1 \frac{dy}{dx} + a_0(x)y = g(x). \tag{2.42}$$

3- Exact Equations

A differential expression $M(x,y)dx + N(x,y)dy$ is an exact differential in a region R of the xy-plane if it corresponds to the differential of some function $f(x,y)$ defined in R. A first-order differential equation of the form

$$M(x,y)dx + N(x,y)dy = 0. \tag{2.43}$$

4- Solutions by Substitutions

Often the first step in solving a differential equation consists of transforming it into another differential equation by means of a substitution. For example, suppose we wish to transform the first-order differential equation $dy/dx = f(x,y)$ by the substitution $y = g(x,u)$, where u is regarded as a function of the variable x. If g possesses first-partial derivatives, then the Chain Rule

$$\frac{dy}{dx} = \frac{\partial g}{\partial x}\frac{dx}{dx} + \frac{\partial g}{\partial u}\frac{du}{dx} \tag{2.44}$$

gives

$$\frac{dy}{dx} = g_x(x,u) + g_u(x,u)\frac{du}{dx}. \tag{2.45}$$

2.7.4 Approximate Theoreticals

Using the Tangent Line, assume that the first-order initial-value problem

$$y' = f(x,y), \quad y(x_0) = y_0 \tag{2.46}$$

possesses a solution [46]. One way of approximating this solution is to use tangent lines. Euler's Method To generalize the procedure just illustrated, we use the linearization of the unknown solution $y(x)\ of\ (2.42)\ at\ x = x_0$:

$$L(x) = y_0 + f(x_0, y_0)(x - x_0) \tag{2.47}$$

2.7.5 Methods for Solving Higher-Order Differential Equations

1- Rank reduction.

2- Define transactions.

3- Swap variables.

4- Cauchy-Euler method for solving equations in which the rank of the derivative is the same as the exponent of its coefficient.

5- Exponential Sequences(Laplace and Bernoulli equations) [68].

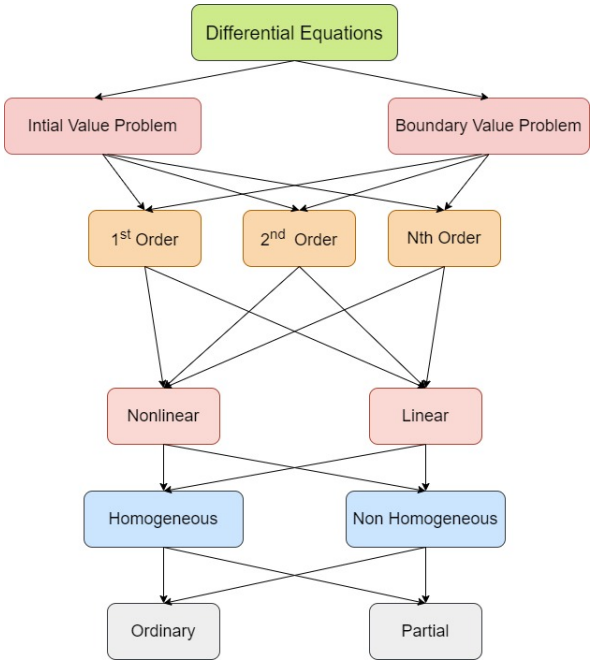


Figure 2.14: Differential Equations

2.8 Graph Theory

A graph is an ordered pair $G = (V, E)$ comprising: [36]

- V, a set of vertices (also called nodes or points).
- $E =$ edges $\subseteq \{\{x, y\} |\ x, y \in V$ and $x \neq y\}$

A collection of edges, also known as links or lines, which are unordered pairings of vertices (that is, an edge is associated with two distinct vertices). This kind of object might be referred to as an undirected simple graph to avoid any confusion. The endpoints of the edge are the vertices x and y in the edge $\{x, y\}$. The edge is described as joining x and y and as incident on both x and y. A vertex in a graph may exist even if it is not an edge. Multiple edges are two or more edges that connect the same two vertices and are not permitted by the definition given above.

2.8.1 Types of Graphs

There are a number of distinct sorts of graphs based on their edges, direction, weight, and more.

1- Simple graph - A simple graph is one in which no two edges connect the same pair of vertices, and in which each edge connects two distinct points. As an example, have a look at this graph. There is no self-loop in this graph, and no more than one

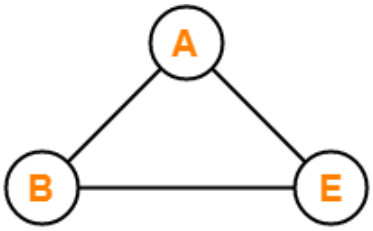


Figure 2.15: Simple Graph

edge connects any two vertices. To designate the edges, the vertices they connect are denoted by the letters A and B.

2- Multigraph - When more than one edge connects the same set of vertices, the graph is referred to as a "multigraph." When two vertices share a common edge, the number of edges reveals how many vertices share a common edge.

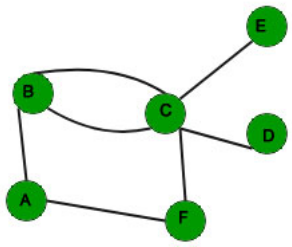


Figure 2.16: Multigraph

The above graph is a multigraph since there are multiple edges between B and C . The multiplicity of the edge $\{B, C\}$ is 2.

The edges in certain graphs are directed, unlike the one illustrated above. This signifies that there is only a one-way relationship between the things. Sometimes, the angle at which a piece's edges meet is critical. [80]

Directed graphs and undirected graphs can be distinguished based on the direction of the edges. Simple directed or undirected graphs and simple directed or undirected multigraphs can be obtained by extending this characteristic to simple graphs and multigraphs. [40]

2.9 Dynamic Optimization Problems

One or more problem parameters, such as objective function, restrictions, or environmental coefficients can be dynamically changed over time in dynamic optimization problems. The dynamic nature of the problem necessitates a more complex optimization approach than a static one [23]. It's not uncommon for the optimal solutions to optimization issues to shift over time because their objective functions vary.

Defining these issues as dynamic optimization issues, we may say that:

$$\begin{cases} \textit{Minimize :} & \Psi(a(t), b(t), c(t), d, t_f) \\ \textit{Subject to :} & \dfrac{da(t)}{dt} = f(a(t), b(t), c(t), d) \\ & g(a(t), b(t), c(t), d) = 0 \\ & a^0 = a(0) \\ & a^{\text{l}} \leq a(t) \leq a^c \\ & b^{\text{l}} \leq b(t) \leq b^c \\ & c^{\text{l}} \leq c(t) \leq c^c \\ & d^{\text{l}} \leq d(t) \leq d^c \end{cases} \tag{2.48}$$

t : time, a : differential variables, b : algebraic variables

t_f : final time, c : control variables, d : time independent parameters.

Nonlinear model predictive control, in particular, and dynamic real-time monitoring Large sets of partial differential equations, with constraints on the control and state variables, can lead to infinite-dimensional issues in dynamic optimization models. The formulation and solution of dynamic optimization problems in critical chemical engineering applications has been made possible by the development of robust numerical techniques and the increase in computational capability. Dynamic optimization is becoming a crucial part of current industrial operations and decision-making processes. see figure (2.17)

2.10 Optimal Control Problems

The first essential element of the optimal control problem is the control component the system. It generates possible behaviours. [4] control systems they are described by

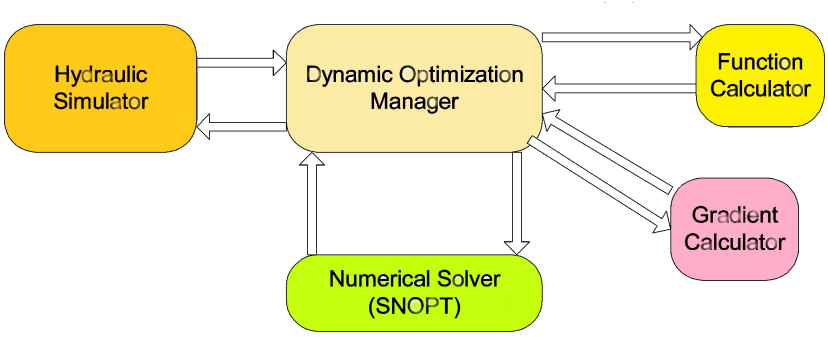


Figure 2.17: Dynamic Optimization

Ordinary Differential Equations (ODEs) of the form

$$y' = f(y, t, u, d), \quad y(t_0) = y_0 \tag{2.49}$$

where y is the state taking values in R^n, u is the control input taking values in some control set $U \subset R^m$, t is time, t_0 is the initial time, and y_0 is the initial state. Both y and u are functions of t, that will often suppress their time arguments .Other controllable parameters might be fixed with time, We refer to these as parameters(d).

$$\begin{cases} \textit{Minimize :} & J = f_{opj}(y, t, u, d) \\ \textit{Dynamic Constraints :} & z = f_{opj}(y, t, u, d) \\ \textit{Time :} & t_{lb} \leq t \leq t_{ub} \\ \textit{State Variables :} & y_{lb} \leq y \leq y_{ub} \\ \textit{Dynamic Controls :} & u_{lb} \leq u \leq u_{ub} \\ \textit{Design Parameters :} & d_{lb} \leq d \leq d_{ub} \\ \textit{Initial Boundary Constraints :} & g_{0,lb} \leq g_0(y_0, t_0, u_0, d) \leq g_{0,ub} \\ \textit{Final Boundary Constraints :} & g_{f,lb} \leq g_f(y_f, t_f, u_f, d) \leq g_{f,ub} \\ \textit{Path Constraints :} & p_{f,lb} \leq g_f(y, t, u, d) \leq p_{f,ub} \end{cases} \tag{2.50}$$

Definition 2.10.1. [1] (Admissible Control) A piecewise continuous control $v(.)$, defined on some time interval $t_0 \leq t \leq t_f$ with range in the control region U

$$v(t) \in v, \quad \forall t \in [t_0, t_f],$$

is said to be an admissible control.

Definition 2.10.2. [71] (Feasible Control, Feasible Pair) An admissible control $\overline{v}(.) \in V[t_0, t_f]$ is said to be feasible, provided that
(i) the response $\overline{x}(.; x_0, v(.))$ is defined on the entire interval $t_0 \leq t \leq t_f$,
(ii) $\overline{v}(.)$ and $\overline{x}(.; x_0, v(.))$ satisfy all of the physical (point and path) constraints during this time interval; the pair $(\overline{v}(.), \overline{x}(.))$ is then called a feasible pair. The set of feasible controls, $\Psi[t_0, t_f]$, is defined as

$$\Psi[t_0, t_f] := \{v(.) \in V[t_0, t_f] : v(.)\ feasible\}$$

Definition 2.10.3. [59] A necessary and sufficient condition for optimality of a control with regard to a loss function is provided by the (Hamilton-Jacobi-Bellman) (HJB) equation. As a rule, the solution to this equation is the value function itself, as it is a nonlinear partial differential equation in the value function. Having found this answer, the optimal controller can be obtained by finding the maximizer (or minimizer) of the Hamiltonian in the HJB equation.

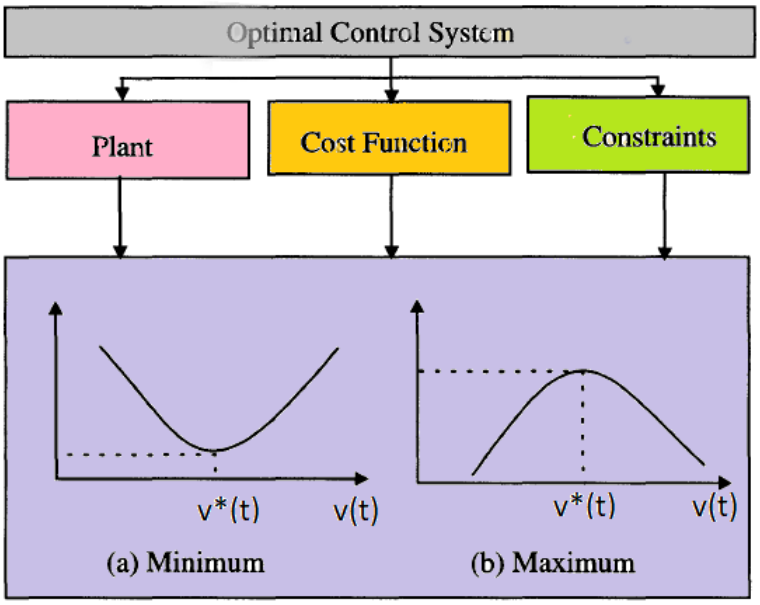


Figure 2.18: Optimal Control

2.11 Optimization of Energy Systems

Energy system models can provide multiple ways to satisfy future demand and environmental goals by guiding decision-making on power capacity expansion. In addition, these models help clarify the function and value of technology in the power system and advise the best periods to invest. Having an accurate picture of technology costs for system planning tools is critical in this situation [28]. Energy System Optimization thermodynamic modeling, analysis and optimization of multiple energy systems in various applications are thoroughly covered. As a new perspective on the system and the process for determining the most appropriate design parameters for enhancing efficiency, cost-effectiveness, and sustainability is provided.

General forms of an unconstraint and constraint optimization problem are expressed by eqs (2.51) .

optimize :

$$\begin{cases} \textit{Maximize :} \quad f_i(x) \quad for \quad i = 1, 2, 3, ...l \\ where \\ \qquad\qquad x = [x_1.x_2....x_n]^T \\ \textit{Subject to :} \\ \qquad\qquad h_i(x) = 0 \qquad\qquad \text{for } i = 1, 2, ..., m \\ \qquad\qquad g_k(x) \leq 0 \qquad\qquad \text{for } k = 1, 2, ..., p \end{cases} \tag{2.51}$$

2.11.1 Application of Dynamic Optimization

Let A and B are the only power generators in a market system with only two power generators, and the goal is to maximize energy and reserve production simultaneously. Each of these generators has a 100 MW power plant at their disposal. Producer A offers energy for sale at \$10 per megawatt hour, while Producer B offers energy for sale at \$30 per megawatt hour. An all-night operation will be explored [56]. 130 MWh will be supplied for 12 hours, followed by 160 MWh for another 12 hours. There is a reserve capacity of 20 megawatts for the first 12 hours and 30 megawatts for the next 12 hours in case of emergency. As opposed to producer A, producer B promises to cover this reserve capacity for \$25/MWh.

As the planning process is carried out hourly, the following equations are relevant (energy and power refer to the same thing).
Objective function

$$\textit{Maximize :} \; TC \sum_t \sum_i (CE_i \times P_{i,t} + CR_i \times R_{i,t}) \tag{2.52}$$

Energy balance equation

$$\sum_i P_{i,t} = DE_t \tag{2.53}$$

Reserve requirement equation

$$\sum_i R_{i,t} = DR_t \tag{2.54}$$

capacity limit equation

$$P_{i,t} + R_{i,t} \leq P_{maxi} \tag{2.55}$$

Other constraints

$$P_{i,t} \geq 0, \quad R_{i,t} \geq 0 \tag{2.56}$$

A number of other restrictions on power plants will be ignored.

Notation

set

i cluster of power generators

t time set.

Parameters

CEi Energy generation cost for producer i [$/MWh]

CRi Reserve capacity price for producer i [$/MW]

DEt Energy demand at time t [MWh]

DRt Reserve capacity requirement at t [MWh]

Pmax,i power plant capacity of the i generator [MWh]

Variables

TC total operating cost

Pi,t planned energy generation value at t for producer i [MW]

Ri,t planned reserve capacity value at t for producer i [MWh]

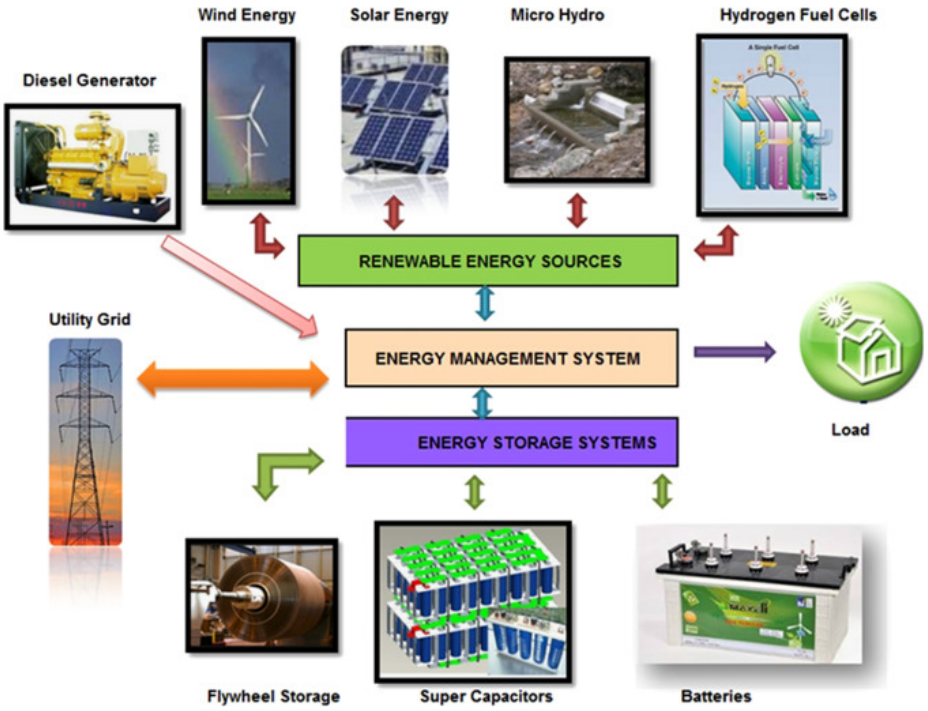


Figure 2.19: Types of Energy

2.12 Algorithm Design

An effective technique to store and organize data is through the use of a data structure. A data structure is a collection of data elements that can be used to execute a variety of different operations on the data. To construct scalable systems, it is essential to grasp algorithm design principles in data structures regardless of the programming language you use [37]. Choosing a suitable algorithm design method is a difficult but critical endeavor. Algorithm design strategies include the following :

- Problem definition
- Development of a model
- Specification of the algorithm
- Designing an algorithm

- Checking the correctness of the algorithm
- Analysis of algorithm
- Implementation of algorithm
- Documentation preparation

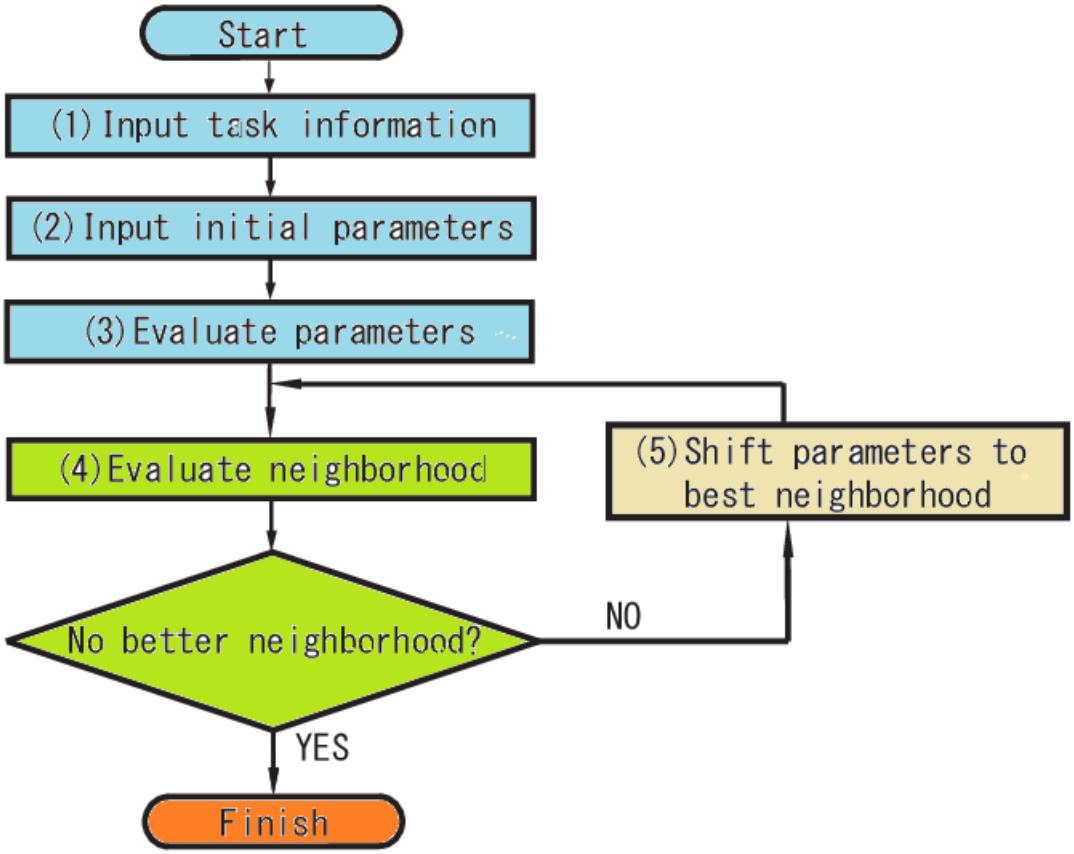


Figure 2.20: Algorithm Design

CHAPTER 3

METHODS TECHNIQUES AND APPLICATIONS

In this chapter, we begin to introduce method and applications to optimization techniques, differential equations, optimization of nonlinear equations, dynamic optimization, energy optimization, and Application of Reliability Optimization and methods of solution.

3.1 Methods Software Technique

Knowledge of the theoretical foundations and range of applications of several important approaches to solving nonlinear and linear programs. Particularly emphasized is their use for solving optimal control problems of ordinary differential equations.

3.1.1 Quadratic Programming

Quadratic programming, or QP, is a way to solve certain optimization problems in mathematics that involve quadratic functions. In particular, the goal is to optimize (minimize or maximize) a multivariate quadratic function when the variables are limited in a linear way. Nonlinear programming includes quadratic programming. In this case, "Programming" means a set of rules for how to solve math problems. This use goes back

to the 1940s, and it has nothing to do with the idea of "computer programming." which is more recent [44]. Quadratic programming (QP) is one of the simplest types of non-linear programming. It is the problem of maximizing a quadratic objective function. [31] Constraints can be of the equality or inequality types, and the objective function can have bilinear or up to second-order polynomial terms. QP is frequently used in the image and signal processing industry, in least-squares regression analysis, in chemical plant scheduling control, and in sequential quadratic programming, a technique for solving more complex non-linear programming problems.

Consider the n-variable, m-constraint quadratic programming issue as follows.

- In which c is an n-dimensional vector of real numbers
- an $(n \times n) - dimensional$ real symmetric matrix Q,
- an $(m \times n) - dimensional$ real matrix A,
- an m-dimensional real vector b.

The goal of quadratic programming is to find the n-dimensional vector x that maximizes a given quadratic function.

$$\begin{cases} Minimize: \ \frac{1}{2}x^T Qx + c^T x \\ Subject\ to: \ Ax \leq l. \end{cases} \tag{3.1}$$

Where x^T is the vector transpose of x and $Ax \leq l$ denotes that each entry of vector Ax is less than or equal to its corresponding entry in vector l. (component-wise inequality).

Example 3.1.1. Combining linear equality and inequality constraints with a quadratic objective function are the main parts of a general quadratic programming formula.

$$\begin{cases} \textit{Minimize :} & f(x) = g^T x + \dfrac{1}{2} x^T G x \\ \textit{Subject to :} & Ax = a \\ & Bx \leq b \\ & x \geq 0. \end{cases} \tag{3.2}$$

The objective function is set up so that the vector g has all of the linear terms (that are only differentiated once) and the vector G has all of the quadratic terms (that are differentiated twice). To put it more simply, G is the objective function's Hessian matrix, and g is its gradient. The general formulation often omits the objective function's constants. By placing a halve in front of the quadratic term, the coefficient resulting from the derivative of a second-order polynomial can be eliminated. Constraints are represented by the matrix equation $Ax = a$, which includes all linear equality constraints, and $(Bx \leq b)$ are the linear inequality constraints.

Solution :

In the same way that other nonlinear programming problems can be solved, quadratic programming problems can also be solved. The simplest approach to solving an unconstrained quadratic programming problem is to set the derivative (gradient) of the objective function to zero and solve the problem. The more constraints a formula has, the more difficult it is to solve. The conjugate gradient method is a common way to find a solution when the goal function is strictly convex and there are only equality constraints. If there are constraints that say $(Bx \leq b)$ must be less than b, then the interior point and active set methods are the best ways to find a solution. Most of the time, trust-region methods are used when there is a range of values for x (in the form $(x^L \leq x \leq x^U)$, which is often the case in image and signal processing.

Example 3.1.2. This example shows how to find the KKT point of a particular QP problem:

$$\begin{cases} \textit{Minimize :} & f(x) = 3x_1^2 + x_2^2 + 2x_1x_2 + x_1 + 6x_2 + 2 \\ \textit{Subject to :} & 2x_1 + 2x_2 \geq 4 \\ & x_1, x_2 \geq 0. \end{cases} \tag{3.3}$$

Rewrite the constraints.

$q_1 = -2x_1 - 2x_2 + 4 \leq 0$

$q_2 = -x_1 \leq 0$

$q_3 = -x_2 \leq 0.$

Construct the gradients

$$\bigtriangledown f = \begin{bmatrix} 6x_1 + 2x_2 + 1 \\ \\ 2x_1 + 2x_2 + 6 \end{bmatrix}, \quad \bigtriangledown q_1 = \begin{bmatrix} -2 \\ \\ -2 \end{bmatrix}, \bigtriangledown q_2 = \begin{bmatrix} -1 \\ \\ 0 \end{bmatrix}, \bigtriangledown q_3 = \begin{bmatrix} 0 \\ \\ -1 \end{bmatrix}.$$

Assuming all constraints are satisfied, set the gradient equal to zero to attempt to find an optimal.

$6x_1 + 2x_2 + 1 = 0$

$2x_1 + 2x_2 + 6 = 0$

$x_1 = \dfrac{5}{4}, \quad x_2 = \dfrac{-17}{4}.$

Make constraints q_1 *and* q_3, which are violated, active. Set both equal to zero.

$q_1 = -2x_1 - 3x_2 + 4 = 0$

$q_3 = -x_2 = 0.$

Solve the resulting system of equations.

$\bigtriangledown f + \mu_1 \bigtriangledown q_1 + \mu_3 \bigtriangledown q_3 = 0$

$q_1 = 0$

$q_3 = 0$

$$\begin{bmatrix} 6x_1 + 2x_2 + 1 \\ \\ 2x_1 + 2x_2 + 6 \end{bmatrix} + \mu_1 \begin{bmatrix} -2 \\ \\ -2 \end{bmatrix} + \mu_3 \begin{bmatrix} 0 \\ \\ -1 \end{bmatrix}$$

$-2x_1 - 2x_2 + 4 = 0$

$-x_2 = 0.$

One of the quickest methods of solving this problem is to formulate the system as a single matrix and then to reduce the rows. Inferences that :

$x_1 = 0.5,\ x_2 = 1,\ \mu_1 = 3.$

Which implies an objective function value of $f = 11.25$

Verify linear dependence of the gradient

$$\bigtriangledown f + \mu_1 \bigtriangledown q_1$$

$$\begin{bmatrix} 6 \\ \\ 9 \end{bmatrix} + 3 \begin{bmatrix} -2 \\ \\ -3 \end{bmatrix} = \begin{bmatrix} 0 \\ \\ 0 \end{bmatrix}.$$

Verify the uniqueness of the solution:

$$\bigtriangledown^2 f(x) = \begin{bmatrix} 6 & 2 \\ \\ 2 & 2 \end{bmatrix} \rightarrow \begin{bmatrix} 6-\lambda & 2 \\ \\ 2 & 1-\lambda \end{bmatrix} = 0 \rightarrow \lambda = 0.298,\ 6.702.$$

The Hessian matrix has a positive determinant due to its two positive eigenvalues, so this local minimum represents the global minimum of the objective function under the given constraints.

3.1.2 Euler's Method

Euler's Method is one of the easiest and oldest ways to approximate solutions to differential equations that can't be solved with a nice formula. Euler's Method is also known as the tangent line method [61]. At its core, it is an algorithm for plotting an approximate solution to an initial value problem through the direction field. It can also be used to get an approximate of the value of that solution for a given value of x.
The a problem with an initial value of the form $\frac{dy}{dx} = f(x, y)$ with $y(x_0) = y_0$. Then be able to find a formula for the solution $y = \Psi(x)$, the know that it exists if $f(x, y)$ is a continuous function. So, the about the solution $y = \Psi(x)$ and what it looks like. Our goal is to say as much as the about this solution, even though we may not be able to write it down as a formula.
The general first order differential equation

$$\frac{dy}{dx} = f(x, y) \tag{3.4}$$

With the initial condition $y(x_0) = y_0$
In this method the solution is in the form of tabulated values.
Integrating both sides of the equation (3,3) we get

$$y_1 = y_0 + \int_{x_0}^{x_1} f(x, y)dx \tag{3.5}$$

Assuming that $f(x, y) = f(x_0, y_0)$ $x_0 \leq x \leq x_1$, this gives Euler's formula

$$y_1 \approx y_0 + hf(x_0, y_0) \tag{3.6}$$

In general formula

$$y_{n+1} = y_n + hf(x_n, y_n), \quad n = 0, 1, 2, \ldots \tag{3.7}$$

Error approximate for the method

$$e_0 = 0$$

$$e_1 = L_1$$

$$e_{n+1} = e_n(1 + hf_y(x_n, \varepsilon_n)) + L_{n+1}$$

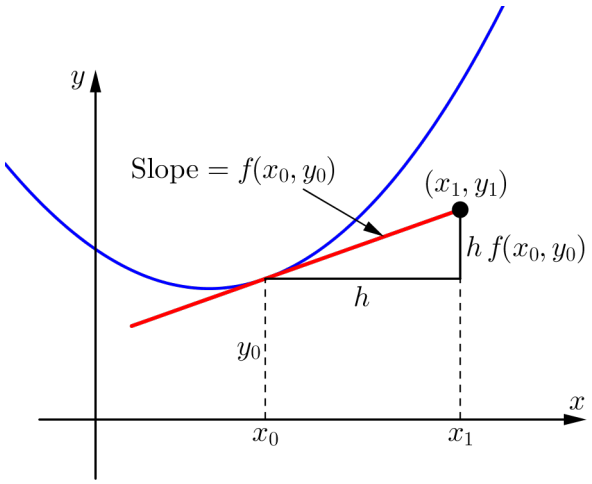


Figure 3.1: Euler's Method

Example 3.1.3. Use Euler's procedure to find $y(0.4)$ from the differential equation $\frac{dy}{dx} = xy, \quad y(0) = 1 \quad with \quad h = 0.1.$

Solution :

$$\begin{aligned}
\text{Given equation } \frac{dy}{dx} &= xy \\
Here\ f(x,y) &= xy \ \ and \ \ x_0 = 0,\ y_0 = 1,\ h = 0.1 \\
So\ that\ x_0 &= 0,\ x_1 = 0.1,\ x_2 = 0.2,\ x_3 = 0.3\ and\ x_4 = 0.4
\end{aligned}$$

By Euler's formula

$$\begin{aligned}
y_{n+1} &= y_n + hf(x_n, y_n),\ n = 0, 1, 2, \ldots \\
For\ n = 0 & \\
y_1 &= y_0 + hf(x_0, y_0) \\
y_1 &= 1 + 0.1(0 \times 1)
\end{aligned}$$

$$
\begin{aligned}
y_1 &= 1 \\
n = 1 & \\
y_2 &= y_1 + hf(x_1, y_1) \\
y_2 &= 1 + 0.1(0.1 \times 1) \\
y_2 &= 1.01 \\
n = 2 & \\
y_3 &= y_2 + hf(x_2, y_2) \\
y_3 &= 1.01 + 0.1(0.2 \times 1.01) \\
y_3 &= 1.0302 \\
n = 3 & \\
y_4 &= y_3 + hf(x_3, y_3) \\
y_4 &= 1.0302 + 0.1(0.3 \times 1.0302) \\
y_4 &= 11.061106 \\
\text{Hence } y(0.4) &= 11.061106.
\end{aligned}
$$

3.1.3 Euler-Lagrange Equations

A variation problem with fixed end points is the simplest variation problem in calculus [7]. On the other hand, the simplest challenge in optimal control is to find a free value for the state variables at the appropriate endpoint (terminal time):

$$
\begin{cases}
\textit{Minimiz :} & \displaystyle\int_{\tau_0}^{\tau_f} \gamma(\tau, \chi(\tau), u(\tau)) d\tau \\
\textit{Subject to :} & \chi'(\tau) = \rho(\tau, \chi(\tau), u(\tau); \quad \chi(t_0) = \chi_0
\end{cases}
\tag{3.8}
$$

with fixed initial time τ_0 and terminal time τ_f. A control function $u(\tau), \tau_0 \leq \tau \leq \tau_f$ together with the initial value problem eq (3.4), determines the response $\chi(\tau), \tau_0 \leq \tau \leq \tau_f$ (provided it exists) [1]. Thus they of finding a control, since the corresponding response

is implied.

To develop first-order necessary conditions in problems of the (Euler's equation), the constructed a one-parameter family of comparison trajectories $\chi(\tau) + \delta\xi(\tau)$, with $\xi \in C^1[\tau_0, \tau_f]^{n_\chi}$ such that $\| \xi \|_{1,\infty} \leq \eta$. However, when it comes to optimal control problems such as eq (3.4), a variation of the state trajectory χ cannot be explicitely related to a variation of the control u, in general, because the state and control variables are implicitly related by the (nonlinear) differential equation eq (3.4). Instead, the consider a one-parameter family of comparison trajectories $u(\tau) + \delta\varphi(\tau)$, with $\varphi \in C^1[\tau_0, \tau_f]^{n_\chi}$ such that $\| \varphi \|_{1,\infty} \leq \eta$.

Example 3.1.4. Consider the optimal control problem

$$\begin{cases} \textit{Minimiz :} \quad \varphi(u) = \displaystyle\int_0^1 [\frac{1}{2}u(\tau)^2 - \chi(\tau)]d\tau \\ \textit{Subject to :} \ \dot{\chi}(\tau) = 2[1 - u(\tau)]; \quad x(0) = 1 \end{cases} \tag{3.9}$$

To find candidate optimal controls for the problem eq (3.5), we start by forming the Hamiltonian function

$$H(\chi, u, \lambda) = \frac{1}{2}u^2 - \chi + 2\lambda(1 - u).$$

Candidate solutions (u^*, χ^*, λ^*) are those satisfying the Euler-Lagrange equations, i.e.,

$$\begin{aligned} \dot{\chi^*}(\tau) &= H_\lambda = 2[1 - u^*]; \qquad \chi^* = 1 \\ \dot{\lambda^*}(\tau) &= -H_\chi = 1; \qquad\qquad \lambda^* = 0 \\ 0 = H_u &= u^*(\tau) - 2\lambda^*(\tau). \end{aligned}$$

The adjoint equation trivially yields

$$\lambda^*(\tau) = \tau - 1,$$

and from the optimality condition, we get

$$u^*(\tau) = 2(\tau - 1).$$

(Note that u^* is indeed a candidate minimum solution for the problem since $H_{uu} = 1 > 0$ for each $0 \leq \tau \leq 1$.) Finally, substituting the optimal control candidate back into (3.5) yields

$$\chi'^{\,*} = 6 - 4\tau; \quad \chi^* = 1.$$

Integrating the latter equation, and drawing the results together, we obtain

$$\begin{aligned} u^*(\tau) &= 2(\tau - 1) \\ x^*(\tau) &= -2\tau^2 + 6\tau + 1 \\ \lambda^*(\tau) &= \tau - 1 \end{aligned}$$

It is also readily verified that H is constant along the optimal trajectory,

$$H(\tau, \chi^*(\tau), u^*(\tau), \lambda^*(\tau)) = -5.$$

Finally, we illustrate the optimality of the control u^* by considering the modified controls $\nu(\tau;\delta) := u^* + \delta\omega(\tau)$, and their associated responses $\mu(\tau;\delta)$). The perturbed cost function reads:

$$\begin{cases} \varphi(\nu(\tau;\delta)) := \displaystyle\int_0^1 \frac{1}{2}[u^*(\tau) + \delta\omega((\tau)]^2 - \mu(\tau;\delta)]d\tau \\ s.t. \;\; \mu' = 2[1 - u^*(\tau) - \delta\omega(\tau)]; \;\; \chi(0) = 1. \end{cases}$$

The cost function $\varphi(\upsilon(\tau;\delta))$ is represented in Fig. 3.2. for different perturbations $\omega(\tau) = \tau^k, \; k = 0, \ldots, 4$. Note that the minimum of $\varphi(\nu(\tau;\delta))$ is always attained at $\delta = 0$.

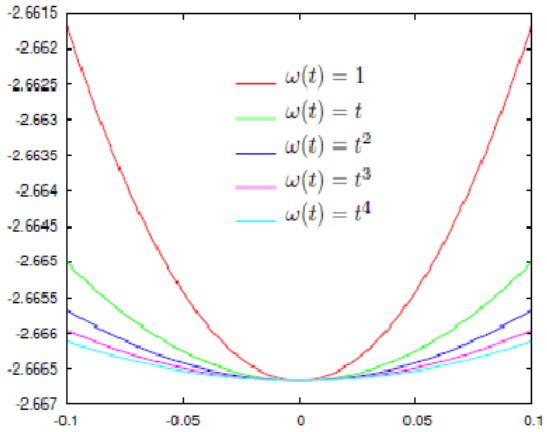


Figure 3.2: Function $\varphi(\nu(\tau, \delta))$ for Various Perturbations $\omega(\tau) = \tau^k$

3.1.4 Direct Methods

Direct optimization approaches have been widely investigated in order to avoid the shortcomings of indirect methods and have proven to be effective instruments for solving actual optimal control problems. Direct optimization approaches are based on the premise that the control problem can be discretized, and then NLP techniques can be used to the resultant n-dimensional problem. There are no adjoint variables in these approaches, for this reason, only variables associated with control or state are employed in optimization [71]. Adjoint variables can also be calculated from the Lagrange multipliers of the nonlinear optimization problem. As a further advantage, they may be easily applied to DAE systems. As is well known, the discretization of the control parameters results in poor solutions when using direct approaches.

1-Direct Sequential Methods

2-Direct Simultaneous Methods

3.1.5 Indirect Methods

The first indirect technique was the calculus of variations, which dates back to the 17th century and Johan Bernoulli's invitation to solve a novel problem involving determining the time-optimal path of a particle in a gravitational field between any two

sites. Instead of explicitly minimizing a cost criteria as in direct methods, an optimal solution is determined by meeting optimality conditions in indirect methods [64]. The optimality conditions result in either a two-point or a multi-point boundary value problem, depending on the supplied optimal control problem. It comprises of an algebraic equation for the continuous control and differential equations for the continuous state and adjacent variables.

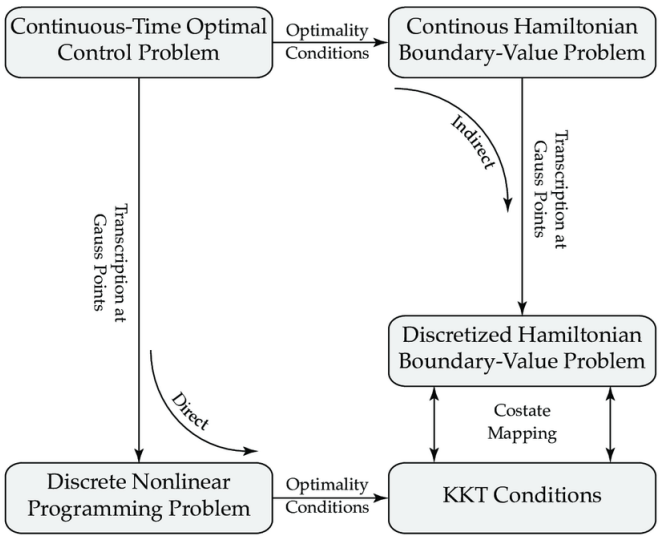


Figure 3.3: Direct and Indirect Methods

3.2 Applications of Dynamic and Control Problems

In this section we review some of the important applications and techniques that made optimal control very important, the tool to improve our working lives.

3.2.1 Optimization Techniques and Applications

Application performance optimization uses a number of techniques that are built into an organization's IT infrastructure to improve the way a network works. These

techniques include monitoring bandwidth capacity, application protocols, overall network traffic, application coding, network latency, potential attacks on software vulnerabilities, and more [66]. Even though each of these methods helps a network's application delivery work better, the best way to keep delivery methods working well is to use a proactive monitoring system to track and measure end-to-end performance metrics. Businesses can get the most out of their content networks by keeping a close eye on how they work and dealing with possible threats before they cause problems. This will make it possible for delivery paths to support the health of an entire IT system and keep all business-critical applications running well.

3.2.2 Optimization with Differential Equations

Differential equations have uses in both the classroom and the real world. A relationship between two quantities, two functions, two variables or a set of variables, or two functions is shown by an equation. A differential equation is a set of formulas that show the relationship between a function and its derivatives. There are a lot of examples, which shows how these equations are used [87]. The differential equation shows the relationship between two functions, each of which stands for a certain kind of operation. The derivative of an operation is the rate of change during that operation. The order of the degree is used to show these equations, such as first order, second order, etc. It is commonly used in engineering, physics, and other fields [14]. A pendulum or other swinging item is an example of a real-world application of ordinary differential equations for the calculation of electrical flow. Aside from that, they're employed in medical settings to track the progression of diseases graphically. Differential equations come particularly handy when modeling population expansion or radioactive decay models in mathematics. Simulation or numerical solution of an ordinary differential equation may be the only way to obtain the values of your objective function or nonlinear constraint function (ODE).

3.2.3 Optimal Control Applications

Optimal control theory is a branch of mathematical optimization that tries to find a way to control a dynamical system over time so that an objective function is maximized. It is used in science, engineering, and operations research in a lot of different ways [81]. For instance, the dynamical system could be a spaceship with controls like rocket thrusters, and the goal could be to get to the moon using the least amount of fuel. Or, the dynamic system could be a country's economy, and the goal could be to keep unemployment as low as possible. In this case, fiscal and monetary policy could be used to control the economy. Operations research problems can also be put into the framework of optimal control theory with the help of a dynamical system. Optimal control is a mathematical method for finding control policies that builds on the calculus of variations. The method owes a lot to the work of Lev Pontryagin and Richard Bellman in the 1950s, which built on the work of Edward J. McShane on the calculus of variations. In control theory, optimal control can be seen as a way to keep things under control. The technique of minimizing a performance index by selecting control and state trajectories for a dynamic system is known as optimal control.There have been several applications of optimal control and its consequences in a wide range of industries, including aerospace, process control, robotics, bioengineering, and economics, finance, and management science. Only simple optimum control issues could be handled before the emergence of the digital computer in the 1950s. Optimal control theory and methods may now be applied to a broader range of situations thanks to the advent of digital computers. Optimizing a dynamical system's objective function is the goal of optimal control theory. The additional optimization problems includes three main parts as follows:

1- Optimal Control with Integral Objective.

2- Optimal Control with Economic Objective.

3- Optimal Control: Minimize Final Time.

3.2.4 Dynamic Optimization with Applications

Optimization models that are dynamic (or multistage) Scheduling, route planning, solving optimal game strategies and inventory control are just a few examples of real-world problems that may be solved using these algorithms. Other examples include DNA sequence matching, stopping problems, airplane rocket flight path optimization and so on [56]. Dynamic optimization approaches can be applied to mobile applications in discrete or continuous time. Only those who understand dynamic programming and optimal control may apply these techniques. Decisions are made using differential and algebraic equation mathematical models to forecast future outcomes in order to devise wise policies. This form of analysis can make use of a wide range of tools and methodologies. The optimal rocket time issue is an example of this type of model. The goal is to decrease the flight time t of a rocket so that the rocket reaches its final destination with zero velocity.

3.2.5 Applications Energy Optimization Problems

The term (energy optimization) refers to the process of maximizing one's use of energy in one's dwelling. Increasing energy efficiency (and savings) in already occupied spaces is an integral part of this strategy, which goes beyond merely reducing energy consumption. Which can aid in the system's efficiency by preventing excessive use [25]. Energy efficiency has many potential benefits for homeowners, including financial savings, reduced impact on the environment, and even better mental and physical health. Energy efficiency has grown in importance as a selling point for houses. Whether one's reasons for wanting to reduce energy consumption are monetary, ecological, or otherwise, everyone stands to gain from adopting more efficient practices. Using less energy is a good way to cut down on the carbon footprint. Residential electricity rates change with the seasons and every year, but that have steadily gone up over the last ten years. This is a trend that is likely to keep going. In addition to lowering that monthly electricity bills, making your home more energy efficient also helps protect you from the financial effects of sharp increases

in energy prices that could happen in the future. Many countries depend on energy to achieve social development and produce enormous economic power. Countries also set out to examine the long-term energy economy. For example, coal mining, oil extraction and the provision of petrochemical products have achieved considerable progress. From an energy point of view, energy not only promotes societal growth, but can also derive and spread various economic forms. Currently, the energy economy helps create jobs while encouraging the development of autos, construction, and other industries.Energy is a critical component of the modern economy. In assessing the economic level, production capacity is somewhat determined by the energy economic position. There must be a focus on efficiency in the mining and processing of energy resources. Energy storage can be used to capture excess generation and satisfy the fluctuating demand while maintaining the base-load generator's production constant.

3.3 Application of Reliability Optimization

Reliability optimization achieving the highest possible level of reliability within the limits imposed. It is possible to create and manufacture a highly reliable product without limits such as budget, time to market, customer expectations, product functional capabilities, and product weight. Making well-informed decisions necessitates an understanding of the relationship between marginal cost and reliability improvement. Enhancing dependability has various financial advantages. A decreasing return occurs when dependability performance is above the customer's expected level. And the cost is immediately noticeable. Finding the perfect balance between the situation, technology, and clients is the goal of reliability optimization [24].

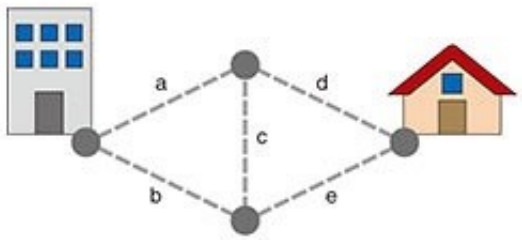
Example 3.3.1. Efficient optimization with binary decision diagrams We propose an efficient algorithm for finding the optimal solutions of budget constrained reliability maximization problems.This algorithm can find optimal solutions for communication networks with more than 100 nodes. The key to our algorithm is that it uses the data

structure called binary decision diagrams (BDDs). If a communication network is judged to be reliable enough, we can keep using it without making any changes. The reliability isn't good enough, steps need to be taken to fix it. Adding more links to the network is a common way to improve its reliability, as this is always the case. The task of finding the best way to add links to make a network more reliable can be set up as a combinatorial optimization problem called the network reliability maximization problem. This problem can be used to find the best way to add links. In what comes next, we make the reasonable assumption that the total budget for adding links to the network is limited and that we want to get the most out of the network's reliability within this budget. This problem is called the budget-constrained network reliability maximization problem. This optimization problem takes as input a communication network with nodes and links, the total budget, and a list of possible places to add new links. The assume that the cost of adding a link to each candidate position and the chance that the link will fail are known ahead of time. The answer to the problem is the topology of the communication network that gets the highest score for network reliability and makes sure that the total cost of adding links doesn't go over the budget. An example of a budget-constrained network reliability maximization problem and a set of possible solutions to the problem. The best candidate solution, is the one in the middle. This is the communication network topology that provides the most reliability while staying within the budget [53].

If a communication network is judged to be reliable enough, we can keep using it without making any changes. If, on the other hand, the reliability isn't good enough, steps need to be taken to fix it. Adding more links to the network is a common way to improve its reliability, as this is always the case. The task of finding the best way to add links to make a network more reliable can be set up as a combinatorial optimization problem called the network reliability maximization problem. This problem can be used to find the best way to add links. In what comes next, we'll show the By solving the network reliability maximization problem, we've seen that we can make a communication network

that works well even when things go wrong. But this problem is known to be hard to solve with computers. Even with the most powerful computers available today, it takes too long to solve.

A simple way to solve the problem of maximizing network reliability is to first list all of the possible network topologies that can be made while staying within the budget and then evaluate the reliability of each candidate. But there are two possible problems with this simple approach. First, the size of a network may cause the number of candidate topologies that meet a constraint to grow at an exponential rate. Second, it takes an exponential amount of time to figure out how reliable a possible solution is. To figure out how reliable a network is, we need to list all the ways a link could fail and still work with the network. Since the number of these failure patterns grows in an exponential way as the size of the network does, so does the amount of time it takes to do the math. Because of these two problems, previous methods could only find the best solutions for very small networks, with less than 20 nodes.



Link	Fail probability	Cost
a	5%	2
b	8%	3
c	9%	3
d	7%	3
e	6%	1

Figure 3.4: Budget Constrained Network Reliability Maximization Problem

CHAPTER 4

THEORETICAL CONVERGENCE PROPERTIES AND NUMERICAL RESULTS

Optimal control theory has been developed over the course of more than three hundred years as an extension of the numerical optimization. Mathematicians have built the theoretical groundwork for optimal control, which now has a wide range of applications in a wide range of scientific fields, from mathematics to engineering to biological and management sciences. This extension can be understood most clearly by looking at the basic problem, which is to select a continuously differentiable function $x(\tau),\ \tau_1 \leq \tau \leq \tau_2$, to solve.

$$\begin{cases} \textit{Minimiz :} & \displaystyle\int_{\tau_1}^{\tau_2} \eta(t, \chi(t), u(\tau))\ d\tau \\ \textit{Subject to :} & \chi(\tau_1) = \chi_1 \end{cases} \tag{4.1}$$

Where χ is the state, and u is the control input taking values in some control set. There are three initial states in a system: χ_0 is the initial state, τ_0 is the initial time. Although both χ and u are functions of t, their time arguments are commonly omitted from the equations.

4.1 Convergence Theoretical Result

Let $h : \mathbb{R}^n \mapsto (-\infty, \infty]$ and $k_z : \mathbb{R}^n \mapsto (-\infty, \infty]$, $z = 1, \ldots, m$, be closed, proper, convex functions in $\mathbb{R}^n$, the n-dimensional Euclidean space [55]. Consider the following convex program associated with h and the k_z $'s$:

(P)

$$\begin{cases} \textit{Minimiz :} & h(\chi) \\ \textit{Subject to :} & k_z(\chi) \leq 0, \quad z = 1, \ldots, m \end{cases} \tag{4.2}$$

The make the following standing assumption about (P):

1-The optimal solution set for (P) is nonempty and bounded.

2-The effective domain of h, the set $\{\chi | h(\chi) < \infty\}$ is contained in the effective domain $\{\chi | k_z(\chi) < \infty\}$ of each k_z. Furthermore, the relative interior of the effective domain of h is contained in the relative interior of the effective domain of each k_z.

4.2 Theoretical of Convergence Properties

Theorem 4.2.1. *(First − Order Necessary Conditions)*

Consider the problem to minimize the functional

$$\begin{cases} \textit{Minimiz :} & \varphi(u) = \int_{\tau_0}^{\tau_f} \eta(\tau, \chi(\tau), u(\tau))\ d\tau \\ \textit{subject to :} & \dot{\chi}(\tau) = \rho(\tau, \chi(\tau), u(t)); \qquad \chi(\tau_0) = \chi_0 \end{cases} \tag{4.3}$$

for $u \in C[\tau_0, \tau_f]^{n_u}$, with fixed endpoints $\tau_0 < \tau_f$ where η and ρ are continuous in (τ, χ, u) and have continuous first partial derivatives with respect to χ and u for all $(\tau, \chi, u) \in [\tau_0, \tau_f] \times \mathbb{R}^{n_\chi} \times \mathbb{R}^{n_u}$. Suppose that $u^* \in C[\tau_0, \tau_f]^{n_u}$ is a (local) minimizer for the problem,

and let $\chi^* \in C^1[\tau_0, \tau_f]^{n_\chi}$ *denote the corresponding response [17]. Then, there is a vector function* $\lambda^* \in C^1[\tau_0, \tau_f]^{n_\chi}$ *such that the triple* (u^*, χ^*, λ^*) *satisfies the system*

$$\dot{\chi} = \rho(\tau, \chi(\tau), u(\tau); \qquad \chi(\tau_0) = \chi_0 \tag{4.4}$$

$$\lambda^* = -\eta_\chi(\tau, \chi(\tau), u(\tau)) - \rho_\chi(\tau, \chi(\tau), u(\tau))^T \lambda^*(\tau); \qquad \lambda(\tau_f) = 0 \tag{4.5}$$

$$0 = \eta_\chi(\tau, \chi(\tau), u(\tau)) + \rho_\chi(\tau, \chi(\tau), u(\tau))^T \lambda^*(\tau); \tag{4.6}$$

for to $\tau_0 \leq \tau \leq \tau_f$. *These equations are known collectively as the Euler-Lagrange equations, and eq* (4.3) *is often referred to as the adjoint equation (or the costate equation).*

Theorem 4.2.2. *If* $z(\chi)$ *is an extremizing function for the problem*

$$Minimiz/Maximiz \;\; L(z) = \int_{\chi_1}^{\chi_2} F(\chi, z, z')\, d\chi \tag{4.7}$$

with end conditions $z(\chi_1) = z_1$ *and* $z(\chi_2) = z_2$ *then* $z(\chi)$ *satisfies the BVP.*

$$\frac{d}{d\chi}\left(\frac{\partial F}{\partial z'}\right) - \frac{\partial F}{\partial z} = 0 \tag{4.8}$$

$$z(\chi_1) = z_1 \quad \text{and} \quad z(\chi_2) = z_2 \tag{4.9}$$

Equation (4.7) is known as the Euler-Lagrange equation [3].

4.3 NUMERICAL RESULTS

In this section, the will present the fundamental software using the Python program with the Gekko library when applied to the proposed models.

4.3.1 Gekko Python

The wide variety of optimization and optimum control software packages and modeling languages available today. A general-purpose, high-level, interpreted programming language, Python. To make the code easier to read, it uses extensive indentation throughout the code [9]. Python is a garbage-collected, dynamically-typed programming language. It can be used in a variety of programming styles, including procedural, object oriented, and functional. It is frequently referred to as a "batteries included" language because of its extensive standard library. It was released as Python 0.9.0 in 1991. List comprehensions, cycle-detecting garbage collection, reference counting, and Unicode support were all part of Python 2.0, which was published in 2000. As of Python 3.0, which was published in 2008, it is not backwards compatible with previous versions. Python 2 was deprecated in 2020 with the release of version 2.7.18. Python is one of the most widely used programming languages in the world. Python was used with the Gekko package to get numerical results for all discrete finite, unrestricted, and persistent problems of all kinds. Linear and nonlinear optimization problems such as linear quadratic programming and integer quadratic programming can also be solved. Optimizing approaches can be used to develop optimal solutions, perform tradeoff assessments, and balance different design choices on the platform [34]. Modeling, analytical tools, and visualization of simulation and optimization are all included in a single Python module called Gekko. It allows for the use of advanced models and solvers because it is a differential and algebraic modeling language. Using Gekko, the model may be written in an easy-to-understand way. It accepts a model that has limitations and a goal to achieve.

4.4 Optimal Control Models

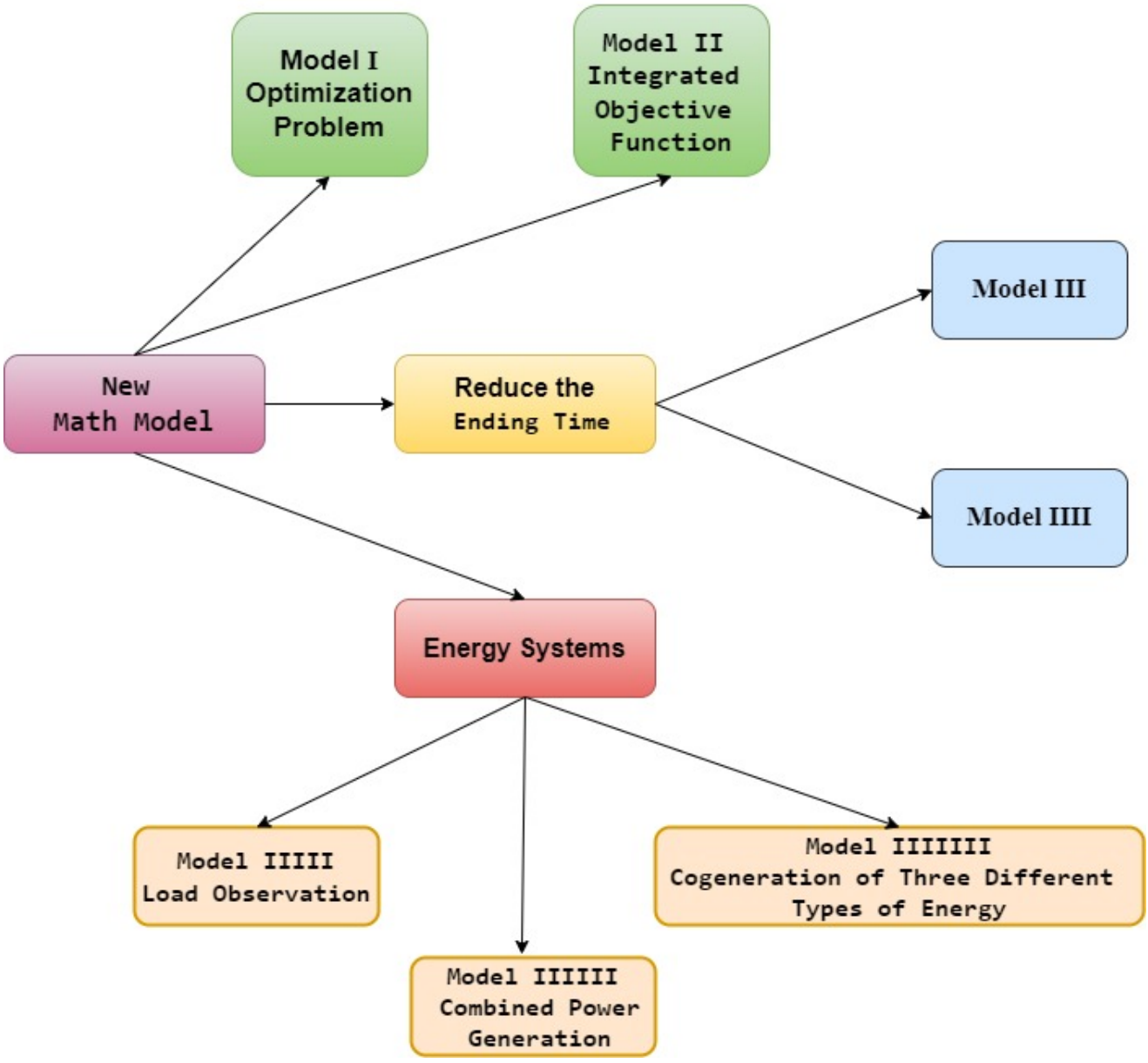


Figure 4.1: New Math Model

4.4.1 Model I : (Dynamic of Control Optimization Problem)

The new optimization models is set up as described in eq: (4.10) which consists of the objective function. Which is subject to constraints according to the nature of the problem in addition to the boundary conditions of the cases, and thus the constraint on the case x. The problem is a classical minimum energy optimal control problem. The formulation is the following:

$$\begin{cases} \textit{Minimize :} & g = \frac{1}{2} \int_0^1 (a(t) + \frac{1}{2})^2 dt \\ \textit{Subject to :} & \frac{dx(t)}{dt} = s(t) \\ & \frac{ds(t)}{dt} = a(t) \\ & x(0) = x(1) = 0 \\ & s(0) = -s(1) = 1 \\ & x(t) \leq k, \quad k = \frac{1}{11} \end{cases} \tag{4.10}$$

The parameter a (acceleration) is placed on the time horizon from a starting time of zero to a final time of one, where a represents the acceleration. The x and s variables denote the position and velocity, respectively.

This can be viewed as a friction less box traveling sideways with a speed of (1 m/s). After moving $\frac{1}{11}$ meters, the box must turn around and do the process over again. There must be no more than a one-second lag between when the box is thrown and when it returns, with the same initial velocity but in the opposite direction.

Algorithm 1 : Model I (Optimization Problem)

Choose initial values $x(0) = x(1) = 0,\ s(0) = -s(1) = 1,\ and\ let\ k = 0$

and go to the main step.

Main Step

1- Calculate the defect $x(t) \leq k$ in the boundary conditions. if $k = \frac{1}{11}$ stop;

2- Calculate the gradients

$$\frac{dx(t)}{dt} = s(t), \quad \frac{ds(t)}{dt} = a(t)$$

3- Solve the integral equation minimize

$$g = \frac{1}{2} \int_0^1 (a(t) + \frac{1}{2})^2 dt$$

4- Compute the new estimates $x^*(t),\ s^*(t),\ a^*(t)$

$$x_0^{k+1} = x_0^k + g_x^k$$

$$s_0^{k+1} = s_0^k + g_s^k$$

$$a_0^{k+1} = a_0^k + g_a^k$$

5- Compute the optimal solution (using quadratic method).

6- Replace

$$k \longleftarrow k + 1,\ and\ go\ to\ step\ 1.$$

Code 1:Use Ipop Solver to solve Constraint Optimization Problems

```
import numpy as np
import matplotlib.pyplot as plt
from gekko import GEKKO
m = GEKKO(remote=False)
nt = 101; m.time = np.linspace(0,1,nt)
# Variables
x = m.Var(value=0,ub=1/11)
s = m.Var(value=1)
a = m.Var(value=-6)
p = np.zeros(nt); p[-1] = 1.0
final = m.Param(value=p)
# Equations
m.Equation(x.dt()==s)
m.Equation(v.dt()==a)
# Final conditions
soft = True
if soft: # soft terminal constraint
m.Minimize(final*1e5*x**2) m.Minimize(final*1e5*(s+1)**2)
else: # hard terminal constraint
xf = m.Param();  sf = m.Param() m.free(xf);  m.free(af)
m.fix_final(xf,0); m.fix_final(sf,-1)
# connect endpoint parameters to x and s
m.Equations([xf==x,sf==s]) # Objective Function
obj = m.Intermediate(0.5*m.integral(a**2))
m.Minimize(final*obj) m.options.IMODE = 6 m.options.NODES = 2
m.options.SOLVER = 2  m.solve()
# Create a figure
plt.figure(figsize=(10,4))  plt.subplot(2,2,1)
plt.plot([0,1],[1/11,1/11],'r:',label=r'$x<\frac{1}{11}$')
plt.plot(m.time,x.value,'k-',lw=2,label=r'$x$')
plt.ylabel('Position')      plt.legend(loc='best')
plt.subplot(2,2,2)
plt.plot(m.time,s.value,'b--',lw=2,label=r'$s$')
plt.ylabel('Velocity')   plt.legend(loc='best')
plt.subplot(2,2,3)
plt.plot(m.time,a.value,'r--',lw=2,label=r'$a$')
plt.ylabel('Thrust') plt.legend(loc='best')
plt.xlabel('Time')
plt.subplot(2,2,4)
plt.plot(m.time,obj.value,
'g-',lw=2,label=r'$\frac{1}{2} \int a^2$')
plt.text(0.5,3.0,'Final Value =
'+str(np.round(obj.value[-1],2)))
plt.ylabel('Objective') plt.legend(loc='best')
plt.xlabel('Time') plt.show()
```

Table 4.1: Numerical results of Model I (Optimization Problem)

Iter	Objective	Convergence
0	4.03130E+05	3.72500E+03
1	1.75743E+05	9.60942E+02
2	9.62023E+04	9.43700E+02
3	-5.86329E+02	2.95236E+01
4	-5.88970E+02	2.21117E+00
5	-5.89174E+02	7.00696E-02
6	3.22054E+02	3.18037E-02
7	4.01824E+00	7.23981E-03
8	4.01774E+00	2.24036E-03
9	4.01760E+00	8.06080E-04
10	4.01765E+00	1.32170E-03
11	4.01769E+00	1.73065E-03
12	4.01756E+00	3.48841E-04
13	4.01753E+00	5.23779E-05
14	4.01752E+00	6.89881E-06
15	4.01951E+00	1.98893E-07

Solution time : 0.2064 sec

Objective : 4.028011909766749

The problem by (Ipop Solver) and the solution succeeded, as the solution was completed within a quarter of a second and we can see the goal function is equal to 4.028011909766749. According to the results, we notice that the iteration took 15 iterations, as we note that the values of goal convergence are confirmed in the last five iterations.

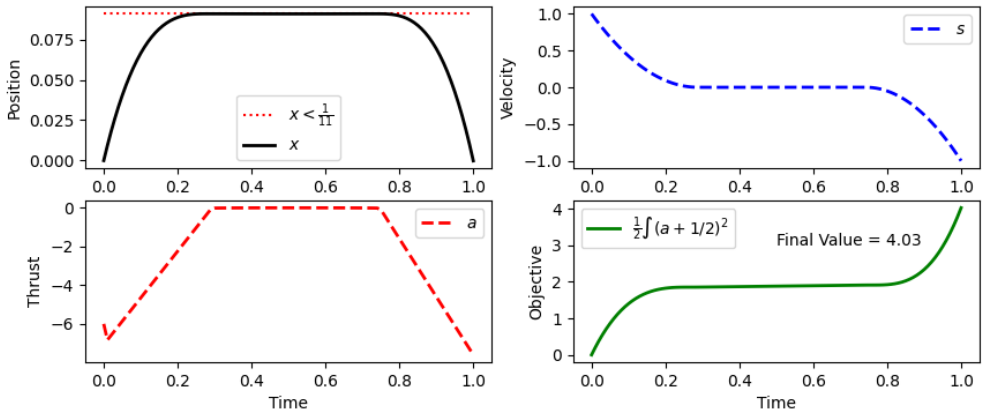


Figure 4.2: Model I (Optimization Problem)

By noticing the figures that appeared, the red dashed line represents the acceleration, and the green line represents the objective value of the function (the final value). The black line is for speed, and the speed is for the blue line. And the result will be for us a logical solution of the equations that show, so we try to reduce the value that appeared to the lowest possible level, as the value of a goes to zero so it does not change much and therefore is able to meet the results within the specified constraints. And that the values of x start from zero, as well as the speed that started at one, so we had to find a way to reach the last point with the minimum value of a. Where we see that the restriction has not been violated, and we ensure that other restrictions have not been violated.

4.4.2 Model II : (Optimal Control of An Integrated Objective Function)

The created a program to track our progress and show the results by assume that two minutes is the estimated time. The optimal control problem has an integral objective function, which means it has a single objective function. In many systems, integrals are a naturally occurring expression of how to keep the accumulation of a quantity to an absolute minimum. The formulation is the following:

$$\begin{cases} \textit{Minimize :} & \frac{1}{2}\int_0^2 \exp(2\ln(z_1))dt \\ \textit{subject to :} & \frac{dz_1(t)}{dt} = v \\ & z_1(0) = 1 \\ & -1 \le v(t) \le 1 \end{cases} \tag{4.11}$$

Integral expressions are reformulated in differential and algebraic form by defining a new variable.

$$z_2 = \frac{1}{2}\int_0^2 \exp(2\ln(z_1))dt \tag{4.12}$$

The new variable and integral are differentiated and included as an additional equation. The problem then becomes a minimization of the new variable z_2 at the final time.

$$\begin{cases} \underset{v}{Minimize} & z_2(t_f) \\ \textit{subject to :} & \frac{dz_1(t)}{dt} = v \\ & \frac{dz_2(t)}{dt} = \frac{1}{2}z_1^2(t) \\ & z_1(0) = 1 \\ & z_2(0) = 0, \quad t_f = 2 \\ & -1 \le v(t) \le 1 \end{cases} \tag{4.13}$$

The optimal control problem has an objective function that is inseparable from it. When the accumulation of a given quantity is either maximizing or minimizing, integrals are a natural expression. Assuming the integral starts at zero, the initial condition becomes the integral between 0 and 2. This is the point at which the optimal control problem's time horizon ends.

Algorithm 2 : Model II to Optimal Control of An Integrated Objective Function

---

Choose initial values $z_1(0) = 1,\ z_2(0) = 0,\ t_f = 2,\ 1 \leq v(t) \leq 1 and\ let\ k = 0$ and go to the main step.

Main Step

1- Calculate $$z_2 = \frac{1}{2}\int_0^2 z_1^2(t)dt$$

2- Calculate the gradients

$$\frac{dz_2(t)}{dt} = \frac{1}{2}z_1^2(t)$$

$$\frac{dz_1(t)}{dt} = v$$

3- Solve the integral equation minimize $\omega = z_2(t_f)$

4- Compute the new estimates

$$z_1^{k+1} = z_1^k + \omega_{z_1}^k$$
$$z_2^{k+1} = z_2^k + \omega_{z_2}^k$$
$$v_0^{k+1} = v_0^k + \omega_v^k$$

5- Compute the optimal solution (using KKT Method).

6- Replace $k \longleftarrow k+1,\ and\ go\ to\ step\ 1.$

Code 2:Use Ipop Solver to Solve Model II to Optimal Control of An Integrated Objective Function

```
import numpy as np
import matplotlib.pyplot as plt
from gekko import GEKKO
m = GEKKO()
nt = 101
m.time = np.linspace(0,2,nt)
# Variables
z1 = m.Var(value=1)
z2 = m.Var(value=0)
v = m.Var(value=0,lb=-1,vb=1)
p = np.zeros(nt)
p[-1] = 1.0
final = m.Param(value=p)
# Equations
m.Equation(z1.dt()==v)
m.Equation(z2.dt()==0.5*z1**2)
# Objective Function
m.Obj(z2*final)
m.options.IMODE = 6
m.solve()
# m.solve(remote=False)  # for local solve
plt.figure(1)
plt.plot(m.time,z1.value,'k-',lw=2,label=r'$z_1$')
plt.plot(m.time,z2.value,'b-',lw=2,label=r'$z_2$')
plt.plot(m.time,v.value,'r--',lw=2,label=r'$v$')
plt.legend(loc='best')
plt.xlabel('Time')
plt.ylabel('Value')
plt.show()
```

Table 4.2: Optimal Control Problem

Iter	Objective	inf pr	inf du	lambda du	upsilon pr
0	0.0000000e+00	5.00e-01	8.29e-03	0.00e+00	0.00e+00
1	9.7317306e-01	2.00e-04	1.16e-03	9.71e-01	1.00e+00h
2	4.0119286e-01	9.02e-02	4.44e-16	1.00e+00	1.00e+00f
3	1.6310287e-01	8.82e-02	1.11e-16	1.00e+00	1.00e+00h
4	1.7184636e-01	1.55e-02	2.87e-06	9.99e-01	1.00e+00h
5	1.6547828e-01	5.45e-03	2.44e-04	8.60e-01	1.00e+00h
6	1.6298261e-01	1.30e-03	6.02e-05	9.17e-01	1.00e+00h
7	1.6196313e-01	4.12e-04	1.63e-05	9.28e-01	1.00e+00h
8	1.6175448e-01	1.10e-04	1.39e-07	9.98e-01	1.00e+00h
9	1.6170941e-01	3.05e-05	5.55e-17	1.00e+00	1.00e+00h
10	1.6170018e-01	6.54e-06	1.08e-07	9.69e-01	1.00e+00h
11	1.6170001e-01	8.70e-07	5.55e-17	1.00e+00	1.00e+00h

Table 4.3: Optimal Control of An Integrated Objective Function

Target	Scaled	Unscaled
Objective	1.6170001214895699e-01	1.6170001214895699e-01
Dual infeasibility	5.5511151231257827e-17	5.5511151231257827e-17
Constraint violation	8.7038250779763370e-07	8.7038250779763370e-07
Complementarity	2.7795446771618800e-08	2.7795446771618800e-08
Overall NLP error	1.7038250779763370e-07	2.7038250779763370e-07

Solution time : 7.450000000244472E-002 sec

Objective : 0.161700012148957

The final value of the objective function is 0.161700012148957.

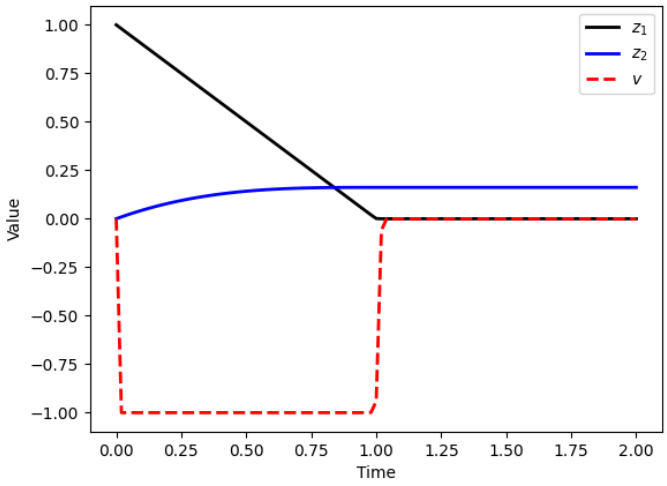


Figure 4.3: Optimal Control of An Integrated Objective Function

In this figure (4.3) we draw the values of z_1, z_2 for this method, implement it in python and then improve it. Also, we're trying to do is reduce $\int_0^2 z_1^2$, so the best way to do

this is to get the value of z_1 to zero as quickly as possible, so that $\int_1^2 z_1^2$ is good as low as possible. So we start in an initial state of one, but are constrained by a differential equation $\frac{dz_1}{dt} = v$ and move to its minimum value from zero to one time period, and the reason it will do that is because then it will have one negative slope which is the fastest it can go down. So we can get a zero slope and keep z_1 at zero, so this is the best solution for $\int_0^2 z_1^2$.

4.4.3 Model III: (Reduce the Ending Time)

The new mathematical model's optimal control problem is aimed at minimizing the total time. Manufacturing, transportation and energy systems are all concerned with minimizing the amount of time it takes to complete a given task to the bare minimum. Every situation has a desirable end-state, and the purpose of the optimizer is to get there in the quickest period of time possible [54].

$$\begin{cases} \underset{v(t)}{Minimize:} & t_f \\ Subject\ to: & \frac{dz_1(t)}{dt} = t_f v \\ & \frac{dz_2(t)}{dt} = t_f \cos(z_1^2(t)) \\ & \frac{dz_3(t)}{dt} = t_f \sin(z_1^2(t)) \\ & z(0) = [\pi/2, 4, 0] \\ & z_2(t_f) = 0 \\ & z_3(t_f) = 0 \\ & -2 \leq v(t) \leq 2 \end{cases} \tag{4.14}$$

A time horizon of 0.0 to 1.0 must be specified, together with an additional t_f parameter that scales the final time to one, to address the new math model problem. End-of-system conditions $z_2(t_f) = 0$ *and* $z_3(t_f) = 0$ prevent a final time that is lower than the conditions. A Feed forward Variable (Fixed Value) is declared for the

parameter t_f, which means that over the time horizon, it only has one value. A Manipulated Variable V is a variable whose value can be changed at any point in time.

Algorithm 3 : (Model III) For Reduce the Ending Time

Choose initial values $z(0) = [\pi/2, 4, 0],\ z_2(t_f) = 0,\ z_3(t_f) = 0,\ k = 0$

and go to the main step.

Main Step

1- Calculate the defect $\dfrac{dz_1(t)}{dt} = t_f v$ in the boundary conditions. if $-2 \leq v(t) \leq 2$ stop;

2- Calculate the gradients

$$\frac{dz_2(t)}{dt} = t_f \cos(z_1^2(t))$$

$$\frac{dz_3(t)}{dt} = t_f \sin(z_1^2(t))$$

3- Solve the integral equation $\min\limits_{u(t)} \quad : \omega = t_f$

4- Compute the new estimates

$$\begin{aligned} z_1^{k+1} &= z_1^k + \omega_{z_1}^k \\ z_2^{k+1} &= z_2^k + \omega_{z_2}^k \\ z_3^{k+1} &= z_3^k + \omega_{z_3}^k \\ v_0^{k+1} &= v_0^k + \omega_v^k \end{aligned}$$

5- Compute the optimal solution (using Euler's Method).

6- Replace $\quad k \longleftarrow k+1,$ *and go to step* 1.

An algorithm is a technique that accepts any input and changes it into the desired output. Sorting can be accomplished using a plethora of different algorithms.

Code 3:Use Ipop Solver to Solve (Model III) for Reduce the Ending Time

```
import numpy as np
from gekko import GEKKO
import matplotlib.pyplot as plt
m = GEKKO()
nt = 501
tm = np.linspace(0,1,nt)
m.time = tm
# Variables
z1 = m.Var(value=np.pi/2.0)
z2 = m.Var(value=4.0)
z3 = m.Var(value=0.0)
p = np.zeros(nt)
p[-1] = 1.0
final = m.Param(value=p)
# FV
tf = m.FV(value=1.0,lb=0.1,vb=100.0)
tf.STATUS = 1
# MV
v = m.MV(value=0,lb=-2,ub=2)
v.STATUS = 1
m.Equation(z1.dt()==u*tf)
m.Equation(z2.dt()==m.cos(z1)^2*tf)
m.Equation(z3.dt()==m.sin(z1)^2*tf)
m.Equation(z2*final<=0)
m.Equation(z3*final<=0)
m.Obj(tf)
m.options.IMODE = 6
m.solve()
print('Final Time: ' + str(tf.value[0]))
tm = tm * tf.value[0]
plt.figure(1)
plt.plot(tm,x1.value,'k-',lw=2,label=r'$z_1$')
plt.plot(tm,x2.value,'b-',lw=2,label=r'$z_2$')
plt.plot(tm,x3.value,'g--',lw=2,label=r'$z_3$')
plt.plot(tm,v.value,'r--',lw=2,label=r'$v$')
plt.legend(loc='best')
plt.xlabel('Time')
plt.ylabel('Value')
plt.show()
```

Table 4.4: Minimize Final Time

Iter	Objective	inf pr	inf du	lambda du	upsilon pr
0	5.0000259e+02	4.01e+00	1.00e+00	0.00e+00	0.00e+00
1	4.9998044e+021	4.00e+00	6.25e+01	2.26e-01	3.50e-03h
2	5.1083876e+02	3.97e+00	2.34e+02	2.38e-01	7.43e-03f
3	5.3684451e+02	3.87e+00	4.29e+02	3.34e-01	2.43e-02h
4	7.9884823e+02	2.97e+00	3.56e+02	3.71e-01	2.31e-01f
5	1.6547828e-01	5.45e-03	2.44e-04	8.60e-01	1.00e+00h
6	1.6298261e-01	1.30e-03	6.02e-05	9.17e-01	1.00e+00h
7	2.0036109e+03	3.21e-02	1.33e+01	8.42e-01	9.90e-01h
8	2.0038380e+03	1.84e-02	2.20e+02	8.30e-01	1.00e+00h
9	2.0036758e+03	8.07e-03	2.63e+03	7.92e-01	1.00e+00h
10	2.0036489e+03	2.70e-03	1.05e+03	7.67e-01	1.00e+00h
11	2.0036193e+03	7.57e-04	1.39e+03	7.75e-01	1.00e+00h
12	2.0035448e+03	2.39e-04	4.81e+01	9.81e-01	1.00e+00h
13	2.0034977e+03	2.34e-05	1.39e-05	1.00e+00	1.00e+00h
14	2.0034828e+03	4.03e-06	2.64e+01	1.00e+00	8.45e-01h
15	2.0034799e+03	1.65e-07	1.06e+00	1.00e+00	9.63e-01h
16	2.0034797e+03	2.86e-08	1.12e-01	1.00e+00	9.54e-01h
17	2.0034796e+03	8.08e-09	1.86e-02	1.00e+00	9.08e-01h
18	2.0034796e+03	3.45e-09	3.92e-03	1.00e+00	8.30e-01h
19	2.0034796e+03	1.52e-09	8.18e-05	1.00e+00	9.79e-01h
20	2.0034796e+03	3.42e-10	1.82e-10	1.00e+00	1.00e+00h

Table 4.5: (Model III) Reduce the Ending Time

Target	scaled	unscaled
Objective	4.0069592942508189e+02	2.0034796471254094e+03
Dual infeasibility	1.8157819579791169e-10	9.0789097898955843e-10
Constraint violation	3.4237856993968308e-10	3.4237856993968308e-10
Complementarity	7.1364389221260054e-11	3.5682194610630024e-10
Overall NLP error	3.4237856993968308e-10	9.0789097898955843e-10

Solution time : 1.33109999999942 sec

Objective : 2003.47964712541

Final Time : 4.0069591632

Any feasible solution of the optimization problem eq: (4.14) consists of the table above and a set of control policies that satisfy the input. For a table there is a meaningful control policy for the systems control policy. The results of the optimal solution obtained with the least time to solve, which is 1.33109999999942 sec, while the final time 4.0069591632.

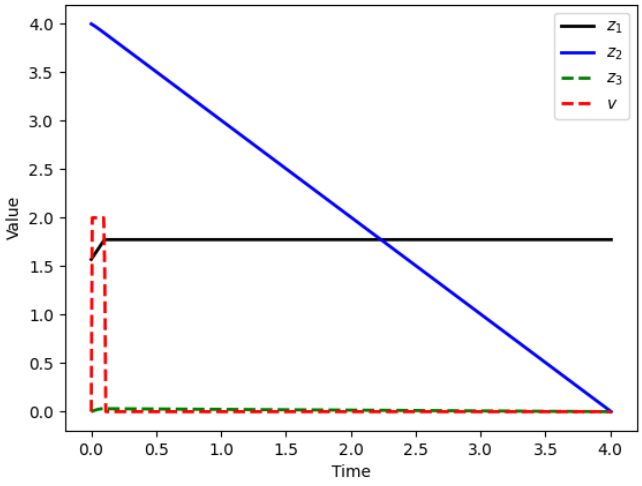


Figure 4.4: (Model III) Reduce the Ending Time

We will change the restrictions for the same problem to obtain more satisfactory results in less time than before ($\frac{dx_2}{dt} = t_f \cos(x_1^3(t)),\ \frac{dx_3}{dt} = t_f \sin(x_1^3(t))$.

$$
\begin{cases}
\underset{v(t)(t_f)}{Minimize:} & t_f \\
subject\ to: & \frac{dz_1(t)}{dt} = t_f v \\
& \frac{dz_2(t)}{dt} = t_f \cos(z_1^3(t)) \\
& \frac{dz_3(t)}{dt} = t_f \sin(z_1^3(t)) \\
& z_1(0) = [\pi/2, 4, 0] \\
& z_2(t_f) = 0 \\
& z_3(t_f) = 0 \\
& -2 \leq v(t) \leq 2
\end{cases}
\tag{4.15}
$$

Algorithm 4 : (Model IIII) Reduce the Ending Time

Choose initial values $z(0) = [\pi/2, 4, 0],\ z_2(t_f) = 0,\ z_3(t_f) = 0,\ k = 0$

and go to the main step.

Main Step

1- Calculate the defect $\dfrac{dz_1(t)}{dt} = t_f v$ in the boundary conditions. if $-2 \leq v(t) \leq 2$ stop;

2- Calculate the gradients

$$\frac{dz_2(t)}{dt} = t_f \cos(z_1^3(t))$$

$$\frac{dz_3(t)}{dt} = t_f \sin(z_1^3(t))$$

3- Solve the integral equation $\min\limits_{u(t)} : \omega = t_f$

4- Compute the new estimates

$$\begin{aligned} z_1^{k+1} &= z_1^k + \omega_{z_1}^k \\ z_2^{k+1} &= z_2^k + \omega_{z_2}^k \\ z_3^{k+1} &= z_3^k + \omega_{z_3}^k \\ v_0^{k+1} &= v_0^k + \omega_v^k \end{aligned}$$

5- Compute the optimal solution (using Euler-Lagrange Method).

6- Replace $k \longleftarrow k+1,$ *and go to step* 1.

Code 4:Use Ipop Solver to Solve (Model IIII) for Reduce the Ending Time

```
import numpy as np
from gekko import GEKKO
import matplotlib.pyplot as plt
m = GEKKO()
nt = 501
tm = np.linspace(0,1,nt)
m.time = tm
# Variables
z1 = m.Var(value=np.pi/2.0)
z2 = m.Var(value=4.0)
z3 = m.Var(value=0.0)
p = np.zeros(nt)
p[-1] = 1.0
final = m.Param(value=p)
# FV
tf = m.FV(value=1.0,lb=0.1,vb=100.0)
tf.STATUS = 1
# MV
v = m.MV(value=0,lb=-2,ub=2)
v.STATUS = 1
m.Equation(z1.dt()==u*tf)
m.Equation(z2.dt()==m.cos(z1)**3*tf)
m.Equation(z3.dt()==m.sin(z1)**2*tf)
m.Equation(z2*final<=0)
m.Equation(z3*final<=0)
m.Obj(tf)
m.options.IMODE = 6
m.solve()
print('Final Time: ' + str(tf.value[0]))
tm = tm * tf.value[0]
plt.figure(1)
plt.plot(tm,x1.value,'k-',lw=2,label=r'$z_1$')
plt.plot(tm,x2.value,'b-',lw=2,label=r'$z_2$')
plt.plot(tm,x3.value,'g--',lw=2,label=r'$z_3$')
plt.plot(tm,v.value,'r--',lw=2,label=r'$v$')
plt.legend(loc='best')
plt.xlabel('Time')
plt.ylabel('Value')
plt.show()
```

Table 4.6: Minimize Final Time

Iter	Objective	inf pr	inf du	lambda du	upsilon pr
0	5.0000259e+02	4.01e+00	1.00e+00	0.00e+00	0.00e+00
1	5.0136352e+02	3.99e+00	1.32e+02	5.20e-01	4.53e-03f
2	7.8288762e+02	3.25e+00	5.34e+01	1.11e-02	1.86e-01f
3	1.5959750e+03	1.64e+00	3.19e+01	2.74e-01	4.95e-01h
4	1.8980621e+03	1.13e+00	1.12e+01	7.56e-01	9.90e-01h
5	1.9529298e+03	4.13e-01	5.16e+01	8.76e-01	9.90e-01h
6	2.0023608e+03	7.26e-02	1.78e+03	8.26e-01	1.00e+00h
7	2.0036109e+03	3.21e-02	1.33e+01	8.42e-01	9.90e-01h
8	2.0038380e+03	1.84e-02	2.20e+02	8.30e-01	1.00e+00h
9	2.0036758e+03	8.07e-03	2.63e+03	7.92e-01	1.00e+00h
10	2.0017729e+03	2.70e-03	1.05e+03	7.67e-01	1.00e+00h
11	2.0017570e+03	7.57e-04	1.39e+03	7.75e-01	1.00e+00h
12	2.0017538e+03	2.39e-04	4.81e+01	9.81e-01	9.00e+00h
13	2.0017534e+03	2.34e-05	1.39e-05	1.00e+00	9.00e+00h
14	2.0017534e+03	4.03e-06	2.64e+01	1.00e+00	9.45e-01h
15	2.0017534e+03	1.65e-07	1.06e+00	1.00e+00	8.63e-01h
16	2.0017534e+03	2.86e-08	1.12e-01	1.00e+00	1.00e+00

Table 4.7: (Model IIII) Reduce the Ending Time

Target	scaled	unscaled
Objective	4.0035067119502833e+02	2.0017533559751416e+03
Dual infeasibility	4.1484881698752255e-10	2.0742440849376128e-09
Constraint violation	3.2241764813534246e-10	3.2241764813534246e-10
Complementarity	8.2742470009101295e-11	4.1371235004550645e-10
Overall NLP error	4.1484881698752255e-10	2.0742440849376128e-09

Solution time : 1.07949999999983 sec

Objective : 2001.75335597514

Final Time : 4.0035066052

The constraints have been changed for the same problem in order to obtain acceptable results that are better than the previous results, where we note the difference in the time of the solution in addition to the difference in the time of the final solution to the problem, and thus, we have obtained better results, more accuracy and less time.

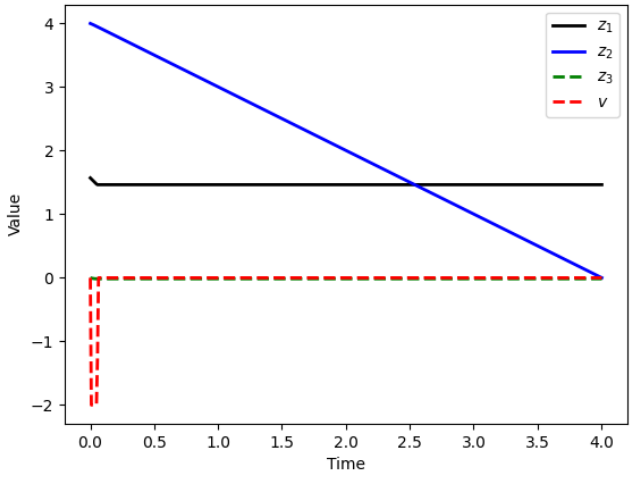


Figure 4.5: (Model IIII) Reduce the Ending Time

The try to reduce the time in the figure (4.5) solution for the last time since we are subject to the constraints, z_1 represents the black line will basically be equal to the value of v, so we can see that it started with a value of z and then it will have a slope and then run for about a little less than one second and it will can see that the slope will be equal to two there. Then if we move to z_1 the second stage, this is the final time multiplying $cos(z_1^3)$, and we can see that z_2 is the blue line starting at 4 and it will try to reach as quickly as possible to the value 0 at the end. So one of the limiting factors z_3 is actually 0, but in order to get to z_2 the value of v must be shifted, as we turn it on and then turn it off again and so we can see one of them goes down to zero as the last two conditions are needed.

4.5 Mathematical Model of Energy Systems

After searching and investigating the systems used in the Babylon gas power station, the most important problems facing the production of electric energy and how to control them, which gives us the lowest costs and the highest production of electric energy. The aim of the visit was to create a mathematical model based on the optimal control of the target function, which represents an increase in the production of electrical energy with the restrictions associated with the target function, which represents the control of temperatures and fuel consumption as well as other materials involved in the production of electrical energy. Through the visit, the required objectives were identified in terms of acceptance by the stakeholders in the power plant, and the visit resulted in the creation of three new mathematical models within this discipline, and energy supply systems can be built more effectively by adopting semi-static, dynamic, linear or non-linear. To achieve semi-static models, all time derivatives must be zeroed in the dynamic models. When the distance between two support points is 0. The models shown here are simply based on the rate at which energy is pumped through them.

4.5.1 Model IIIII: Load Observation

The term "Load observation" refers to a situation that occurs frequently in grids. Changes in demand dynamics are taken into account by the optimizer when balancing supply and demand an attempt is made at perfect foresight when a single generator has to respond to a single load. An initial increase in output is necessary to maintain supply and demand in balance and prevent overproduction.

$$\begin{cases} \underset{r}{Minimize}: & q = \int_{t=0}^{1} [1000\ max(0, d-g) + max(0, g-d)]\ dt \\ subject\ to: & \frac{dg}{dt} = r \\ & d = cos(2\pi t) + 3 \\ & -1 \leq r \leq 1 \\ & g(0) = d(0) = 4. \end{cases} \tag{4.16}$$

Variables used in the benchmark problems are defined below.

q is objective function, d is demand, g is generation and r is ramp rate.

Algorithm 5 :Load Observation

Choose initial values $g(0) = d(0) = 4,\ k = 0$

and go to the main step.

Main Step

1- Calculate the defect $d = cos(2\pi t) + 3$ in the boundary conditions.

If $-1 \leq r \leq 1$ stop;

2- Calculate the gradients

$$\frac{dg}{dt} = r$$

3- Solve the integral equation $\min_{r} \quad q = \int_0^1 [1000\ max(0, d-g) + max(0, g-d)]\ dt$

4- Compute the new estimates

$$g^{k+1} = g^k + q_g^k$$
$$d^{k+1} = d^k + qi_d^k$$
$$r^{k+1} = r^k + q_r^k$$

5- Compute the optimal solution (using KKT Method).

6- Replace $k \longleftarrow k+1,$ *and go to step* 1.

Code 5:Use Ipop Solver to Solve Load Observation

```
from gekko import GEKKO
import numpy as np
import matplotlib.pyplot as plt
t = np.linspace(0,1,101)
m = GEKKO(remote=False); m.time=t
d = m.Param(np.cos(2*np.pi*t)+3)
g = m.Var(d[0])
q  = m.CV(0)
q.STATUS=1; q.SPHI=q.SPLO=0
q.WSPHI=1000; q.WSPLO=1
r = m.MV(0,lb=-1,ub=1); r.STATUS=1
m.Equations([g.dt()==r, q==d-g])
m.options.IMODE=6; m.solve()
plt.plot(t,g,'b:',label='Production')
plt.plot(t,d,'r-',label='Demand')
plt.plot(t,r,'k--',label='Ramp Rate')
plt.legend(); plt.grid(); plt.show()
```

Table 4.8: Load Observation

Iter	Objective	inf pr	inf du	lambda du	upsilon pr
0	1.0009991e+003	2.00e+000	9.90e+001	0.00e+00	0.00e+00
1	3.2555231e+003	3.99e+00	1.32e+02	5.20e-01	4.53e-03f
2	2.9833158e+003	3.25e+00	5.34e+01	1.11e-02	1.86e-01f
3	8.1166395e+002	1.64e+00	3.19e+01	2.74e-01	4.95e-01h
4	6.6791895e+002	1.13e+00	1.12e+01	7.56e-01	9.90e-01h
5	3.4761100e+002	9.67e-002	5.43e+000	8.85e-001	8.12e-001f
6	2.0006301e+002	2.21e-002	3.68e+000	9.23e-001	7.72e-001f
7	1.1974314e+002	4.16e-003	1.56e-001	6.83e-001	8.12e-001f
8	8.8132974e+001	2.82e-004	5.37e-001	6.84e-001	9.32e-001f
9	8.3224841e+001	1.17e-004	1.28e-001	7.72e-001	5.84e-001f
10	7.9161016e+001	4.03e-005	3.00e-001	8.02e-001	6.57e-001f
11	7.7263578e+001	9.89e-006	8.38e-002	7.80e-001	7.54e-001f
12	7.6547543e+001	4.33e-007	4.96e-001	7.59e-001	9.56e-001f
13	7.6463437e+001	3.97e-016	5.96e-002	8.85e-001	1.00e+000h
14	7.6247138e+001	4.30e-016	5.10e-011	1.00e+00	1.00e+000f
15	7.6216539e+001	3.83e-016	9.01e-004	9.81e-001	9.96e-001f
16	7.6208823e+001	5.22e-016	2.09e-005	1.00e+00	9.95e-001f
17	7.6207386e+001	4.22e-016	1.37e-005	1.00e+00	9.97e-001f
18	7.6206934e+001	4.32e-016	3.07e-0051	1.00e+00	9.88e-001f
19	7.6206818e+001	4.35e-016	1.32e-004	1.00e+00	8.20e-001f
20	7.6206789e+001	4.37e-016	1.14e-007	1.00e+00	9.99e-001f

Table 4.9: The Load Observation

Target	scaled	unscaled
Objective	7.6206788841459385e+000	7.6206788841459385e+001
Dual infeasibility	1.1438565650223609e-007	1.1438565650223609e-006
Constraint violation	4.3736715638065249e-016	4.3736715638065249e-016
Complementarity	2.3596316301226464e-011	2.3596316301226463e-010
Overall NLP error	1.1438565650223609e-007	1.1438565650223609e-006

Solution time : 0.4628 sec

Objective : 76.2077888958377

Our optimization problem is to adjust the slope rate, which is often variable. The solution technique lies in reducing the objective function that violates the lack of demand, and this is equal to the total of three thousand plus the amount of demand and the error rate in the supply if it produces more than what is required. It is therefore very important not to short supply more than 3000 times to avoid power outages as well as adjust the ramp rate to reduce overproduction when available. Thus, we see that the average slope goes to negative one, which does not require a large amount of electricity during this time, and the slope rate reaches one when it is expected in the middle of the day that we will have more supply that we need later. And therefore begins to rise here in order to be able to meet this offer later. Thus, we can test the ability of the optimizer to proactively adjust the slope rate at an early stage so that we get the goal, as we have reached the goal in (0.4628) sec time and objective (76.2077888958377).

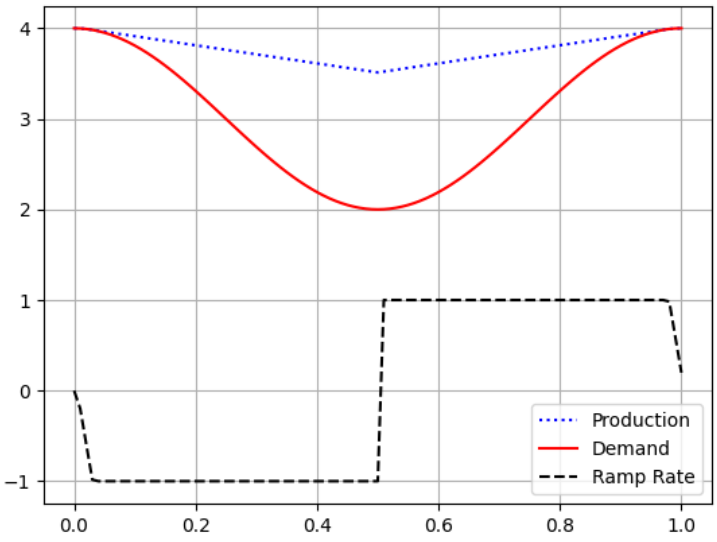


Figure 4.6: The Load Observation

As we can see in the figure that the blue dotted line represents generation and the red line represents the demand for production as it was consistent with the slope rate that represents the black line, and by testing the ability of the optimizer to proactively adjust the slope rate early so that it can meet the goal later.

4.5.2 Model IIIII: Combined Power Generation

In order to solve the problem of combined power generation, a producer must balance two competing aims, each of which has a distinct level of constraint. Combining both electricity and heat production in response to demand increases load observation's level of foresight to the same extent as load observation.

$$\begin{cases} \underset{r}{Minimize}: & q = \sum_{i=1}^{n} \int_{t=0}^{1} [3000\ max(0, d_i - g_i) + max(0, g_i - d_i)]\ dt \\ subject\ to: & \frac{dg_1}{dt} = r, \quad g_2 = 2g_1 \\ & d_1 = cos(2\pi t) + 1 \\ & d_2 = 1.5sin(2\pi t) + 5 \\ & -1 \leq r \leq 1 \\ & g_1(0) = d_1(0) = 2 \\ & g_2(0) = 8, \quad d_2(0) = 5 \end{cases} \tag{4.17}$$

Algorithm 6 : Combined Power Generation

Choose initial values $g_1(0) = d_1(0) = 2,\ k = 0$

and go to the main step.

Main Step

1- Calculate the defect

$$d_1 = cos(2\pi t) + 1$$
$$d_2 = 1.5sin(2\pi t) + 5$$
$$g_1(0) = d_1(0) = 2$$
$$g_2(0) = 8, \quad d_2(0) = 5$$
$$g_2 = 2g_1 \text{ in the boundary conditions.}$$
$$\text{If } -1 \leq r \leq 1 \text{ stop;}$$

2- Calculate the gradients

$$\frac{dg_1}{dt} = r$$

3- Solve the integral equation

$$\min_r \quad q = \sum_{i=1}^{n} \int_{t=0}^{1} [3000\ max(0, d_i - g_i) + max(0, g_i - d_i)]\ dt$$

4- Compute the new estimates

$$g_1^{k+1} = g_1^k + q_{g_1}^k$$
$$g_2^{k+1} = g_2^k + q_{g_2}^k$$
$$d_1^{k+1} = d_1^k + qi_{d_1}^k$$
$$d_2^{k+1} = d_2^k + qi_{d_2}^k$$
$$r^{k+1} = r^k + q_r^k$$

5- Compute the optimal solution (using Lagrange Method).

5- Replace $k \longleftarrow k+1$, *and go to step* 1.

Code 6:Use Ipop Solver to Solve Combined Power Generation

```
from gekko import GEKKO
import numpy as np
import matplotlib.pyplot as plt
t = np.linspace(0,1,101)
m = GEKKO(remote=False); m.time=t
d1  = m.Param(np.cos(2*np.pi*t)+1)
d2  = m.Param(1.5*np.sin(2*np.pi*t)+5)
g1  = m.Var(d1[0]); g2 = m.Intermediate(g1*2)
q1  = m.CV(0); q1.STATUS=1; q1.SPHI=q1.SPLO=0;
q1.WSPHI=3000; q1.WSPLO=1
q2  = m.CV(0); q2.STATUS=1; q2.SPHI=q2.SPLO=0;
q2.WSPHI=3000; q2.WSPLO=1
r = m.MV(0,lb=-1,ub=1); r.STATUS=1
m.Equations([g1.dt()==r, q1==d1-g1, q2==d2-g2])
m.options.IMODE=6; m.solve()
plt.figure(figsize=(8,5))
plt.subplot(3,1,1)
plt.plot(t,d1,'r-',label='Demand 1')
plt.plot(t,g1,'b:',label='Prod 1')
plt.grid(); plt.legend()
plt.subplot(3,1,2)
plt.plot(t,d2,'r-',label='Demand 2')
plt.plot(t,g2,'b:',label='Prod 2')
plt.grid(); plt.legend()
plt.subplot(3,1,3)
plt.plot(t,r,'k--',label='Ramp Rate')
plt.grid(); plt.legend(); plt.xlabel('Time')
plt.show()
```

Table 4.10: Cogeneration

Iter	Objective	inf pr	inf du	lambda du	upsilon pr
0	1.0009991e+003	2.00e+000	9.90e+001	0.00e+00	0.00e+00
1	1.4579254e+005	3.99e+00	1.32e+02	5.20e-01	4.53e-03f
2	1.4579254e+005	3.25e+00	5.34e+01	1.11e-02	1.86e-01f
3	1.4579254e+005	1.64e+00	3.19e+01	2.74e-01	4.95e-01h
4	1.4579254e+005	1.13e+00	1.12e+01	7.56e-01	9.90e-01h
5	1.4579254e+005	9.67e-002	5.43e+000	8.85e-001	8.12e-001f
6	1.4579254e+005	2.21e-002	3.68e+000	9.23e-001	7.72e-001f
7	1.1974314e+002	4.16e-003	1.56e-001	6.83e-001	8.12e-001f
8	1.4579254e+005	2.82e-004	5.37e-001	6.84e-001	9.32e-001f
9	1.4579254e+0051	1.17e-004	1.28e-001	7.72e-001	5.84e-001f
10	2.1535944e+005	4.03e-005	3.00e-001	8.02e-001	6.57e-001f
11	2.1535944e+005	9.89e-006	8.38e-002	7.80e-001	7.54e-001f
12	2.1535944e+005	4.33e-007	4.96e-001	7.59e-001	9.56e-001f
13	2.1535944e+005	3.97e-016	5.96e-002	8.85e-001	1.00e+000h
-	-	-	-	-	-
-	-	-	-	-	-
-	-	-	-	-	-
23	2.1686210e+005	1.69e-005	5.39e-001	7.76e-001	7.40e-001h
24	2.1686160e+005	6.68e-016	2.77e+000	8.34e-001	1.00e+000h
25	2.1685954e+005	4.44e-016	5.68e-004	1.00e+000	9.99e-001f
26	2.1685908e+005	2.62e-015	3.95e-003	1.00e+000	9.50e-001f
27	2.1685905e+005	2.02e-013	3.05e-004	1.00e+000	9.99e-001f
28	2.1685905e+005	3.06e-011	1.14e+001	1.00e+000	5.55e-001f
29	2.1685905e+005	4.44e-016	3.64e-012	1.00e+000	1.00e+000f

Table 4.11: Combined Power Generation

Target	scaled	unscaled
Objective	7.2286349285738079e+003	2.1685904785721423e+005
Dual infeasibility	3.6379788070917130e-012	1.0913936421275139e-010
Constraint violation	4.4408920985006262e-016	4.4408920985006262e-016
Complementarity	1.0966433988261922e-007	3.2899301964785765e-006
Overall NLP error	1.0966433988261922e-007	3.2899301964785765e-006

Solution time : 1.2430999999999999 sec

Objective : 216859.05209525148

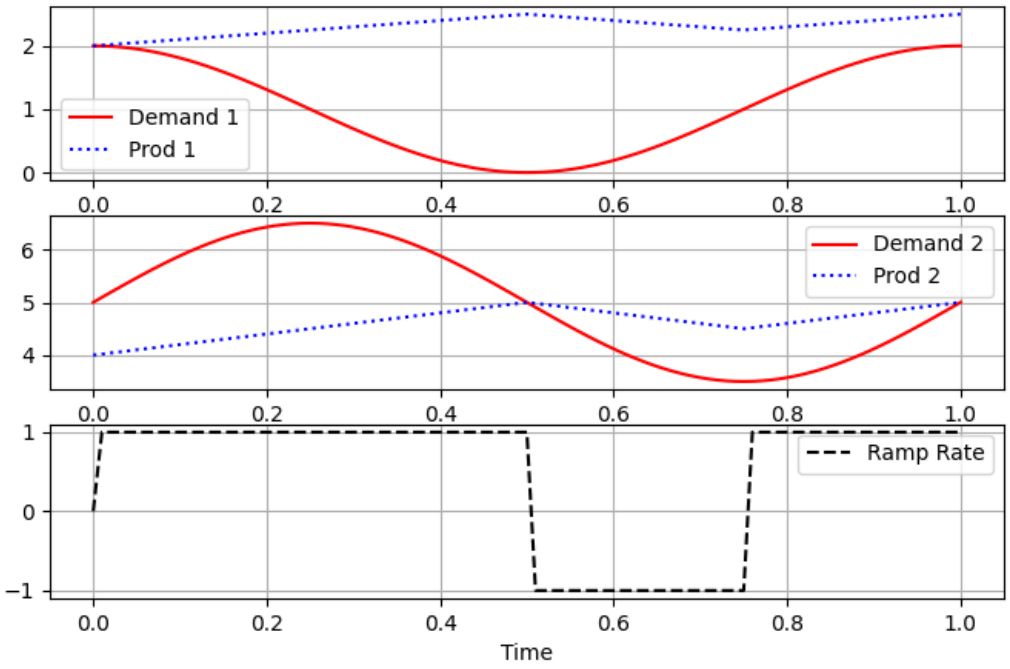


Figure 4.7: Combined Power Generation

The problem of combined Power Generation is common in places like refineries and chemical plants where an can produce electricity and steam anywhere there is some waste heat. That we can take advantage and use in particular, the power generation may be adjacent to the heating system which is often done in very cold countries where it makes sense to run the district heating system. So in this case, we have one product that we are scaling up or downgrading, but we are trying to produce two things with one order. The need two may be electricity and heat. The can see the first demand and the second demand where the demand is much higher than the other one where the heat is constrained. In order to produce heat you don't need much electricity, we need more heat. So it is very important not to short supply more than 3000 times to avoid power failure and also to adjust the ramp rate to reduce overproduction when available. Thus we see that the average slope goes to negative one, which does not require a large amount of electricity during this time, and the average slope goes to one when in the middle of the day we expect to have more supply than we need later. So here it starts to climb so we can meet that offer later. Thus, we can test the ability of the optimizer to proactively adjust the gradient rate at an early stage so that we can achieve the target, as we hit the target at (1.2430999999999999) with time and target (216859.05209525148).

We also see in the figure that the blue dotted line represents generation and the red line represents production demand for electricity and heat in the two figures because it was consistent with the slope rate representing the black line, and by testing the ability of the optimizer to proactively adjust the slope assessment early so that it can achieve the goal later.

4.5.3 Model IIIIIII: Cogeneration of Three Different Types of Energy

Cogeneration of three different types of energy system $n = 3$ with two producers, three products, and three demand profiles is created in the third model issue. When it comes to manufacturing the two basic goods, the primary producer is bound by ramp-rate (e.g., electricity and heat). For example, a solid oxide electrolysis cell can use these first two products to produce a third product (e.g., hydrogen), thus optimizing production while avoiding supply shortages for products one and two. The optimizer's insight is as good as ever.

$$
\begin{cases}
\underset{r_1,r_2}{Minimize}: & q = \sum_{i=1}^{n} \int_{t=0}^{1} [3000\ max(0, d_i - g_i) + max(0, g_i - d_i)]\ dt - \int_{t=0}^{1} g_3 dt \\
subject\ to: & \frac{dg_1}{dt} = r_1, \quad \frac{dg_3}{dt} = r_3 \\
& g_2 = 2g_1 \\
& d_1 = cos(2\pi t) + 3 + 2g_3 \\
& d_2 = 1.5sin(2\pi t) + 7 + 3g_3 \\
& d_3 = max[0, -0.2\sin(2\pi t)] \\
& -1 \leq r_1 \leq 1, \quad -1 \leq r_3 \leq 1 \\
& g_1(0) = d_1(0) = 4 \\
& g_2(0) = 8, \quad d_2(0) = 7 \\
& g_3(0) = d_3(0) = 0
\end{cases}
\tag{4.18}
$$

Algorithm 7 : Cogeneration of Three Different Types of Energy

Choose initial values $g_1(0) = d_1(0) = 4,\ \ g_2(0) = 8,\ \ \ d_2(0) = 7,\ \ g_3(0) = d_3(0) = 0,\ \ \ k = 0$ and go to the main step.

Main Step

1- Calculate the defect

$$d_1 = cos(2\pi t) + 3 + 2g_3,$$
$$d_2 = 1.5sin(2\pi t) + 7 + 3g_3,$$
$$d_3 = max[0, -0.2\sin(2\pi t)],$$
$$g_1(0) = d_1(0) = 4,$$
$$g_2(0) = 8, \quad d_2(0) = 7,$$
$$g_3(0) = d_3(0) = 0, \ \ g_2 = 2g_1, \text{ in the boundary conditions.}$$
$$\text{if } -1 \leq r_1 \leq 1, \ -1 \leq r_3 \leq 1 \text{ stop;}$$

2- Calculate the gradients

$$\frac{dg_1}{dt} = r_1, \quad \frac{dg_3}{dt} = r_3$$

3- Solve the integral equation

$$\min_r \ : q = \sum_{i=1}^{n} \int_{t=0}^{1} [3000\ max(0, d_i - g_i) + max(0, g_i - d_i)]\ dt - \int_{t=0}^{1} g_3 dt$$

4- Compute the new estimates

$$g_1^{k+1} = g_1^k + q_{g_1}^k, \quad g_2^{k+1} = g_2^k + q_{g_2}^k, \quad g_3^{k+1} = g_3^k + q_{g_3}^k, \quad d_1^{k+1} = d_1^k + qi_{d_1}^k,$$
$$d_2^{k+1} = d_2^k + qi_{d_2}^k, \quad d_3^{k+1} = d_3^k + qi_{d_3}^k, \quad r_1^{k+1} = r_1^k + q_{r_1}^k, \quad r_3^{k+1} = r_3^k + q_{r_3}^k$$

5- Compute the optimal solution (using KKT Method).

6- Replace $k \longleftarrow k + 1$, *and go to step* 1.

Code 7:Use Ipop Solver to solve Cogeneration of Three Different Types of Energy

```
import numpy as np
from gekko import GEKKO
import matplotlib.pyplot as plt
t = np.linspace(0,1,101)
m = GEKKO(remote=False); m.time=t
d1 = m.Param(np.cos(2*np.pi*t)+3)
d2 = m.Param(1.5*np.sin(2*np.pi*t)+7)
d3 = m.Param(np.clip(-0.2*np.sin(2*np.pi*t),0,None))
d1v = m.Var(d1[0]); d2v=m.Intermediate(d1v*2)
d3v = m.Var(0,lb=0); d3v1=m.Var(0); d3v2=m.Var(0)
t1 = m.Intermediate(d1+d3v1); t2 = m.Intermediate(d2+d3v2)
e  = m.CV(0); e.STATUS=1; e.SPHI=e.SPLO=0; e.WSPHI=3000;
e.WSPLO=1
h  = m.CV(0); h.STATUS=1; h.SPHI=h.SPLO=0; h.WSPHI=3000;
h.WSPLO=1
z  = m.CV(0); z.STATUS=1; z.SPHI=z.SPLO=0; z.WSPHI=3000;
z.WSPLO=1
r1 = m.MV(0,lb=-1,ub=1); r1.STATUS=1; r1.DCOST=0.0
r3 = m.MV(0,lb=-1,ub=1); r3.STATUS=1; r3.DCOST=0.0
m.Equations([d1v.dt()==r1, e==t1-d1v, h==t2-d2v])
m.Equations([d3v.dt()==r3, z==d3-d3v, d3v1==d3v*2,
d3v2==d3v*3]) m.Maximize(d3v)
m.options.IMODE=6; m.options.SOLVER=1; m.solve()
plt.figure(figsize=(8,7))
plt.subplot(4,1,1)
plt.plot(t,t1,'r-',label='Demand 1')
plt.plot(t,d1v,'b:',label='Prod 1')
plt.legend(); plt.grid()
plt.subplot(4,1,2)
plt.plot(t,t2,'r-',label='Demand 2')
plt.plot(t,d2v,'b:',label='Prod 2')
plt.legend(); plt.grid()
plt.subplot(4,1,3)
plt.plot(t,d3,'r-',label='Demand 3')
plt.plot(t,d3v,'b:',label='Prod 3')
plt.legend(); plt.grid()
plt.subplot(4,1,4)
plt.plot(t,r1,'k:',label='Ramp Rate 1')
plt.plot(t,r3,'k--',label='Ramp Rate 3')
plt.legend(); plt.grid(); plt.xlabel('Time')
plt.show()
```

Table 4.12: Cogeneration of Three Different Types of Energy

Iter	Objective	Convergence
0	1.44645E+03	1.00000E+00
1	6.00263E+02	2.70000E-09
2	6.00160E+02	3.47576E-01
3	6.00160E+02	6.00000E-10
4	6.00160E+02	2.36790E-16
5	6.00160E+02	2.36790E-16

Solution time : 3.6222000000000003 sec

Objective : 600.1604788785155

Electricity, heat, and hydrogen are three common byproducts. First, a cogeneration system generates electricity and heat, while a second product (such as a solid An oxide electrolysis cell) generates hydrogen. In addition to the fact that the first's outputs are inputs, they both face the identical slope rate limitations (3 grams). Two units of electricity and three units of heat are required to produce one unit of hydrogen. Hydrogen production takes advantage of any additional electricity and heat that may be available. Increase and reduce production as before All three goods have been designated as subject to sanctions at this time. term also added to the objective function of hydrogen catalyst synthesis without creating an increase in energy or heat production. It was conceived. It ramps up production at its highest rate until t = 0:65, then it drops production at its maximum rate until the end time horizon. The second product is able to fulfill the hydrogen need at t = 0:9 by doing so. Again, this problem is not specific to any particular system, but the components are the same as those in the transmission Optimization problems, such as cogeneration of energy and heat combined with chemical synthesis. In addition, several district-wide systems generate electricity,

heat, and cold as another example of triple generation. Combined with Cases, a third product, demand for other products is increased significantly.

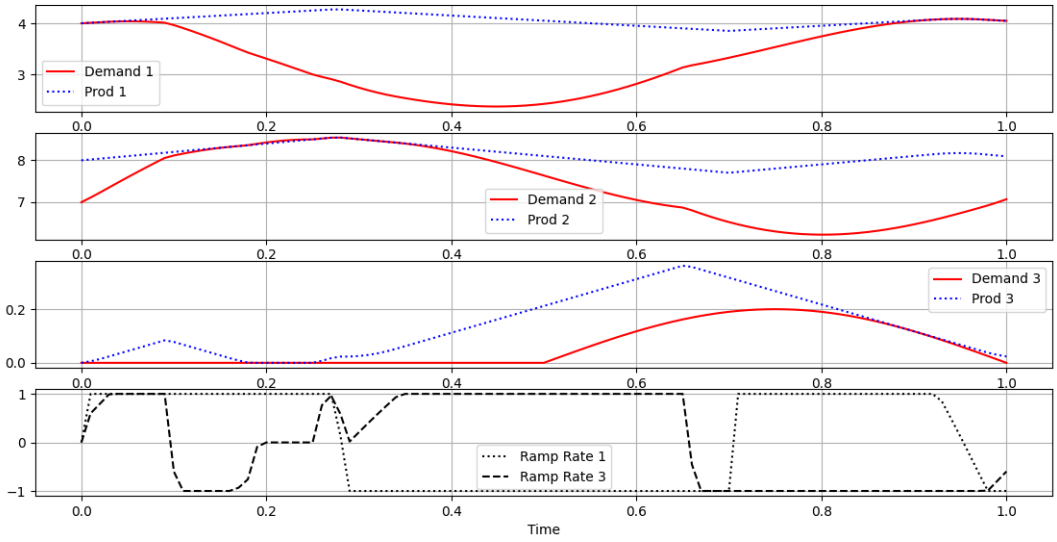


Figure 4.8: Cogeneration of Three Different Types of Energy

We also see in the four figures that the blue dotted line represents generation and the red line represents the production demand for electricity and heat in the first and second figures because it was consistent with the regression rate, which represents the black line and the third figure, which represents hydrogen production by electricity and heat.

4.6 Conclusions

An effective approach to optimal control was developed by applying algorithms that showed us a high level of accuracy convergence to reach the goal. Seven models were selected within this thesis, as follows :

- The numerical solution is developed with the desired path and variation is minimized by modifiable parameters in the model which may change at each time step. We perform the initial control action, and then do it all over again the next time.

- A program has been developed to track our progress and show results assuming that 2 minutes is the estimated time. The cost function is rounded as a cost function. Then, the function approximation coefficients are treated as optimization variables and the problem is "copied" into a nonlinear optimization problem.

- We developed a new mathematical model for any time-dependent function subject to certain restrictions in order to solve some contemporary problems in various sciences, by controlling the minimum and optimal time of completion.

- We will change the limitations of the previous issue to get satisfactory results in less time. we developed a new mathematical model for any time-dependent function subject to certain restrictions in order to solve some contemporary problems , by controlling the minimum and optimal time of completion.

- A loading model is developed, which is typical in the network architecture. The objective of the optimizer is to find a balance between supply and demand in the face of dynamic changes in demand. A single generator, limited by installation space, makes an effort to anticipate and handle a single load appropriately.

- A high-resolution dynamic model of an electric and thermal power plant has been developed aiming to achieve two constrained goals at different times in response to electricity demand and a new heat demand profile, again with perfect insight.

4.7 Future Work

These algorithms can be developed to work in several areas that require optimal control, Some suggestions for what should be done in the future are provided below. As well as potential tweaks, these also include fresh uses of the technique :

- Optimum control of real life problems in engineering and environmental sciences.
- Optimum control of sterilization of canned foods.
- Optimal control for optimal management of waste water treatment plants.
- Optimal control problems with aerospace applications are handled in a realistic model-based manner. Particular attention is paid to issues that involve multiple forms of restriction.
- Optimal control of drug injection or resource allocation problems, it is not practical to constantly change control. In dealing with these problems it is highly preferred to change control only at certain moments of time and to keep it constant.
- Problems of optimal control for the treatment of cancer patients The two most widely used cancer treatments, radiotherapy and systemic therapy, as they naturally lend themselves to the theory of optimal control as a means of personalizing treatment plans in a rigorous manner.
- The primary objective of vehicle improvement in motor sports is to reduce the amount of time required to cover a set distance. To foresee how adjusting certain design parameters will affect a racing car's performance, engineers often turn to computer simulations. When viewed as a problem of determining the best possible trajectory, of course.

REFERENCES

[1] Om Prakash Agrawal. Generalized variational problems and euler–lagrange equations. *Computers & Mathematics with Applications*, 59(5):1852–1864, 2010.

[2] Amir Ali Ahmadi and Jeffrey Zhang. On the complexity of finding a local minimizer of a quadratic function over a polytope. *Mathematical Programming*, pages 1–10, 2022.

[3] Ahmed Al-Jilawi. *Solving the Semidefinite Programming Relaxation of Max-cut Using an Augmented Lagrangian Method*. Northern Illinois University, 2019.

[4] Joel AE Andersson, Joris Gillis, Greg Horn, James B Rawlings, and Moritz Diehl. Casadi: a software framework for nonlinear optimization and optimal control. *Mathematical Programming Computation*, 11(1):1–36, 2019.

[5] Johannes M Bader. *Hypervolume-based search for multiobjective optimization: theory and methods*. Number 112. Johannes Bader, 2010.

[6] Rabindra R Bajracharya, Paul J Emigh, and Corinne A Manogue. Students' strategies for solving a multirepresentational partial derivative problem in thermodynamics. *Physical Review Physics Education Research*, 15(2):020124, 2019.

[7] Dumitru Baleanu, Samaneh Sadat Sajjadi, Amin Jajarmi, and Jihad H Asad. New features of the fractional euler-lagrange equations for a physical system within non-singular derivative operator. *The European Physical Journal Plus*, 134(4):1–10, 2019.

[8] Jonathan F Bard. *Practical bilevel optimization: algorithms and applications*, volume 30. Springer Science & Business Media, 2013.

[9] Logan DR Beal, Daniel C Hill, R Abraham Martin, and John D Hedengren. Gekko optimization suite. *Processes*, 6(8):106, 2018.

[10] Pietro Belotti, Pierre Bonami, Matteo Fischetti, Andrea Lodi, Michele Monaci, Amaya Nogales-Gómez, and Domenico Salvagnin. On handling indicator constraints in mixed integer programming. *Computational Optimization and Applications*, 65(3):545–566, 2016.

[11] Lorenz T Biegler, Omar Ghattas, Matthias Heinkenschloss, David Keyes, and Bart van Bloemen Waanders. *Real-time PDE-constrained Optimization*. SIAM, 2007.

[12] Julian Blank and Kalyanmoy Deb. Pymoo: Multi-objective optimization in python. *IEEE Access*, 8:89497–89509, 2020.

[13] Taras Bodnar, Stepan Mazur, and Yarema Okhrin. Bayesian estimation of the global minimum variance portfolio. *European Journal of Operational Research*, 256(1):292–307, 2017.

[14] Alfio Borzì and Volker Schulz. *Computational optimization of systems governed by partial differential equations*. SIAM, 2011.

[15] Radu Ioan Bot, Sorin-Mihai Grad, and Gert Wanka. *Duality in vector optimization*. Springer Science & Business Media, 2009.

[16] Arthur E Bryson and Yu-Chi Ho. *Applied optimal control: optimization, estimation, and control*. Routledge, 2018.

[17] Jingjing Bu, Afshin Mesbahi, Maryam Fazel, and Mehran Mesbahi. Lqr through the lens of first order methods: Discrete-time case. *arXiv preprint arXiv:1907.08921*, 2019.

[18] Aurel Bulgac and Michael McNeil Forbes. Use of the discrete variable representation basis in nuclear physics. *Physical Review C*, 87(5):051301, 2013.

[19] María J Cánovas, René Henrion, Marco A López, and Juan Parra. Outer limit of subdifferentials and calmness moduli in linear and nonlinear programming. *Journal of Optimization Theory and Applications*, 169(3):925–952, 2016.

[20] Coralia Cartis, Nicholas IM Gould, and Philippe L Toint. Evaluation complexity bounds for smooth constrained nonlinear optimization using scaled kkt conditions and high-order models. *Approximation and Optimization: Algorithms, Complexity and Applications*, pages 5–26, 2019.

[21] Temuer Chaolu and Pang Jing. An algorithm for the complete symmetry classification of differential equations based on wu's method. *Journal of Engineering Mathematics*, 66(1):181–199, 2010.

[22] Jianchun Chu and Heming Jiao. Curvature estimates for a class of hessian type equations. *Calculus of Variations and Partial Differential Equations*, 60(3):1–18, 2021.

[23] Francis Clarke. *Necessary conditions in dynamic optimization.* American Mathematical Soc., 2005.

[24] David W Coit and Enrico Zio. The evolution of system reliability optimization. *Reliability Engineering & System Safety*, 192:106259, 2019.

[25] Peijin Cong, Junlong Zhou, Liying Li, Kun Cao, Tongquan Wei, and Keqin Li. A survey of hierarchical energy optimization for mobile edge computing: A perspective from end devices to the cloud. *ACM Computing Surveys (CSUR)*, 53(2):1–44, 2020.

[26] Richard Cottle, Mukund N Thapa, et al. *Linear and nonlinear optimization*, volume 253. Springer, 2017.

[27] Damek Davis and Wotao Yin. Convergence rate analysis of several splitting schemes. In *Splitting methods in communication, imaging, science, and engineering*, pages 115–163. Springer, 2016.

[28] Ibrahim Dincer, Marc A Rosen, and Pouria Ahmadi. *Optimization of energy systems*. John Wiley & Sons, 2017.

[29] Shinto Eguchi and Osamu Komori. Path connectedness on a space of probability density functions. In *International Conference on Geometric Science of Information*, pages 615–624. Springer, 2015.

[30] Anders Eriksson, Carl Olsson, Fredrik Kahl, and Tat-Jun Chin. Rotation averaging and strong duality. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 127–135, 2018.

[31] Hans Joachim Ferreau, Christian Kirches, Andreas Potschka, Hans Georg Bock, and Moritz Diehl. qpoases: A parametric active-set algorithm for quadratic programming. *Mathematical Programming Computation*, 6(4):327–363, 2014.

[32] Gerald B Folland. Introduction to partial differential equations. In *Introduction to Partial Differential Equations*. Princeton university press, 2020.

[33] Virginie Gabrel and Céline Murat. Robustness and duality in linear programming. *Journal of the Operational Research Society*, 61(8):1288–1296, 2010.

[34] Nathaniel S Gates, Daniel C Hill, Blake W Billings, Kody M Powell, and John D Hedengren. Benchmarks for grid energy management with python gekko. In *2021 60th IEEE Conference on Decision and Control (CDC)*, pages 4868–4874. IEEE, 2021.

[35] Benyamin Ghojogh, Ali Ghodsi, Fakhri Karray, and Mark Crowley. Kkt conditions, first-order and second-order optimization, and distributed optimization: Tutorial and survey. *arXiv preprint arXiv:2110.01858*, 2021.

[36] Chris Godsil and Gordon F Royle. *Algebraic graph theory*, volume 207. Springer Science & Business Media, 2001.

[37] Michael T Goodrich and Roberto Tamassia. *Algorithm design and applications*. Wiley Hoboken, 2015.

[38] M Eshaghi Gordji, M Rostamian Delavar, and M De La Sen. On ϕ-convex functions. *J. Math. Inequal*, 10(1):173–183, 2016.

[39] John Gregory and Cantian Lin. *Constrained optimization in the calculus of variations and optimal control theory*. Chapman and Hall/CRC, 2018.

[40] Vince Grolmusz. A note on the pagerank of undirected graphs. *Information Processing Letters*, 115(6-8):633–634, 2015.

[41] Shuhang Gu, Lei Zhang, Wangmeng Zuo, and Xiangchu Feng. Weighted nuclear norm minimization with application to image denoising. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2862–2869, 2014.

[42] Dajun Guo and Vangipuram Lakshmikantham. *Nonlinear problems in abstract cones*, volume 5. Academic press, 2014.

[43] Muneeb Ul Hassan, Mubashir Husain Rehmani, and Jinjun Chen. Differential privacy techniques for cyber physical systems: a survey. *IEEE Communications Surveys & Tutorials*, 22(1):746–789, 2019.

[44] Jack Heider. Quadratic programming, 2021.

[45] Mark Hillery. Partial particle and wave information and weak duality games. *Journal of Physics A: Mathematical and Theoretical*, 54(49):495301, 2021.

[46] Kolumban Hutter. *Theoretical glaciology: material science of ice and the mechanics of glaciers and ice sheets*, volume 1. Springer, 2017.

[47] Prateek Jain and Purushottam Kar. Non-convex optimization for machine learning. *arXiv preprint arXiv:1712.07897*, 2017.

[48] Yetao Ji, Jiaze Tu, Hanfeng Zhou, Wenyong Gui, Guoxi Liang, Huiling Chen, and Mingjing Wang. An adaptive chaotic sine cosine algorithm for constrained and unconstrained optimization. *Complexity*, 2020, 2020.

[49] Sai Praneeth Karimireddy, Lie He, and Martin Jaggi. Learning from history for byzantine robust optimization. In *International Conference on Machine Learning*, pages 5311–5319. PMLR, 2021.

[50] Mohammad Khishe and Mohammad Reza Mosavi. Chimp optimization algorithm. *Expert systems with applications*, 149:113338, 2020.

[51] Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. A diversity-promoting objective function for neural conversation models. *arXiv preprint arXiv:1510.03055*, 2015.

[52] Anna Lischke, Guofei Pang, Mamikon Gulian, Fangying Song, Christian Glusa, Xiaoning Zheng, Zhiping Mao, Wei Cai, Mark M Meerschaert, Mark Ainsworth, et al. What is the fractional laplacian? a comparative review with new results. *Journal of Computational Physics*, 404:109009, 2020.

[53] Takanori Maehara, Hirofumi Suzuki, and Masakazu Ishihata. Exact computation of influence spread by binary decision diagrams. In *Proceedings of the 26th International Conference on World Wide Web*, pages 947–956, 2017.

[54] Mazin Kareem Marjan and Ahmed Sabah Aljilawi. A new mathematics model to find optimal control of optimization problem. *Journal of Positive School Psychology*, pages 2346–2351, 2022.

[55] Carlos Mastalli, Rohan Budhiraja, Wolfgang Merkt, Guilhem Saurel, Bilal Hammoud, Maximilien Naveau, Justin Carpentier, Ludovic Righetti, Sethu Vijayakumar, and Nicolas Mansard. Crocoddyl: An efficient and versatile framework for multi-contact optimal control. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2536–2542. IEEE, 2020.

[56] Michalis Mavrovouniotis, Changhe Li, and Shengxiang Yang. A survey of swarm intelligence for dynamic optimization: Algorithms and applications. *Swarm and Evolutionary Computation*, 33:1–17, 2017.

[57] Lester James Miranda. Pyswarms: a research toolkit for particle swarm optimization in python. *Journal of Open Source Software*, 3(21):433, 2018.

[58] Kazuo Murota. Linear programming. In *Computer Vision: A Reference Guide*, pages 1–7. Springer, 2020.

[59] Tenavi Nakamura-Zimmerer, Qi Gong, and Wei Kang. Adaptive deep learning for high-dimensional hamilton–jacobi–bellman equations. *SIAM Journal on Scientific Computing*, 43(2):A1221–A1247, 2021.

[60] Trung Thanh Nguyen, Shengxiang Yang, and Juergen Branke. Evolutionary dynamic optimization: A survey of the state of the art. *Swarm and Evolutionary Computation*, 6:1–24, 2012.

[61] Abraham Ochoche. Improving the modified euler method. *Leonardo Journal of sciences*, 6(10):1–8, 2007.

[62] Romeo Orsolino, Michele Focchi, Stéphane Caron, Gennaro Raiola, Victor Barasuol, Darwin G Caldwell, and Claudio Semini. Feasible region: An actuation-aware extension of the support region. *IEEE Transactions on Robotics*, 36(4):1239–1255, 2020.

[63] Quan-Ke Pan, Hong-Yan Sang, Jun-Hua Duan, and Liang Gao. An improved fruit fly optimization algorithm for continuous function optimization problems. *Knowledge-Based Systems*, 62:69–83, 2014.

[64] Michael Posa and Russ Tedrake. Direct trajectory optimization of rigid body dynamical systems through contact. In *Algorithmic foundations of robotics X*, pages 527–542. Springer, 2013.

[65] Belén Pulido, Alba María Franco-Pereira, and Rosa Elvira Lillo. Functional clustering via multivariate clustering. *arXiv preprint arXiv:2108.00217*, 2021.

[66] Gade Pandu Rangaiah. *Multi-objective optimization: techniques and applications in chemical engineering*, volume 5. world scientific, 2016.

[67] R Rao. Jaya: A simple and new optimization algorithm for solving constrained and unconstrained optimization problems. *International Journal of Industrial Engineering Computations*, 7(1):19–34, 2016.

[68] Huilong Ren, Xiaoying Zhuang, and Timon Rabczuk. A higher order nonlocal operator method for solving partial differential equations. *Computer Methods in Applied Mechanics and Engineering*, 367:113132, 2020.

[69] E Skelton Robert, Tetsuya Iwasaki, and M Grigoriadis Karolos. *A unified algebraic approach to linear control design*. Routledge, 2017.

[70] Suresh P Sethi. What is optimal control theory? In *Optimal Control Theory*, pages 1–26. Springer, 2019.

[71] Kamran Ali Shah, Fantai Meng, Ye Li, Ryozo Nagamune, Yarong Zhou, Zhengru Ren, and Zhiyu Jiang. A synthesis of feasible control methods for floating offshore wind turbine system dynamics. *Renewable and Sustainable Energy Reviews*, 151:111525, 2021.

[72] Mehran Shariat, Mahdi Shariati, Amirhossein Madadi, and Karzan Wakil. Computational lagrangian multiplier method by using for optimization and sensitivity analysis of rectangular reinforced concrete beams. *Steel Compos. Struct*, 29(2):243–256, 2018.

[73] Igor Škrjanc, Sašo Blažič, Edwin Lughofer, and Dejan Dovžan. Inner matrix norms in evolving cauchy possibilistic clustering for classification and regression from data streams. *Information Sciences*, 478:540–563, 2019.

[74] Hira Soomro, Nooraini Zainuddin, Hanita Daud, and Joshua Sunday. Optimized hybrid block adams method for solving first order ordinary differential equations. *Computers, Materials and Continua*, 72(2):2947–2961, 2022.

[75] Gilbert Strang. *Differential equations and linear algebra*. Wellesley-Cambridge Press Wellesley, 2014.

[76] Wenyu Sun and Ya-Xiang Yuan. *Optimization theory and methods: nonlinear programming*, volume 1. Springer Science & Business Media, 2006.

[77] Yifei Suna, Yasunori Koborib, Minh Tri Tranc, Anna Kuwanad, and Haruo Kobayashie. Single-inductor dual-output converters with pwm, hysteretic, soft switching and current controls. *J. of Tech. & Soc. Sci*, 4(2):7–32, 2020.

[78] Vendoti Suresh, M Muralidhar, and R Kiranmayi. Modelling and optimization of an off-grid hybrid renewable energy system for electrification in a rural areas. *Energy Reports*, 6:594–604, 2020.

[79] György Szabó, Kinga S Bodó, Benjamin Allen, and Martin A Nowak. Fourier decomposition of payoff matrix for symmetric three-strategy games. *Physical Review E*, 90(4):042811, 2014.

[80] Zekun Tong, Yuxuan Liang, Henghui Ding, Yongxing Dai, Xinke Li, and Changhu Wang. Directed graph contrastive learning. *Advances in Neural Information Processing Systems*, 34:19580–19593, 2021.

[81] Emmanuel Trélat. Optimal control and applications to aerospace: some results and challenges. *Journal of Optimization Theory and Applications*, 154(3):713–758, 2012.

[82] Dongshu Wang, Dapei Tan, and Lei Liu. Particle swarm optimization algorithm: an overview. *Soft Computing*, 22(2):387–408, 2018.

[83] Jianfeng Wang, Mei Lu, Francesco Belardo, and Milan Randić. The anti-adjacency matrix of a graph: Eccentricity matrix. *Discrete Applied Mathematics*, 251:299–309, 2018.

[84] Zhenlin Wang, Xun Huan, and Krishna Garikipati. Variational system identification of the partial differential equations governing the physics of pattern-formation: Inference under varying fidelity and noise. *Computer Methods in Applied Mechanics and Engineering*, 356:44–74, 2019.

[85] Zhiyuan Wang and Gade Pandu Rangaiah. Application and analysis of methods for selecting an optimal solution from the pareto-optimal front obtained by multiobjective optimization. *Industrial & Engineering Chemistry Research*, 56(2):560–574, 2017.

[86] Dongpo Xu, Yili Xia, and Danilo P Mandic. Optimization in quaternion dynamic systems: gradient, hessian, and learning algorithms. *IEEE transactions on neural networks and learning systems*, 27(2):249–261, 2015.

[87] Xin-She Yang. *Optimization techniques and applications with examples*. John Wiley & Sons, 2018.

[88] Danial Yazdani, Ran Cheng, Donya Yazdani, Jürgen Branke, Yaochu Jin, and Xin Yao. A survey of evolutionary continuous dynamic optimization over two

decades—part b. *IEEE Transactions on Evolutionary Computation*, 25(4):630–650, 2021.

[89] SM Zahraee, M Khalaji Assadi, and R Saidur. Application of artificial intelligence methods for hybrid energy system optimization. *Renewable and Sustainable Energy Reviews*, 66:617–630, 2016.

[90] Huan Zhang, Pengchuan Zhang, and Cho-Jui Hsieh. Recurjac: An efficient recursive algorithm for bounding jacobian matrix of neural networks and its applications. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 5757–5764, 2019.

المستخلص

تهدف هذه الاطروحة إيجاد تقنية جديدة للتحكم الأمثل تتطلب التحكم في المستوى من حيث توفير الوقت والسرعة والدقة. في الواقع ، تمت دراسة مجموعة من الأنظمة الرياضية ، والتي تعتمد جميعها على المعادلات التفاضلية ، لأن هذه الأنظمة تمثل ارتباطًا وثيقًا بالتحكم الأمثل. تكمن معظم مشاكل هذه الأنظمة في التحكم في إيجاد الحل الأمثل. لهذا ، تم اعتماد طريقة التحسين العددي كوسيلة لإيجاد حلول لمشاكل هذه الأنظمة الرياضية من خلال إيجاد حل تقريبي. في النموذج الأول ، نتعامل مع الحركة ، يمكن عكس كتلة الاحتكاك الأقل انزلاقًا بأقل جهد تحكم عن طريق تقييد مقدار الإزاحة المسموح للكتلة بالتحرك. الثاني تجسيد تم تحسين التحكم في نظام ديناميكي على مدى فترة زمنية ليكون وظيفة موضوعية. في هذا الصدد ، تم تطبيق خوارزمية فعالة حول النظام الديناميكي ، وهو عبارة عن مركبة فضائية ذات ضوابط مشابهة لدوافع الصواريخ ، والهدف من التي تصل إلى القمر بأقل كمية من الوقود المستهلك. النموذج الثالث والرابع يخضعان لقيود تفاضلية. تسعى العديد من المجالات بما في ذلك نقل الإنتاج وأنظمة الطاقة إلى تقليل الوقت الإجمالي المطلوب لمهمة ما. التحكم الأمثل تسعى المشكلة لتقليل الوقت النهائي. والهدف من المُحسِّن هو تحقيق الحالة النهائية المطلوبة في أقل وقت ممكن. بالنسبة لأنظمة الشبكة ، فإن مشكلة النموذج الخامس هي تمثيل لسيناريو التحميل التالي الشائع جدًا.

جمهورية العراق
وزارة التعليم العالي
والبحث العلمي
جامعة بابل
كلية التربية للعلوم الصرفة
قسم الرياضيات

توظيف نماذج التحكم الرياضية لتحسين التفاضل العددي

أطروحة مقدمة الى مجلس كلية التربية للعلوم الصرفة
في جامعة بابل
جزءا من متطلبات نيل درجة الدكتوراه فلسفة في
التربية / الرياضيات

من قبل

مازن كريم كاظم مرجان

بإشراف

أ. م. د احمد صباح احمد الجيلاوي

١٤٤٤ هـ ٢٠٢٣ م