Republic of Iraq

Ministry of Higher Education & Scientific Research

University of Babylon

College of Education for Pure Sciences

Department of Mathematics

New Approaches to Solve Two-sided Optimization Problems in Numerical Analysis

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

Ammar Imad Nadhim Nomi

Supervised by

Assist. Dr. Ahmed Sabah Al-Jilawi

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

بِسْمِ اللَّهِ الرَّحْمَنِ الرَّحِيمِ

وَمَن يَتَّقِ اللَّهَ يَجْعَل لَّهُ مَخْرَجًا
وَيَرْزُقْهُ مِنْ حَيْثُ لَا يَحْتَسِبُ
وَمَن يَتَوَكَّلْ عَلَى اللَّهِ فَهُوَ حَسْبُهُ
إِنَّ اللَّهَ بَالِغُ أَمْرِهِ قَدْ جَعَلَ اللَّهُ
لِكُلِّ شَيْءٍ قَدْرًا

الطلاق : ٢-٣

صدق الله العظيم

The Supervisor Certificate

I certify that this Dissertation entitled " New Approaches to Solve Two-sided Optimization Problems in Numerical Analysis " was prepared by the student " Ammar Imad Nadhim " under my supervision at the University of Babylon, College of Education for Pure Sciences as a partial fulfillment of the requirement the Degree of Doctor of Philosophy in Education /Mathematics.

Signature: Name: Dr. Ahmed Sabah Al-Jilawi

Scientific Grade: Assistant Professor

Date: / / 2023

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

Signature:

Name: Dr. Azal Jaafar Musa

Head of Mathematics Department

Scientific Grade: Assistant Professor

Date: / / 2023

Scientific Supervisor's Certification

This is to certify that I have read this Dissertation entitled " New Approaches to Solve Two-sided Optimization Problems in Numerical Analysis " and I found it is qualified for debate.

Signature:

Name: Dr. Mushtaq Karim Abed Alrahim

Title: Assistant Professor

Address : College of Administration and Economics , University of Karbala

Date: / / 2023

Scientific Supervisor's Certification

This is to certify that I have read this Dissertation entitled " New Approaches to Solve Two-sided Optimization Problems in Numerical Analysis " and I found it is qualified for debate.

Signature:

Name: Dr.Saad Shaker Mahmoud

Title:Professor

Address : College of Education , University of Mustansiriyah

Date: / / 2023

Linguistic Supervisor's Certification

This is to certify that I have read this Dissertation entitled " New Approaches to Solve Two-sided Optimization Problems in Numerical Analysis " and I found it is qualified for debate.

Signature:

Name:Dr. Haider Ghazi Jassim Al-Moosawi

Title:Professor

Address: English Department, University of Babylon

Date: / / 2023

Examining Committee Certificate

We certify that we have read this Dissertation entitled " New Approaches to Solve Two-sided Optimization Problems in Numerical Analysis " as an examination committee, examined the student " Ammar Imad Nadhim " in its contents and that in our opinion it meets the standard of a dissertation for the Degree of Doctor of Philosophy in Education /Mathematics.

Signature:	Signature:
Name:	Name :
Title:	Title:
Date : / / 2023	Date : / / 2023
Chairman	Member
Signature:	Signature:
Name:	Name :
Title:	Title:
Date : / / 2023	Date : / / 2023
Member	Member / Supervisor

Approved by the Dean of College

Signature:

Name:

Title:

Address: Dean of the College of Education for Pure Sciences

Date : / / 2023

CONTENTS

Notations used in the Thesis **vii**

abstract **vii**

1 Introduction and Related Work **2**

1.1 Introduction . . . 3

1.2 Related Work . . . 4

1.2.1 Optimization Algorithms . . . 4

1.2.2 Numerical Analysis . . . 5

1.2.3 Numerical Methods . . . 6

1.2.4 Numerical Algorithms . . . 7

1.2.5 Numerical Differential Equations . . . 8

1.2.6 Numerical Optimization . . . 9

2 some definition and basic concept **11**

2.1 Convex Set and Convex Function . . . 12

2.2 The Dual of The Primal Problem . . . 20

2.3 The General Differential Equation . . . 24

2.3.1 The Differentiability of the Vector-Valued Functions . . . 30

2.3.2 The Optimality Conditions of Unconstrained Optimization 32

2.3.3 Optimality Conditions for Constrained Optimization 35

3 One-Dimensional And N-Dimensional of Numerical Optimization 36

3.1 Numerical Challenges 37

3.2 Gradient Descent Method 39

3.3 One-Dimensional of Numerical Optimization 42

3.3.1 Unconstraint Optimization 44

3.3.2 Constraint Optimization 50

3.4 Numpy Package in Python 52

3.5 GEKKO Package in Python 53

3.6 Multivariable Optimization 55

3.6.1 Constraint Optimization 56

3.7 Control Optimization 62

3.8 Euler's method 62

3.9 Dynamic Optimization with Linear and Non Linear Models 65

4 Solving mathematical models using numerical optimization methods 71

4.1 Introduction 72

4.2 Numerical Optimization 72

4.3 Python Software 73

4.4 Package Ipopt 74

4.5 Developing Mathematical Models Based on Numerical Analysis 76

4.5.1 Solving NLP of Unconstrained Optimization Problem 76

4.5.2 Solving Dynamic Optimization Model With Multivariable Based on Constraint Differential Equation 79

4.5.3 Solving Dynamic Optimization Model With Initial Value Problem Constraint 85

4.5.4 Solving Nonlinear Numerical Optimization Model with Large Scale Variable 89
4.5.5 The Large Scale Variables in Control Optimization Problem 92
4.5.6 The Large Scale Variables in Control Optimization Problem 97
4.5.7 The Large Scale Variables in Control Optimization Problem 102
4.5.8 The Large Scale Variables in Control Optimization Problem 107
4.6 Mathematical models applications 112
4.7 Conclusions 116
4.8 Future Work 117

LIST OF FIGURES

2.1 Optimization Diagram 13
2.2 Classification of Optimization 13
2.3 Convex Set and Non Convex Set 14
2.4 The General Convex Function 15
2.5 Convex Function 15
2.6 Concave Function 16
2.7 Convex Function 18
2.8 Feasible Region 18
2.9 Local ,Global 19
2.10 Differential Equation 26
2.11 Epigraph 27
2.12 Hypograph 28

3.1 Analytical Approach 38
3.2 Numerical Approaches 38
3.3 Gradient Descent Method 40
3.4 Diagram of Golden Section Search 45
3.5 Numerical Result Base on Golden Search Method 47
3.6 The Result Base on Newton Method 49

3.7 The Result Base on Lagrange Method 51
3.8 Multivariable Optimization with Equality Constraint 60
3.9 The Dynamic Optimization 69

4.1 The Optimal Solution With Time Via value 84
4.2 The Optimal Solution of the Dynamic Model Based on (Time Via Objective Function) 88
4.3 The Optimal Solution of the Nonlinear Model With Large Scale 91
4.4 The Optimal Solution With the Large Scale Variables in Control Optimization Problem 96
4.5 The Optimal Solution With the Large Scale Variables in Control Optimization Using KKT Condtion 100
4.6 The Optimal Solution With Control Optimization Problme Wih Large Scale Variables 106
4.7 The Optimal Solution With Control Optimization Problem Using the Gradient Descent Method 111
4.8 Dynamic of Floating Offshore Wind Turbines 113

LIST OF TABLES

1 **Notations used in the Thesis** x

2 **Notations used in the Thesis** xi

3.1 **Our Results by Using the Gradient Descent Method** 42

3.2 **The Results of Applying the Golden Section Search Method** . . . 46

3.3 **The Results of Solving the Example are the Application of Newton's Method** 49

3.4 **Numerical Result Euler Method** 65

3.5 **Numerical Result Dynamic Optimization** 69

3.6 **Numerical Result Dynamic Optimization** 69

4.1 **Our Results Using the Golden Section Search Method** 78

4.2 **Numerical Result For Dynamic Optimization Model With Multivariable Based on Constraint Differential Equation** 81

4.3 **Numerical Result For Dynamic Optimization Model With Multivariable Based on Constraint Differential Equation** 81

4.4 **Our Results The Dynamic Optimization Model Using The Lagrange Method** 82

4.5 Numerical Result for Solving Dynamic Optimization Model With Initial Value Problem Constraint 87
4.6 Our Results the Dynamic Optimization Model Using the Euler Method 87
4.7 Numerical Result for Solving Dynamic Optimization Model With Initial Value Problem Constraint 88
4.8 Numerical Results For the Large Scale Variables in Control Optimization Problem 94
4.9 Our Results the Control Optimization Model Using the Lagrange Method , where we get the optimal solution by 22 iteration . . . 95
4.10 Numerical Results For the Large Scale Variables in Control Optimization Problem 99
4.11 Our Results the Control Optimization Model Using the KKT Condtion Method ,where we get the optimal solution by 6 iteration 100
4.12 Numerical Results For Solving the Large Scale Variables in Control Optimization Problem 104
4.13 Numerical Results For Solving the Large Scale Variables in Control Optimization Problem 104
4.14 Our Results the Control Optimization Problem Using the KKT Condtion Method, where we get the optimal solution by 21 iteration 105
4.15 Numerical Results For the Large Scale Variables in Control Optimization Problem 109
4.16 Numerical Results For the Large Scale Variables in Control Optimization Problem 109
4.17 Our Results the Control Optimization Problem Using the Gradient Descent Method 110

Dedication

To whom Allah entrust status and dignity.. for my example in life Instilled the love of science for those who allah surround me with a wall his tenderness... whose name I carry with pride... May allah grant you success You will see the fruits whose harvest has come at long last, and your words will remain The stars that guide you today, tomorrow, and forever

(He is my dear father).

To my angel in life.. to the meaning of love, to the meaning of tenderness and Sincerity.. to the grace of life.. the secret of existence.. to whom was her prayer.. The secret of my success and her tenderness is my surgical ointment

(She is my dear mother).

To those in whose presence I gain boundless strength and love... for those with whom He knew the meaning of life, for the one who stood by me and supported me

(My wife and children).

To those who are closest to my soul, my heart and my support in life

(My dear brothers and sisters).

To all my friends and loved ones who have stood with me throughout my academic career, you have my love and gratitude

Ammar 2023

ACKNOWLEDGEMENTS

Praise be to Allah, Lord of the worlds, and prayers and peace be upon the most honorable of creation, Muhammad, and the God of the good, for what the Almighty said. In Surat Ibrahim (And when your Lord declared: “If you are grateful, I will increase you, but if you are not thankful, then My punishment is truly severe.”) Great truth of allah
Praise be to allah Almighty, who bestowed upon me the grace and beauty of His gift and guided me to follow the path of knowing His grace to complete this work. To the one with a big heart, long spirit, and abundant knowledge, who showered me with his kindness, and cared for me with kind guidance and direction.His Excellency
(Dr. Ahmed Sabah Al-Jilawi) who supervises this thesis. I also extend my sincere thanks and appreciation to my family who have supported and helped me throughout my life. I extend my thanks and appreciation to my friends and to everyone who helped us in this research and planted optimism on our path and provided assistance and facilities. They have all the appreciation, before and after. Praise be to allah, and praise be to allah in the first and the last.

Table 1: **Notations used in the Thesis**

$f(x)$	The Objective function
$Min f(x)$	The Minimum of Objective Function
$Max f(x)$	The Maximum of Objective Function
$g(x)$	The Inequality Constraint
$h(x)$	The Equality Constraint
C^1	Continuous Differential
QP	Quadratic programming
LP	Linear Programming
$MILP$	Programming with Mixed-Integers
$MIQP$	Quadratic Programming using Mixed Integers
$MINLP$	The Nonlinear Programming with Mixed-Integers
$DAEs$	The Theory of Differential Equations
$MPCCs$	The Complementarity-Constrained Mathematical Programming Problem
NLP	non linear problems
KKT	Karush-Kuhn-Tucker conditions
$epi(f)$	The epigraph of a function f
$hyp(f)$	The hypograph of a function f
$tr(X)$	The Trace of a matrix X
$\nabla f(x)$	The Gradient of f at x
$\nabla^2 f(x)$	The Hessian matrix of f

Table 2: **Notations used in the Thesis**

$[\nabla f(x)]^T$	The Jacobian of f at x
$minf(x)$	Minimize$f(x)$
$maxf(x)$	Maximize $f(x)$
$s.t$	Subject to
C	Convex set
R	Real Numbers
R^n	Dimensional Space
x^k	The Current Iterate
$x^k + 1$	The Next Iterate

Abstract

Numerical optimization techniques are important tool which can be determine the optimal Solution.The main mission of the study is to develop new algorithms for two-side (constrained and unconstrained nonlinear optimization).This research explores the interaction between different models based on control and dynamical optimization.Different numerical optimization problems require different classes of numerical optimization algorithms for efficient solution.The study focus on one dimensional optimization models and n-dimensional optimization models with their Applications and improved the theoretical convergence properties, APMintor solver Python is used for numerical computation, its free optimization software.The research developed the different classes of modeling of differential optimization problems depend on the dynamical and control systems with large scales variable.

CHAPTER 1

INTRODUCTION AND RELATED WORK

1.1 Introduction

Optimization is the art of finding the best solution from the group.There are many applications for improvement in science, engineering, finance, medicine, economics, and big data. It is generally divided into two subfields: discrete optimization and continuous optimization [23].Maximizing or minimizing a real function is the optimization problem, and this is done by systematically choosing input values from within a permitted set and then calculating the value of the function.There is a large subfield of applied mathematics dedicated to optimization methods for a variety of different formulas. Optimization, in its broadest sense, is the process of determining the "best available" values for a given objective function within a given domain [86].The improvement consists of objective function, which is a set of variables through to find the best solution, and the constraints consist of a set of variables that limit the solution area.To find the best solutions of any problem, if it is constraint, it has special numerical solutions, and if it is unconstraint, it has different numerical methods through algorithms. The variables in the constraints are of one or more dimensions(large scale), and there are also solutions of the problems with control optimization and dynamic optimaztion.The primary action required for optimization.A lack of realism in the model would prevent it from shedding light on the real-world issue. Solve can be challenging to implement if the problem is particularly complex [5]. The solution to the problem can be found by computer using optimization algorithm once the form is formulated. There is no global improvement An algorithm is a set of algorithms [74], Each particular type has its own design in optimization problem. To solve a specific application depends on choosing the correct algorithm ,they are important, as they may be determine whether the problem will be resolved quickly or slowly, After applying the optimization algorithm to the model.Mathematical expressions that define the best possible circumstances for doing this checking exist in many cases, and they can be quite elegant. It turns out that the problem is resolved with the current set of variables. If the conditions are perfect not satisfied [74].

1.2 Related Work

The six main subjects of this work that will mutually aid the efforts to goal:

1 - Optimization Algorithms

2 - Numerical Analysis

3 - Numerical Methods

4 - Numerical Algorithms

5 - Numerical Differential Equations

6 - Numerical Optimization

1.2.1 Optimization Algorithms

The term ”optimization” is used to describe the process of determining the values of parameters or arguments to a function that would yield the best possible or least possible outcome from the function [6]. Continuous function optimization, in which the arguments to the function take the form of real-valued numeric values (e.g., floating point values), is the most prevalent sort of optimization problem faced in machine learning. The function returns a real-valued evaluation of the given arguments. These kind of issues can be distinguished from their discrete-variable counterparts, known as combinatorial optimization problems, by the term ”continuous function optimization.”

Optimization techniques for continuous function optimization issues are diverse, with potentially as many ways to classify and summarize them [60]. As a means of categorizing them, optimization algorithms can be sorted according to the quantity of information they have access to on the goal function [79]. The formulation of optimization algorithms problems varies depending on the problem . They are

(i) Linear constraints and objective function

(ii) Nonlinear constraints and objective function.

Each optimization problem can be solved with the same general optimization algorithm.

Algorithms for optimizing are several categories, which are briefly discussed below.

1- One-variable optimization algorithms.

These algorithms can be down into two distinct groups(direct methods and gradient-based methods). Only the values of the objective function are used to direct the search in direct methods; no derivatives are taken into account.Gradient-based methods make use of derivative (first and/or second order) information to direct the search.

2 - Multivariable (large scale) optimization algorithms.

These algorithms illustrate the n-dimensional nature of the search for optimality. These algorithms can be further categorized into direct and gradient-based approaches based on whether or not the gradient information is used [9].

3 - Algorithms of Constrained optimization.

These algorithms iteratively employ both single- and multi-variable optimization algorithms to keep the search space constrained to be within what's practically possible. Optimization issues in engineering are a common application of these algorithms [73].

4 - Algorithms of Specialized optimization.

Integer programming and geometric programming are two algorithms that come in handy when solving design issues in the engineering field. Optimization issues with integer design variables are amenable to solution via integer programming techniques. Using objective functions and constraints, optimization problems can be solved using a technique called geometric programming [38].

5 - Algorithms of non-traditional optimization.

The nontraditional algorithms divided to two algorithms (algorithms Genetic and Simulated annealing).

1.2.2 Numerical Analysis

Numerical analysis is a field that has been around for a long time and has seen rapid growth and long stagnation periods. Many of these breakthroughs occurred around the time new computer designs first appeared. For example, immediately after the widespread availability of desktop adding machines at the turn of the the beginning of the twentieth century, several basic theories related to arithmetic solutions of Ordinary Differential Equations. The advent of digital computer in Mid twentieth century sparked an uptick in research in many areas, including the solution of partial differential equations and large systems of linear equations. Analysis is always about assessing limits.Thus, it differs from the discrete fields of algebra and mathematics in that it is concerned with infinity.Such as operators in vector spaces, require uniform boundary conditions as a first step. Numerical analysis often uses ”algorithms which are themselves finite notation” to identify objects that are inherently infinite.In most cases to prove that algorithms always work as intended and provide the desired results [18]. As a field, numerical analysis can be somewhat polarized between theoretical and computational courses. Since algorithms are studied for computing with real numbers, numerical analysis can easily be called computational analysis,and the two fields should be combined as perfectly as possible. Many computational algorithms are presented, The software development (also known as programming)is not covered. Math majors should gain some programming experience, as doing so is a lot like proving a theorem. Because computers are so good at spotting logical fallacies, the organization necessary to make programs legible can serve as a model for communicating difficult mathematical ideas [36].

1.2.3 Numerical Methods

In mathematics, the numerical method refers to a specific technique for dealing with numerical data. A scalar algorithm is an algorithmic implementation of a numerical approach that includes proper convergence checks. The term ”numerical methods” often refers to approaches that center around the use of algorithms, perhaps in determine the underlying mathematical theory that underpins the efficacy of these methods; those with the title numerical analysis tend to place a greater emphasis on this underlying mathematical theory, potentially at the expense of some implementation concerns. As with every problem, the optimal method requires a combination of theoretical research and practical application (”Methods”) Look at the mathematical reasoning (”analysis”) behind it. Numerical methods and analysis are often mentioned where a computer is being used for something like airplane design, weather forecasting, or the solution of a difficult scientific or engineering problem [45]. An ever-increasing variety of issues and fields of study necessitate the application of such tools. A common example of this is the employment of mathematics (also known as the substance) in the context of a study. The foundations of today’s financial markets and investment structures can be traced back to the advancements in weather prediction technology. Solving the incredibly complicated equations that govern the universe using numerical approaches and analysis exchange of fluids and heat among and within the atmosphere, oceans, and land numerous approaches exist for dissecting the subject into manageable chunks. algorithms, their development, and their implementation, and Mathematical analysis of algorithms to determine their optimal implementation and performance. All that matters is the most effective method. It is common for two problems to conflict with one another:

• ACCURACY: It is likely that to get an exact result from the calculations very often; in those cases, future occurrences.

1.2.4 Numerical Algorithms

Numerical algorithms are behind the design of shapes (such as the shapes of cars, planes, and lines), computing intensity to display graphics, animate moving objects, study the spread of diseases, model the orbits of planets and satellites, support search engines such as Google and others [15]. More practical math-style problems. The study of numerical algorithm optimization, which includes looking for the points where the function $f(x)$ is maximized or minimized, is the primary focus here. Since improvement problems, also known as variance problems, can be posed for many different types of arithmetic operations, we will use this terminology extensively throughout the development of numerical algorithms. In general about optimization problems it involves find a maximum value of $f(x)$, perhaps topic for constraints defining points $x \in R$ It mentions the systems physical looking for high or low power configurations,Sometimes $f(x)$ is referred to as energy. Numerical algorithms are the oldest algorithms used by the Egyptians. These were used in Greece to find roots in Solving equation. Many other innovations in numerical methods also had their genesis in ancient Greece among its mathematicians. In particular, Archimedes and Eudoxus of Cnidus developed a system for determining the sizing and volume of geometric figures. If use it to find approximations,it find that it follows in the footsteps of contemporary numerical integration; Isaac Newton , Gottfried Leibniz were precursors that played a significant role in the birth of calculus [52]. Calculus of physical reality has led to accurate mathematical models, in engineering, medicine, and other works in the physical sciences. Mathematical models are often so complex that they are difficult to solve explicitly, and efforts to obtain approximate, but very useful, solutions used in numerical analysis. Replace the tedious multiplication and division logarithms After converting the original values into logarithms with their corresponding simple addition and subtraction through special [37].

1.2.5 Numerical Differential Equations

Differential equations are an essential tool in a wide variety of applications. The reason for this is that many phenomena can be modeled by the relationship between a function and its derivatives. Differential equations describe almost all systems such as engineering and other fields as diverse as economics, sociology, biology, business, healthcare, etc. There are a large number of mathematicians in this world. has been studying these equations for hundreds of years and there are many complex solution techniques. In most cases, Since a purely analytical solution to the differential equations cannot be found when the systems they describe are either very complex or very large, differential equations are often used to describe systems that are intractable to solve numerically.

There is a need for computer simulations and numerical methods in these kinds of complex systems. on the basis of numerical approximation techniques for solving differential equations have been developed by Programmable computers. In world war II, rooms for people (usually women) were found [19]. Work use of mechanical calculators for military numerical solution of systems of differential equations arithmetic. Before programmable computers, comparisons of electrical systems to analog design were also common A computer for studying mechanical, thermal or chemical systems [47]. With the increase in the speed of programmable computers less expensive and Simple computer programs can solve complex systems of differential equations.Another reason for the popularity of modeling with differential equations is that such equations can usually be solved effectively. For some equations it is possible to find an explicit form of the unknown function,but this is rare. For a wide range of equations, it is possible to calculate the good approximation to solution by numerical algorithms,numerical methods for differential equations are euler methods forward euler methods, reverse euler method, modified euler method, Runge-Kutta Methods, [39]

1.2.6 Numerical Optimization

Numerical optimization is defined as minimizing(maximizing) an evaluation scale of a so-called ”objective function” with several design limitations. It is one of the central techniques used in machine learning. There are many problems, it is difficult to directly find the best solution, but it is relatively easy to find a loss function that measures how good the solution is and then reduce the parameters of that function to find the solution. Optimization problems refer to finding the optimal solution among all possible solutions finding the best answer out of many alternatives. When applied to design challenges, optimization yields new insights. By definition, an optimization problem is a computational example in which the goal is to identify the best possible answers. The model includes data about the limitations and hidden costs of discretionary laws and policy decisions, allowing for more accurate comparisons with the current system. Analysis is supported by a well-designed improvement model, showing where improvements are being made or where trade-offs may be necessary. Optimization techniques are used in many fields of engineering. The optimal data model is selected from a set of candidates using a mathematical method. The constraints, such as the values allowed for a function, define the problem domain. Finding the best way to do a job requires a thorough review of the options available. The minimum error is zero because optimal solutions have very little of it. Every optimization problem is unique [83].Problems with simple solutions or with no decision variables usually do not require elaborate optimization techniques. However, most situations require a mathematical answer if one wants to get the best results. Most problems can be fixed by making some kind of modification [12]. The goal is lower problem cost and associated risks. It is possible to have multiple goals. The objective, variables and constraints are the three basic parts of an optimization problem. The goal is to determine the best possible value for each variable.

CHAPTER 2
SOME DEFINITION AND BASIC CONCEPT

In this chapter, the optimization problems are defined , lay out the notation used to describe it, and present the necessary solutions to any such problems that may arise.The introduction to the fundamental ideas and data that underpin rigorous mathematical analysis are provided.

2.1 Convex Set and Convex Function

Definition 2.1.1. [87]

Optimization Problem is a mathematical problem in which the goal is to find the best solution out of a set of solutions in feasible region, which has the minimum (or maximum) value of the objective function.

$$\begin{cases} \text{minimize} & f(x) \\ \text{subject to} & h_i(x) \le 0, \quad i = 1, \dots, I, \\ & g_j(x) = 0, \quad j = 1, \dots, E, \\ & x \in X. \end{cases} \tag{2.1}$$

where f, g_i and h_i defined from $X \subseteq \mathbb{R}^n$ into $\mathbb{R}$ are assumed to be continuously differentiable functions

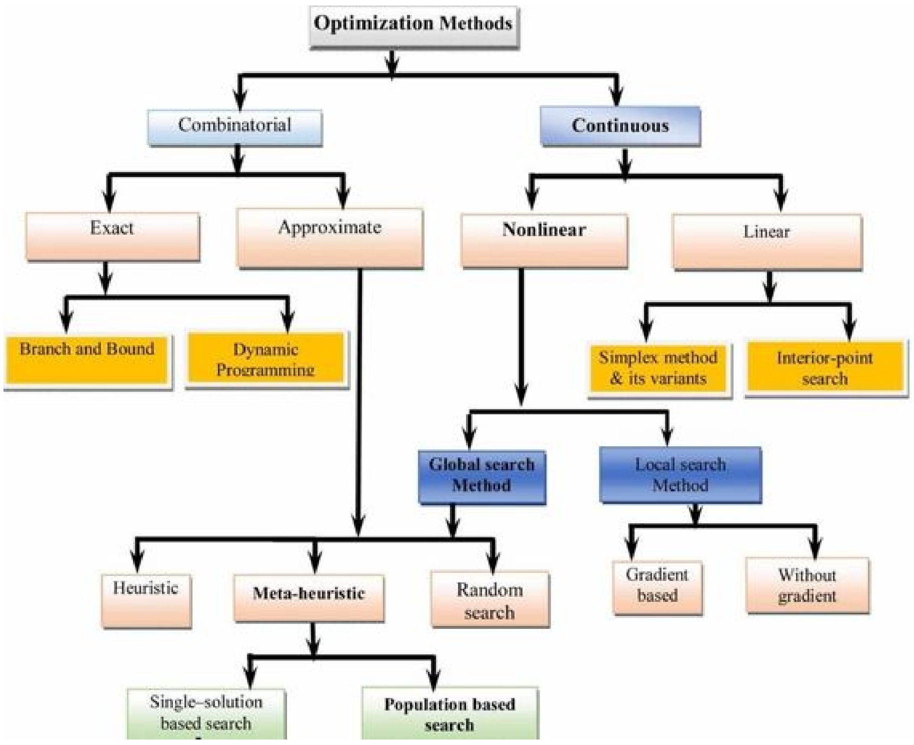


Figure 2.1: Optimization Diagram

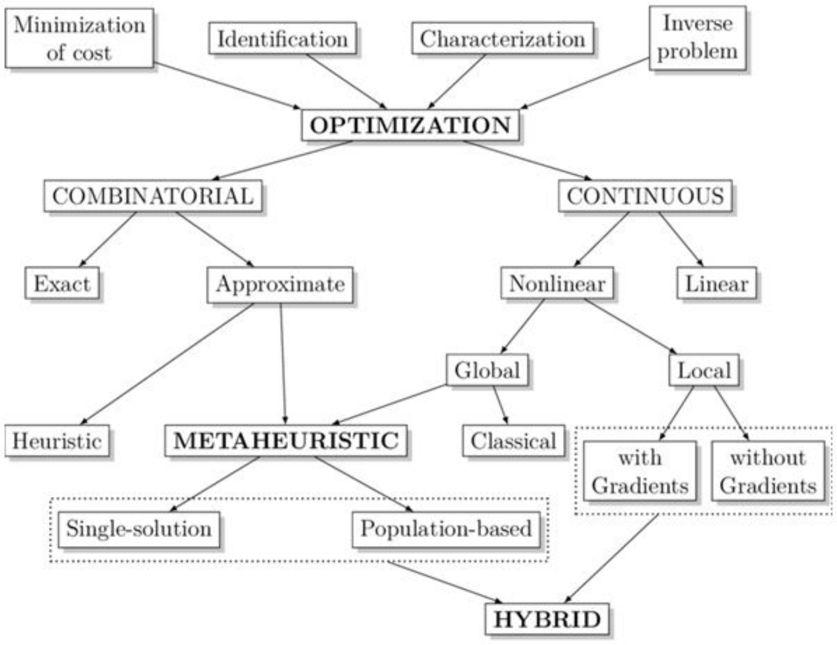


Figure 2.2: Classification of Optimization

Definition 2.1.2. [14] Convex Set :If the line segment between any two points in C ,lies in C.Then the set C is Convex Set

$$\lambda x_1 + (1-\lambda)x_2 \in C \quad , \forall x_1, x_2 \in C, \, \forall \lambda \in [0,1]$$

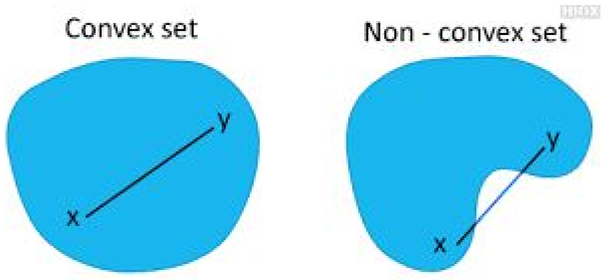


Figure 2.3: Convex Set and Non Convex Set

Example 2.1.1. $\Omega_1 = \{z \in R^n : \|z\| \leq 1\}$
For any $z, u \in \Omega_1, \lambda \in (0,1)$ *then* $\|z\| \leq 1$, $\|u\| \leq 1$
we have

$$\|\lambda z + (1-\lambda)u\| \leq \|\lambda z\| + \|1-\lambda)u\| = \lambda\|z\| + (1-\lambda)\|u\| \leq \lambda(1) + (1-\lambda)1 = 1$$

Definition 2.1.3. [14] Convex Function: A function $f : R^n \longrightarrow R$ is **Convex**, if for every $x_1, x_2 \in R^n$, $0 \leq t \leq 1$ the inequality

$$f(t\ x_1 + (1-t)\ x_2) \leq tf(x_1) + (1-t)\ f(x_2)$$

If the above inequality is true as a strict inequality for all $x_1 \neq x_2$ and for all $t \in (0,1)$, then f is called a strictly convex function

Definition 2.1.4. [14] Concave Function: A function $f : R^n \longrightarrow R$ is **Concave**, if for every $x_1, x_2 \in R^n$, $0 \leq t \leq 1$ the inequality

$$f(t\ x_1 + (1-t)\ x_2) \geq tf(x_1) + (1-t)\ f(x_2)$$

▷ f is convex → - f is Concave

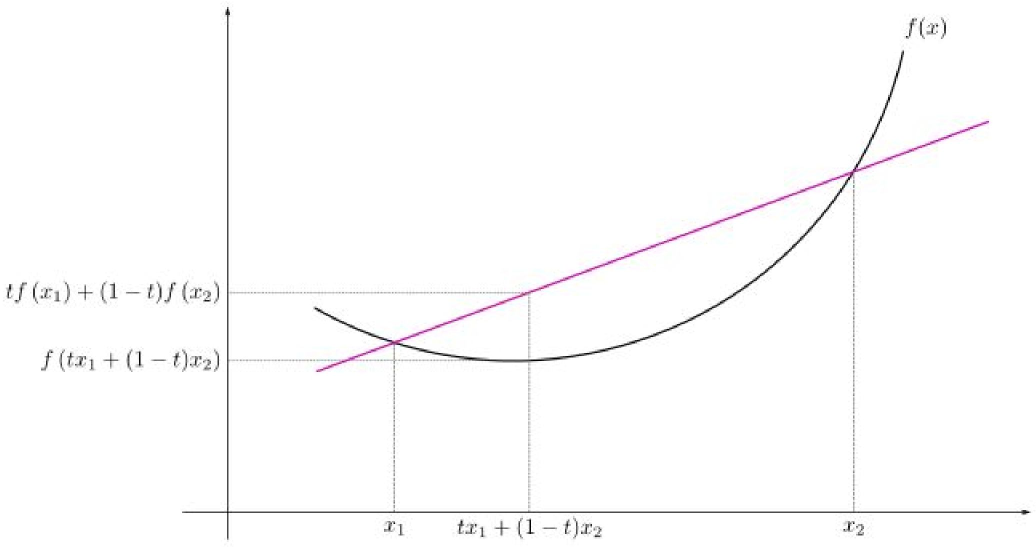


Figure 2.4: The General Convex Function

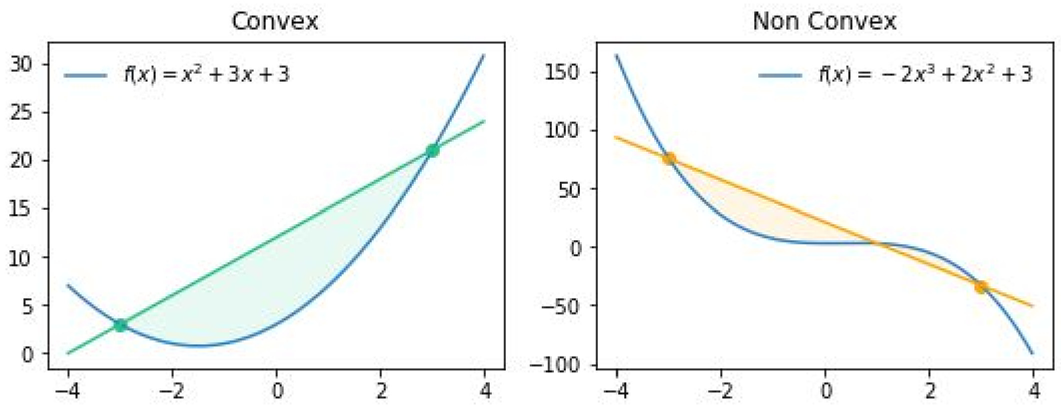


Figure 2.5: Convex Function

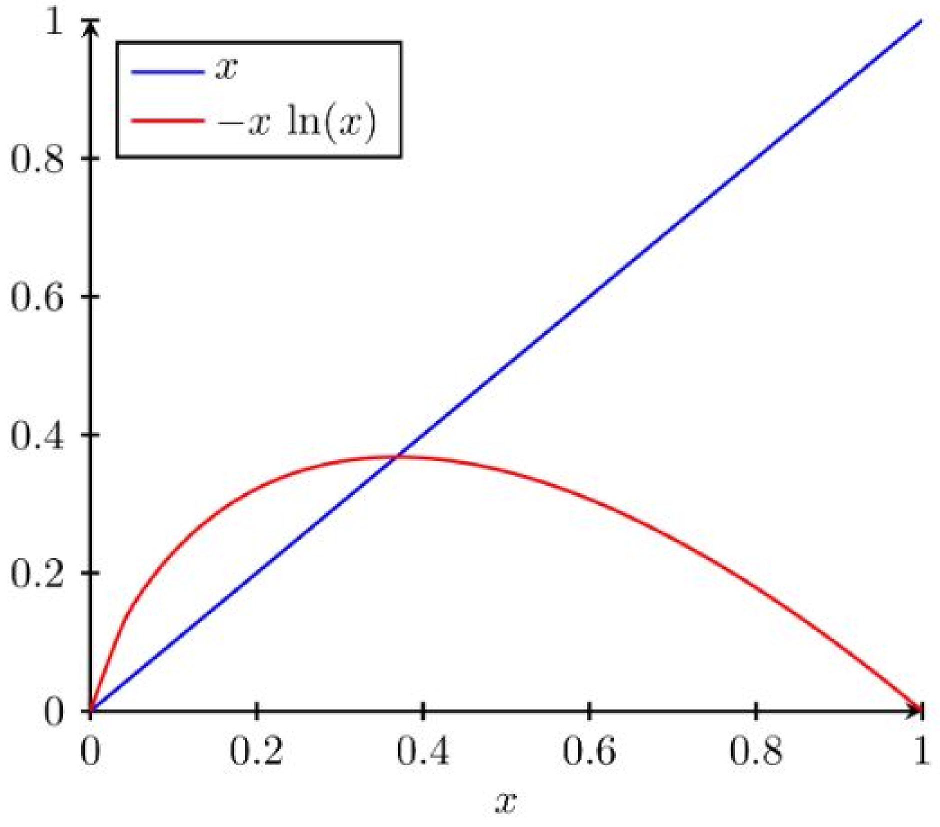
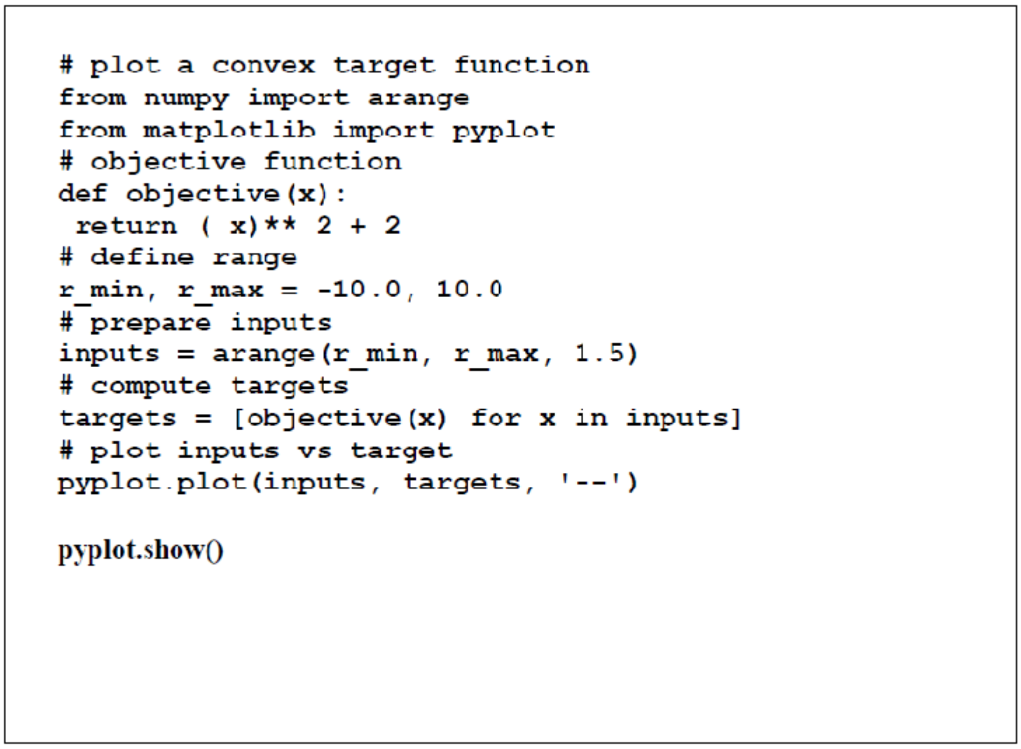


Figure 2.6: Concave Function

Example 2.1.2. Using Python to prove the function is convex function

$$f(x) = x^2 + 2$$

Code 1:**Use Python to prove the function is convex function**



```
# plot a convex target function
from numpy import arange
from matplotlib import pyplot
# objective function
def objective(x):
 return ( x)** 2 + 2
# define range
r_min, r_max = -10.0, 10.0
# prepare inputs
inputs = arange(r_min, r_max, 1.5)
# compute targets
targets = [objective(x) for x in inputs]
# plot inputs vs target
pyplot.plot(inputs, targets, '--')

pyplot.show()
```

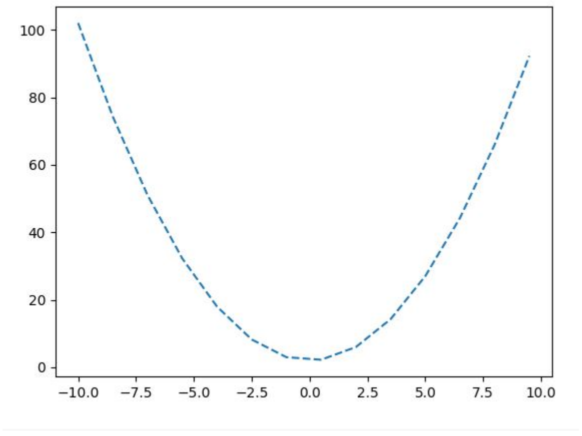


Figure 2.7: Convex Function

Definition 2.1.5. [68] Feasible Region : is the set of all possible points (sets of values of the choice variables) of an optimization problem that satisfy the problem's constraints and the best solution which gives the maximum or the minimum objective function is optimal solution.

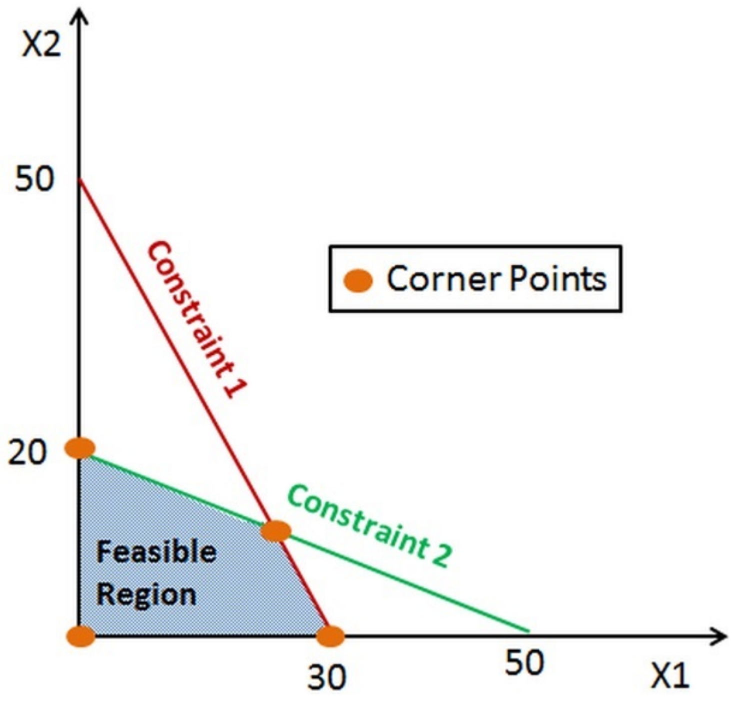


Figure 2.8: Feasible Region

Definition 2.1.6. [84] Let $f\colon \mathbb{R}^n \to \mathbb{R}$ and $x^* \in \mathbb{R}^n$. Then x^* is called a local minmimizer of f if there is a scalar $t > 0$ such that $f(x^*) \le f(x)$ for all $x \in \mathcal{B}(x^*, t) = \{x \in \mathbb{R}^n \mid \|x - x^*\| \le t\}$. We call x^* a strict local minmimizer of f if there is a scalar $t > 0$ such that $f(x^*) < f(x)$ for all $x \neq x^*$ such that $x \in \mathcal{B}(x^*, t)$.
We call x^* a global minimizer of f if $f(x^*) \le f(x)$ for all $x \in \mathbb{R}^n$. Finally, we say that x^* is a strict global minimizer of f if $f(x^*) < f(x)$ for all $x \neq x^*$

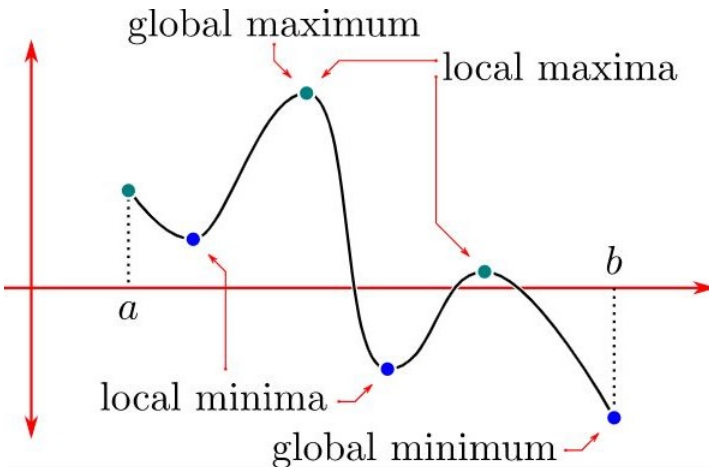


Figure 2.9: Local ,Global

The figure contains a local maximum and a local minimum. In the middle green point the global maximum occurs (which is also a local maximum), while in the blue point the rightmost global minimum occurs (which is not a local minimum).

2.2 The Dual of The Primal Problem

Duality is essential in optimization. The term dual problem usually refers to the dual Lagrangian problem. Lagrangian duality theory is about solving an optimization problem by finding a bound. Lagrangian duality gives lower or upper bounds to the original problem and can be used to evaluate how far we are from optimality [14]. The standard form of the nonlinear optimization problem is given by

$$(P)\begin{cases} \text{minimize} & f_0(x) \\ \text{subject to} & f_i(x) \leq 0, \quad i = 1, \ldots, I, \\ & g_i(x) = 0, \quad i = 1, \ldots, E, \\ & x \in X. \end{cases} \tag{2.2}$$

We define the Lagrangian function as

$$\mathcal{L}(x, \beta, \gamma) = f_0(x) + \sum_{i=1}^{I} \beta_i f_i(x) + \sum_{i=1}^{E} \gamma_i g_i(x),$$

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

$$\psi(\beta, \gamma) = \min_{x \in X} \ \mathcal{L}(x, \beta, \gamma).$$

We denote the optimal value of the Lagranian dual problem by d^*, the optimal value of primal problem by p^*, and x^* is an optimal solution of (P). The Lagrange dual problem with dual variables (β, γ) is given by

$$(D)\begin{cases}\text{maximize} & \psi(\beta,\gamma)\\ \text{subject to} & \beta \geq 0.\end{cases} \tag{2.3}$$

The dual function provide lower bounds on the optimal value p^* of the problem (2.2) ; that is, for any $\beta \geq 0$ and any γ we have

$$\psi(\beta,\gamma) \leq p^*.$$

Definition 2.2.1. [70] KKT Conditions Method (Karush-Kuhn-Tucker Conditions)
KKT Conditions method,which describe the optimality of a constrained local optimum, are pivotal in the study and design of optimization algorithms with constraints. For a solution to be optimal in a mathematical optimization problem,it must meet a set of requirements known as the KKT conditions of optimal. In nonlinear programming,they are both required and sufficient for a local minimum to exist.
Given general problem

$$\begin{cases} \text{minimize} & f(x) \\ \text{subject to} & h_i(x) \le 0, \quad i = 1, \dots, M, \\ & g_j(x) = 0, \quad j = 1, \dots, R, \\ & x \in R. \end{cases}$$

The KKT Conditions are

1 − $\quad 0 \in \partial f(x) + \sum_{i=1}^{m} u_i \partial h_i(x) + \sum_{j=1}^{r} v_j \partial g_j(x) \quad (stationarity)$

2 − $\quad u_i h_i(x) = \; 0 \;\; for\ all \;\; i \quad (complementary\ slackness)$

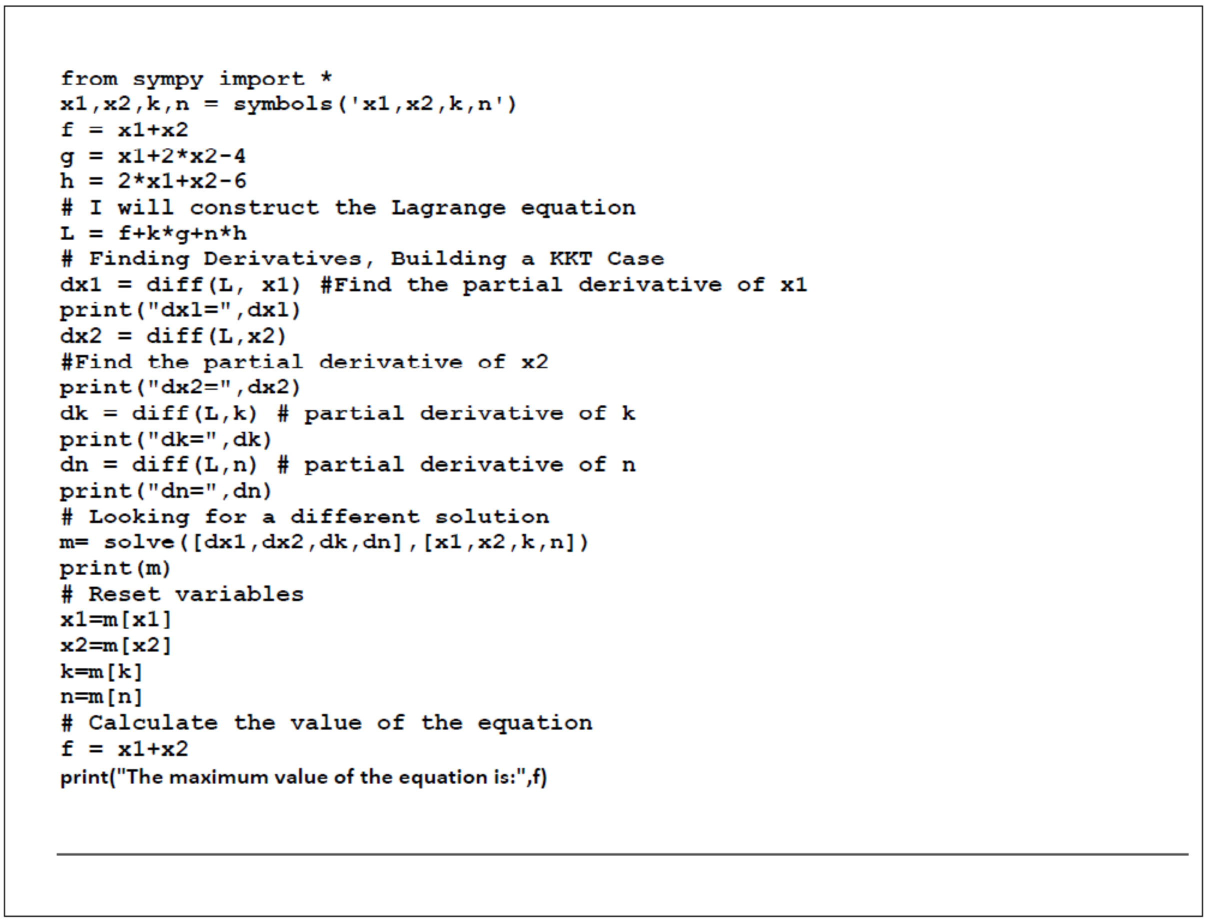
3 − $\quad h_i(x) \le 0, g_j = \; 0 \;\; forall \;\; i, j \quad (primal\ feasibility)$

4 − $\quad u_i \ge 0 \;\; for\ all \;\; i \quad (dual\ feasibility)$

Example 2.2.1. Find the optimal solution to constrained optimization problem by using KKT condition

$$\begin{cases} \text{minimize} & f(x) = x_1 + x_2 \\ \text{subject to} & g(x) = x_1 + 2x_2 - 4 \\ & h(x) = 2x_1 + x_2 - 6 \\ & x \in R. \end{cases}$$

Code 2:Using KKT method



```
from sympy import *
x1,x2,k,n = symbols('x1,x2,k,n')
f = x1+x2
g = x1+2*x2-4
h = 2*x1+x2-6
# I will construct the Lagrange equation
L = f+k*g+n*h
# Finding Derivatives, Building a KKT Case
dx1 = diff(L, x1) #Find the partial derivative of x1
print("dx1=",dx1)
dx2 = diff(L,x2)
#Find the partial derivative of x2
print("dx2=",dx2)
dk = diff(L,k) # partial derivative of k
print("dk=",dk)
dn = diff(L,n) # partial derivative of n
print("dn=",dn)
# Looking for a different solution
m= solve([dx1,dx2,dk,dn],[x1,x2,k,n])
print(m)
# Reset variables
x1=m[x1]
x2=m[x2]
k=m[k]
n=m[n]
# Calculate the value of the equation
f = x1+x2
print("The maximum value of the equation is:",f)
```

the numerical result are

$dx_1 = k + 2n + 1$

$dx_2 = 2k + n + 1$

$d_k = x_1 + 2x_2 - 4$

$d_n = 2x_1 + x_2 - 6$

$k : -1/3, n : -1/3, x_1 : 8/3, x_2 : 2/3$

The maximum value of the equation is:10/3

2.3 The General Differential Equation

A differential equation is an equation that includes the derivatives of one or more functions. At any given point, a function's derivative represents its rate of change. Applications in physics, engineering, biology, etc. abound. Differential equations are used primarily for the study of solutions that satisfy the equations and the properties of solutions are equations that involve the derivatives of one or more functions. Typically, the functions stand in for physical quantities, while the derivatives of these functions reveal the rates of change, and the differential equation shows the connection between them [19]. Are the easiest approach to compute numbers (x) A differential equation is the formal name for this type of formula. Derivatives, either partial or ordinary, can be found in a differential equation. Differential equations show the connection between two quantities, one of which is continually altering with regard to the other, and the derivative stands for the rate of change it reflects. Differential equations formulas to discover derivative solutions are many. Among the many categories into which differential equations can be categorized

1- Ordinary differential equations.

2- Equations with Partial Differentiation

3- Equations of Linear Differential Form

4 - Differential equations that are nonlinear

5- Equations with Homogeneous Variables

6- Differential Equations in Non homogeneous Media and What determines the order of a differential equation is the highest order derivative included in it.

1 - Differential Equation of the First Order

2 - Differential Equation of the Second Order

Applications of differential equations vary from classical mechanics, electrodynamics, general relativity, and quantum physics to chemistry, biology, and even economics. Therefore, modern scientific studies place a premium on having a solid grasp of differential equations.

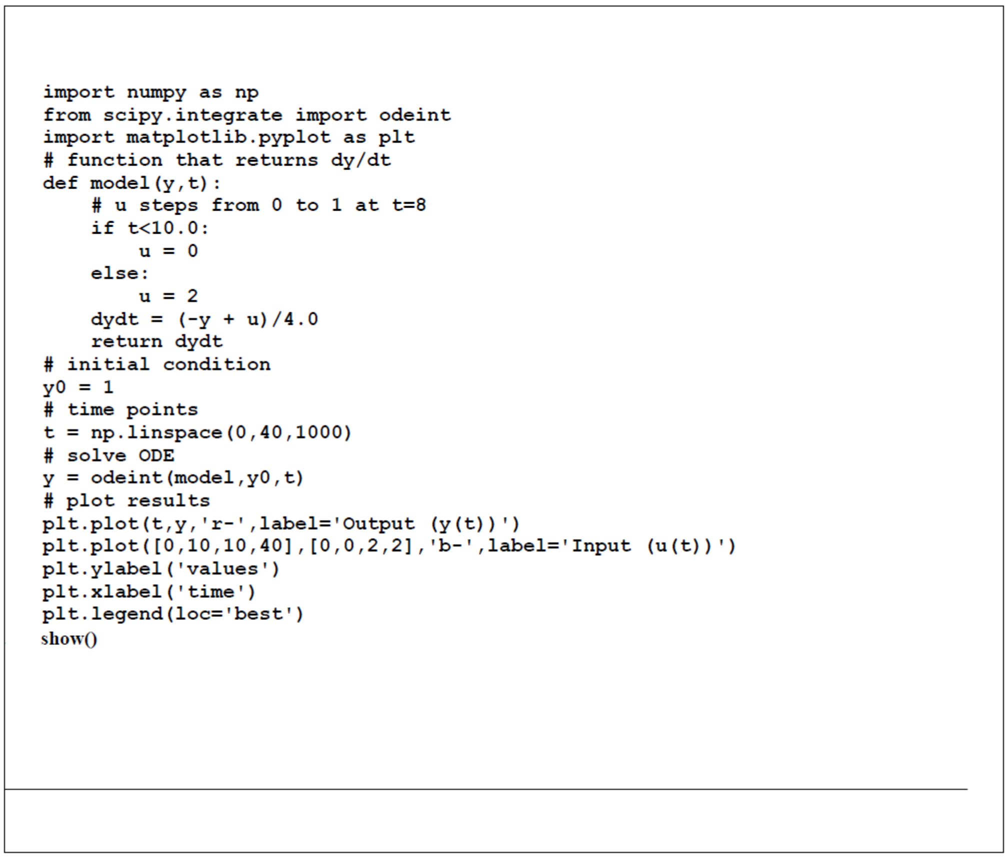
Differential Equation on general form:

$$\frac{dx}{dy} = f(t, y), \quad y(t_0) = y_0 \;\; is\ initial \;\; condition$$

Example 2.3.1. : Solving the defferential equation with initial Conditions

$$\begin{cases} 4\dfrac{dy(t)}{dt} = -y(t) + u(t) \\ y(0) = 1 \\ \text{u steps from } 0 \ to \ 1 \ at \ \ t = 8 \end{cases}$$

Code 3:Solving differential equation with initial condition



```
import numpy as np
from scipy.integrate import odeint
import matplotlib.pyplot as plt
# function that returns dy/dt
def model(y,t):
    # u steps from 0 to 1 at t=8
    if t<10.0:
        u = 0
    else:
        u = 2
    dydt = (-y + u)/4.0
    return dydt
# initial condition
y0 = 1
# time points
t = np.linspace(0,40,1000)
# solve ODE
y = odeint(model,y0,t)
# plot results
plt.plot(t,y,'r-',label='Output (y(t))')
plt.plot([0,10,10,40],[0,0,2,2],'b-',label='Input (u(t))')
plt.ylabel('values')
plt.xlabel('time')
plt.legend(loc='best')
show()
```

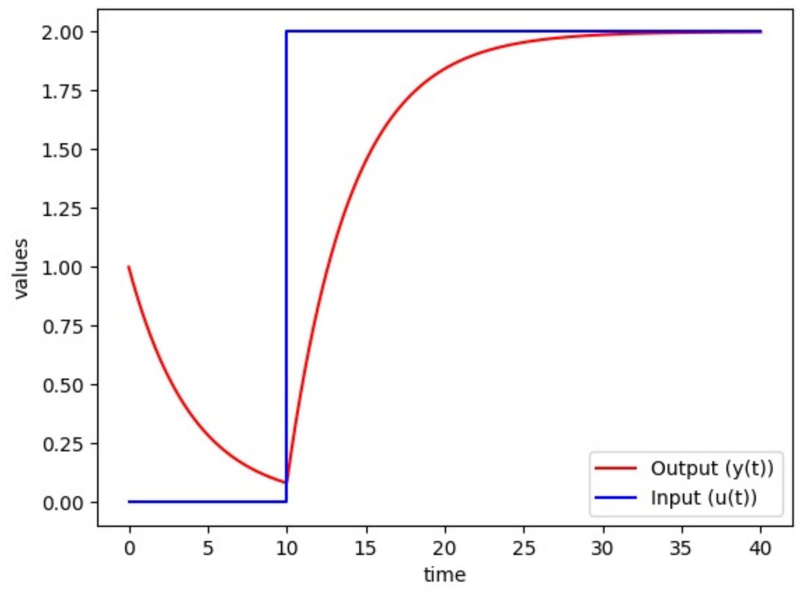


Figure 2.10: Differential Equation

In this figure (2.10) we plot the values of $u(t)$ which represents the blue line (input), $y(t)$ which represents the red line (exit mode) where we start with the point$(0,0)$ and the initial condition $y(0) = 1$, Using Python, the differential equation has been solved

Definition 2.3.1. [82] A function $\langle \cdot, \cdot \rangle : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ is an Inner Product if

(i) $\langle z, z \rangle > 0$, $\langle z, z \rangle = 0 \Leftrightarrow z = 0$ (positivity).

(ii) $\langle z, u \rangle = \langle u, z \rangle$ (symmetric).

(iii) $\langle z + u, k \rangle = \langle z, k \rangle + \langle u, k \rangle$ (additivity).

(iv) $\langle \alpha z, u \rangle = \alpha \langle z, u \rangle$ for all $\alpha \in \mathbb{R}$ (homogeneity).

Definition 2.3.2. [22] A function $\| \cdot \| : \mathbb{R}^n \to \mathbb{R}$ is Norm, If the properties are met

(i) $\|z\| \geq 0$, $\forall z \in \mathbb{R}^n$; $\|z\| = 0$ if and only if $z = 0$.

(ii) $\|\lambda z\| = |\lambda| \|z\|$, $\forall \lambda \in \mathbb{R}$, $\forall z \in \mathbb{R}^n$.

(iii) $\|z + u\| \leq \|z\| + \|u\|$, $\forall z, u \in \mathbb{R}^n$.

The Euclidean Norm is Defined

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

- The Inner product of any two real n-vectors z and y is defined by

$$\langle z, u \rangle = z^T u = \sum_{i=1}^{n} z_i u_i = z_1 u_1 + z_2 u_2 + \cdots + z_n u_n.$$

- The Cauchy-Schwarz Inequality: For vectors x and y in $\mathbb{R}^n$, we have

$$|\langle z, u \rangle| \leq \|z\| \|u\|.$$

Definition 2.3.3. [14] let w which is nonemptey convex set .the Epigraph of function $f : W \longrightarrow R$, denoted by $epi(f)$,is a subset of R^n defined by

$$epi(f) = \big\{(x, \gamma) \bigm| f(x) \leq \gamma, x \in W, \gamma \in \mathbb{R}\big\}.$$

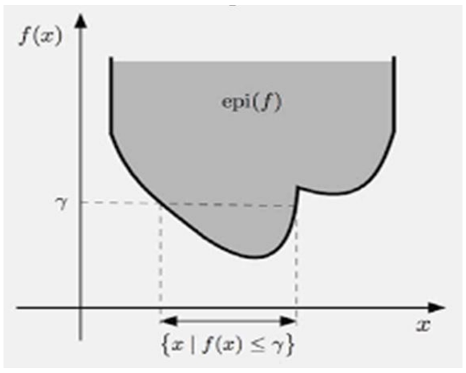


Figure 2.11: Epigraph

The $hyp(f)$ denoted to Hypograph and is sub set of R^{n+1} defined by

$$hyp(f) = \{(z,k) \mid f(z) \geq k, z \in W, k \in \mathbb{R}\}.$$

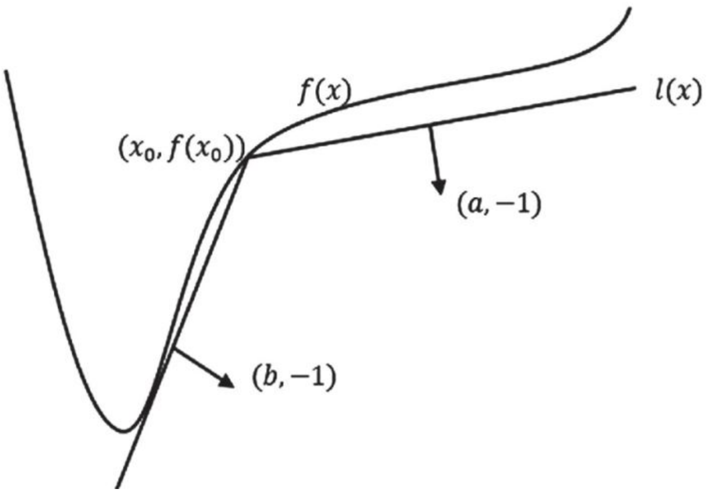


Figure 2.12: Hypograph

Definition 2.3.4. [2] Let $W \subseteq \mathbb{R}^n$ be a nonempty set . If $f\colon W \to \mathbb{R}$, the Argument of the minimum is the set of elements in W that achieve the global minimum in w, which is defined by

$$\text{argmin}_{x \in W} f(x) = \{x \in W \mid f(y) \geq f(x),\ \forall y \in W\}.$$

Definition 2.3.5. [2] suppose $W \subseteq \mathbb{R}^n$ which is nonempty set . If $f\colon W \to \mathbb{R}$, the Argument of the maximum The set of elements in W by which the global minimum is satisfy in W, which is defined by

$$argmax f(x) = \{x \in W \mid f(y) \leq f(x),\ \forall y \in W\}.$$

Definition 2.3.6. [64] A Linear Transformation (**linear map**) is a function $L\colon \mathbb{R}^n \to \mathbb{R}^m$ that satisfies the following properties:

$$L(z+u) = L(z) + L(u) \quad \text{and} \quad L(\alpha z) = \alpha L(z),$$

for any vectors $z, u \in \mathbb{R}^n$ and any scalar $\alpha \in \mathbb{R}$.

Definition 2.3.7. [4] The Trace of a matrix $X \in \mathbb{R}^{n\times n}$ is defined by

$$tr(X) = \sum_{i=1}^{n} X_{ii}.$$

Example 2.3.2. Find the trace for matrix (A)

$$A = \begin{bmatrix} 4 & 3 & 2 \\ 5 & -2 & 6 \\ -3 & 4 & 8 \end{bmatrix}$$

tr(A) = 4 -2 + 8 =10

The function $tr : R^{n\times n} \to R$ is linear; that is, for any two $n \times n$ matrices X and Y and two scalars α and β, we have

$$tr(\alpha X + \beta Y) = \sum_{i=1}^{n} \left(\alpha X_{ii} + \beta Y_{ii}\right) = \alpha tr X + \beta tr Y.$$

For any two symmetric matrices $X, Y \in S^n$, the Frobenius inner product by

$$< X, Y >= \sum_{i=1}^{n} \sum_{j=1}^{n} X_{ij} Y_{ij} = tr(XY).$$

Definition 2.3.8. [1]

Lower bound (LB) is the lowest number that can be rounded to get an estimated value.And the highest number that can be rounded to get an estimated value is upper bound (UB).

Definition 2.3.9. [81]

CPU Time is the exact amount of time that the (cpu) has spent processing data for a specific program or process.

2.3.1 The Differentiability of the Vector-Valued Functions

In this Section, we introduce the definition of the differentiability and derivative of a scalar-valued function

Definition 2.3.10. [74] Let $f\colon \mathbb{R}^n \to \mathbb{R}$ and $x \in \mathbb{R}^n$. Then the Partial derivative of f at x with respect to x_i is defined as

$$\frac{\partial f(x)}{\partial x_i} = \lim_{t\to 0} \frac{f(x+te_i) - f(x)}{t},$$

where e_i is ith unit vector. The Gradient of f at x is defined as the column vector

$$\nabla f(x) = \begin{bmatrix} \dfrac{\partial f(x)}{\partial x_1} \\ \vdots \\ \dfrac{\partial f(x)}{\partial x_n} \end{bmatrix}.$$

The Hessian matrix is defined as the $n \times n$ symmetric matrix

$$\nabla^2 f(x) = \begin{bmatrix} \dfrac{\partial^2 f_1}{\partial x_1 \partial x_1} & \dfrac{\partial^2 f_1}{\partial x_1 \partial x_2} & \cdots & \dfrac{\partial^2 f_1}{\partial x_1 \partial x_n} \\ \dfrac{\partial^2 f_2}{\partial x_2 \partial x_1} & \dfrac{\partial^2 f_2}{\partial x_2 \partial x_2} & \cdots & \dfrac{\partial^2 f_2}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \dfrac{\partial^2 f_n}{\partial x_n \partial x_1} & \dfrac{\partial^2 f_n}{\partial x_n \partial x_2} & \cdots & \dfrac{\partial^2 f_n}{\partial x_n \partial x_n} \end{bmatrix}.$$

The Directional derivative of the function f at x in the direction d given by

$$f'(x,d) = \lim_{x\to 0^+} \frac{f(x+td)-f(x)}{t}.$$

We say the function f is Differentiable at x.

Definition 2.3.11. [67] Let $f\colon \mathbb{R}^n \to \mathbb{R}$ be Differentiable at $x \in \mathbb{R}^n$.The $d \in \mathbb{R}^n$ is a Descent Direction of the function f at x if

$$\langle \nabla f(x), d\rangle < 0.$$

Theorem 2.3.1. *(Taylor Expansion) [85] Let $f\colon \mathbb{R}^n \to \mathbb{R}$ differentiable be continuously . Then, for all $z_1, z_2 \in \mathbb{R}^n$, there exist $\alpha \in [0,1]$, such that*

$$f(z_2) = f(z_1) + \nabla f(\alpha z_1 + (1-\alpha)z_2)^T(z_2 - z_1).$$

Furthermore, if f is twice differentiable continuously , then, for all $z_1, z_2 \in \mathbb{R}^n$, there exist $\alpha \in [0,1]$, such that

$$f(z_2) = f(z_1) + \nabla f(z_1)^T(z_2 - z_1) + \frac{1}{2}(z_2 - z_1)^T \nabla^2 f(\alpha z_1 + (1-\alpha)z_2)(z_2 - z_1).$$

In addition, if $z, u \in \mathbb{R}^n$ and $\eta \in \mathbb{R}$, we have

$$f(z+\eta u) = f(z) + \eta u^T \nabla f(z) + \frac{\eta^2}{2} u^T \nabla^2 f(z) u + o(\eta^2) \ \ \textit{as}\ \eta \to 0.$$

Definition 2.3.12. [69] A function $f\colon \mathbb{R}^n \to \mathbb{R}^m$, with component functions $f_1, \dots, f_m$, is called Differentiable If each Component is Differentiable. The Gradient Matrix of f, called $\nabla f(x)$, also the $n \times m$ Matrix is ith Column of the Gradient $\nabla_i f(x)$ of f_i:

$$\nabla f(x) = \big[\nabla f_1(x) \cdots \nabla f_m(x)\big].$$

Then the Jacobian of f at x is defined as

$$D(x) = [\nabla f(x)]^T = \begin{bmatrix} \nabla f_1(x)^T \\ \vdots \\ \nabla f_m(x)^T \end{bmatrix} = \begin{bmatrix} \frac{\partial f_1(x)}{\partial x_1} & \dots & \frac{\partial f_1(x)}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m(x)}{\partial x_1} & \dots & \frac{\partial f_m(x)}{\partial x_n} \end{bmatrix}.$$

Definition 2.3.13. [54]

Quadratic Programming (QP) is a qadratic ojective Function optimization problem, and is one of the forms of nonlinear programming. An objective function can have terms of a binary or even quadratic polynomial, and the equality (inequality)constraints are linear.

The general quadratic programming formulation contains a quadratic objective function and linear equality and inequality constraints

$$\begin{cases} \text{minimize} & f(x) = q^T x + \frac{1}{2} x^T Q x \\ \text{subject to} & Ax = a \\ & Bx \le b \\ & x \ge 0 \end{cases}$$

2.3.2 The Optimality Conditions of Unconstrained Optimization

In this study, the problem for optimizations with unconstrained If $X = \mathbb{R}^n$, i.e., The minimize f unconstraints , it can write it as:

$$\underset{x \in \mathbb{R}^n}{\text{minimize}} \quad f(x). \tag{2.4}$$

- If f is continuously differentiable, then a necessary condition for $x^* \in \mathbb{R}^n$ to be a solution of problem is

$$\nabla f(x^*) = 0.$$

- The sufficient conditions for $x^* \in \mathbb{R}^n$ to be a local solution of problem are

$$\nabla f(x^*) = 0, \ \ \nabla^2 f(x^*) > 0.$$

Definition 2.3.14. [30] Let $X = [x_{ij}]$ which is square matrix. So the matrix X is said Symmetric if $X = X^T$ (i.e., $x_{ij} = x_{ji}$). Let S^n be of all the symmetric $n \times n$ matrices: The notation $X \succ 0$ ($X \succeq 0$) which is X said to be symmetric and Positive definite (semidefinite); for all $u \in \mathbb{R}^n$ ($u^T X u = \sum_{ij} X_{ij} u_i u_j \geq 0$ for all $u \in \mathbb{R}^n$).

Theorem 2.3.2. *(First-Order Necessary Condition) [11, 27] Let $f \colon \mathbb{R}^n \to \mathbb{R}$ be differentiable. If x^* is a local minimizer of f, then $\nabla f(x^*) = 0$.*

Proof: Define $h \colon \mathbb{R} \to \mathbb{R}$ as $h(\eta) = f(x^* + \eta u)$ for some $u \in \mathbb{R}^n$, then $h'(\eta) = u^T \nabla f(x^* + \eta u)$. If $\eta = 0$, we get $h'(0) = u^T \nabla f(x^*)$. By definition,

$$h'(0) = \lim_{\eta \to 0} \frac{f(x^* + \eta u) - f(x^*)}{\eta}.$$

Since x^* is a local minimizer, there exists $t > 0$, such that $f(x^* + \eta u) \geq f(x^*)$ for all $0 < \eta \leq t$, therefore we get $u^T \nabla f(x^*) \geq 0$. let u is arbitrary, so the change u by $-u$, and therefore $-u^T \nabla f(x^*) \geq 0$. Therefore, $u^T \nabla f(x^*) = 0$, for all $u \in \mathbb{R}^n$. Thus, $\nabla f(x^*) = 0$.
■

Theorem 2.3.3. *(Second-Order Necessary Condition) [11, 27] Let $f \colon \mathbb{R}^n \to \mathbb{R}$ be twice differentiable. If x^* is a local minimizer of f, then $\nabla f(x^*) = 0$ and $\nabla^2 f(x^*)$ is positive semidefinite.*

Proof: Let $u \in \mathbb{R}^n$. We want to show that $u^T \nabla^2 f(x^*) u \geq 0$. By using Taylor expansion of f at x^*, we obtain

$$f(x^* + \eta u) = f(x^*) + \eta u^T \nabla f(x^*) + \frac{\eta^2}{2} u^T \nabla^2 f(x^*) u + o(\eta^2).$$

Since $\nabla f(x^*) = 0$ by First-Order Necessary Condition theorem, we have

$$f(x^* + \eta u) = f(x^*) + \frac{\eta^2}{2} u^T \nabla^2 f(x^*) u + o(\eta^2).$$

Dividing by both sides by η^2, we have

$$\frac{f(x^* + \eta u) - f(x^*)}{\eta^2} = \frac{1}{2} u^T \nabla^2 f(x^*) u + \frac{o(\eta^2)}{\eta^2}.$$

Taking the limit of both sides, and using the fact that x^* is a local minimizer, we obtain

$$0 \le \lim_{\eta \to 0} \frac{f(x^* + \eta u) - f(x^*)}{\eta^2} = \lim_{\eta \to 0} \left\{ \frac{1}{2} u^T \nabla^2 f(x^*) u + \frac{o(\eta^2)}{\eta^2} \right\}.$$

Since

$$\lim_{\eta \to 0} \frac{o(\eta^2)}{\eta^2} = 0,$$

we conclude that $u^T \nabla^2 f(x^*) u \ge 0$. Therefore, $\nabla^2 f(x^*)$ is positive semidefinite. ■

Theorem 2.3.4. *(Second-Order Sufficient Condition) [11, 27] Let $f \colon \mathbb{R}^n \to \mathbb{R}$ be twice continuously differentiable. If $\nabla f(x^*) = 0$ and $\nabla^2 f(x^*)$ is(positive definite), therefore x^* is a strict local minimizer.*

Proof: Since $\nabla^2 f(x^*)$ which is positive definite and $\nabla^2 f$ to be continuous, we have that $u^T \nabla^2 f(x) u > 0$, for all $x \in \mathcal{B}(x^*, t)$ for some $t > 0$, and for any $u \neq 0$.

Now let $u \neq 0$, s.t $x^* + u \in \mathcal{B}(x^*, t)$. Since $\nabla f(x^*) = 0$, and by using the Taylor Expansion

Theorem (2.1.2), there exists a point q that lies between x^* and $x^* + u$ such that

$$\begin{aligned} f(x^* + u) &= f(x^*) + \nabla f(x^*)^T u + \frac{1}{2} u^T \nabla^2 f(q) u \\ &= f(x^*) + \frac{1}{2} u^T \nabla^2 f(q) u \\ &> f(x^*). \end{aligned} \tag{2.5}$$

Hence $f(x^* + u) > f(x^*)$, so x^* is a strict local minimum. ■

Definition 2.3.15. [62] Let $f\colon \mathbb{R}^n \to \mathbb{R}$ be differentiable. the point $x^* \in \mathbb{R}^n$ is denoted to be a stationary point of f if $\nabla f(x^*) = 0$.

2.3.3 Optimality Conditions for Constrained Optimization

In the presence of additional constraints equality (inequality) or both of them , the optimization problem becomes known as the general nonlinear programming problem. And it should be be expressed as [11]

$$\begin{cases} \text{minimize} & f(x) : \\ \text{subject to:} & g_i(x) = 0 \quad i = 1, \dots, E, \\ & x \in X \end{cases} \tag{2.6}$$

where f and g_i defined from $X \subseteq \mathbb{R}^n$ into $\mathbb{R}$ are assumed which is continuously differentiable functions. The feasible set of (2.6) is called by

$$The feasible(S) = \{x \in X \mid g(x) = 0\},$$

so g is function represented functions $g_1, \dots, g_E$. assume $X = \mathbb{R}^n$.

The Lagrangian function $\mathcal{L}\colon \mathbb{R}^n \times \mathbb{R}^E \to \mathbb{R}$ is which implies $\beta = (\beta_1, \dots, \beta_E)^T$ is called the Lagrange multiplier vector.

CHAPTER 3

ONE-DIMENSIONAL AND N-DIMENSIONAL OF NUMERICAL OPTIMIZATION

In this chapter, we begin by discussing the One-dimensional of numerical optimization(constrained problems,unconstrained problems),concepts and terminology.Furthermore,Python packages is used in the study ,multivariable optimization(constrained problems ,unconstrained problems),Euler's method ,Dynamic Optimization .

3.1 Numerical Challenges

Numerical analysis is a powerful tool because it can be used to investigate and provide accurate answers to real-world issues in many different disciplines, including but not limited to science, engineering, biology, astrophysics,and finance. Using techniques from algebra, partial differential equations, calculus, and other related branches of mathematics [61].Analysis seeks to simplify the basic equations of a problem in order to arrive at a solution.Numerical analysis uses the same operations of addition, subtraction, multiplication, division and comparison to arrive at numerical answers for the same reasons [7]. Numerical analysis relies heavily on computers perform these calculations with great accuracy.In many fields, including the life sciences. The ultimate goal of any numerical analysis is to provide a reasonably close yet accurate answer to the complex problem [18].Different uses determine the level of precision required. There are many problems that can be solved easily and find the optimal solution, as in the example.

Example 3.1.1. $\begin{cases} \text{maximize} f(x) = -x^3 + 3x^2 + 9x + 10 \\ \qquad\qquad \textit{in the interval of } [-2, 4] \end{cases}$

$$\frac{df}{dx}|_{x=x^*} = -3x^2 + 6x + 9 = 0 \Rightarrow x^* = 3, -1$$
$$\text{and} f(3) = 37, f(-1) = 5$$
$$\therefore 3 \text{ is the optimum point.}$$

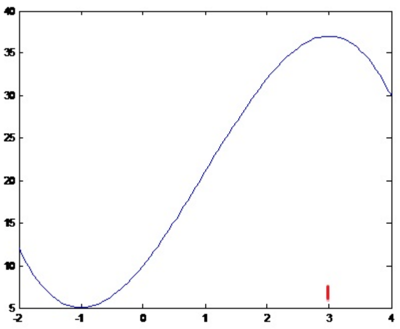


Figure 3.1: Analytical Approach

There are a variety of difficulties posed by modeling [13], the task of describing a realistic phenomenon from a mathematical perspective. Assuming that this has already been done, there are many major obstacles to solving a mathematical problem.Facing which cannot determine the optimal solution as in the example below

Example 3.1.2. Find the minimum of $f(x) = (10x^3 + 3x^2 + x + 5)^2$

$$2(10x^3 + 3x^2 + x + 5)(30x^2 + 6x + 1) = 0$$

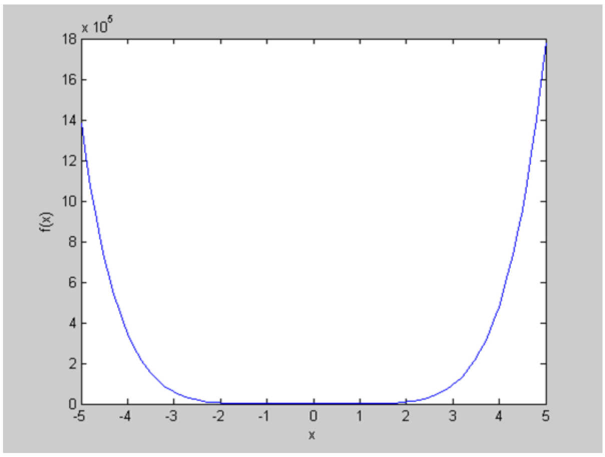


Figure 3.2: Numerical Approaches

The problem with the above example is the difficulty in finding the root in the above equation which leads it hard to find the optimal solution

3.2 Gradient Descent Method

Gradient descent initialize the gradient and learning rate at the beginning of each iteration to provide new points for each iteration. Vector of objective function that contains partial derivatives with respect to points' dimensions or coordinates. In other words, the learning rate tells us how close the optimal point and how much time it will take to get there [65]. In minimization problems, the gradient is negative, and in maximization problems, the gradient is positive. The Gradient Descent algorithm can be visualized as a hill that is gradually descending if we are briefed on its principle, goal of descending the mountain in the quickest feasible time by meticulous planning and adherence to detailed instructions [21].The starting point, progress rate, and general direction are all determined by it. It get a head start on achieving our objective it these factors as a guide. When the gradient is zero. The reached the lowest since the function's derivatives are all zero at either the local or global minimum (maximum) [55]. According to the function's nature and location, it might be either global or local. In the case of convex functions, gradient descent can be used to ensure that the function under study converges to a single minimum. In more complex functions, most algorithms ultimately stabilize (and may even be optimum). The gradient is zero points, based on the step size. In order to pick the precise step size, there are several criteria and approaches to consider, such as. A simple linear regression model will be used to demonstrate how the gradient descent works to reduce costs by optimizing the cost function using intercept and slope

$$x^{k+1} = x^{k} - \triangle\, f(x^{k})\tau$$

Where, (x^{k+1}) is the next point get x^{k} is the current location, the gradient of the function at the current location, and"τ" is the learning rate [65].

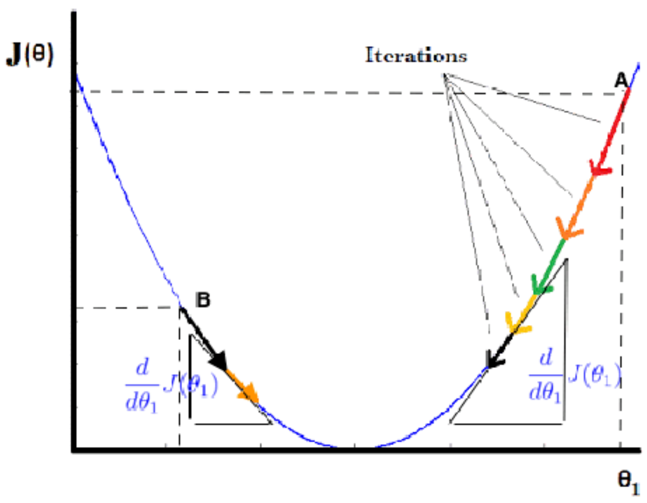
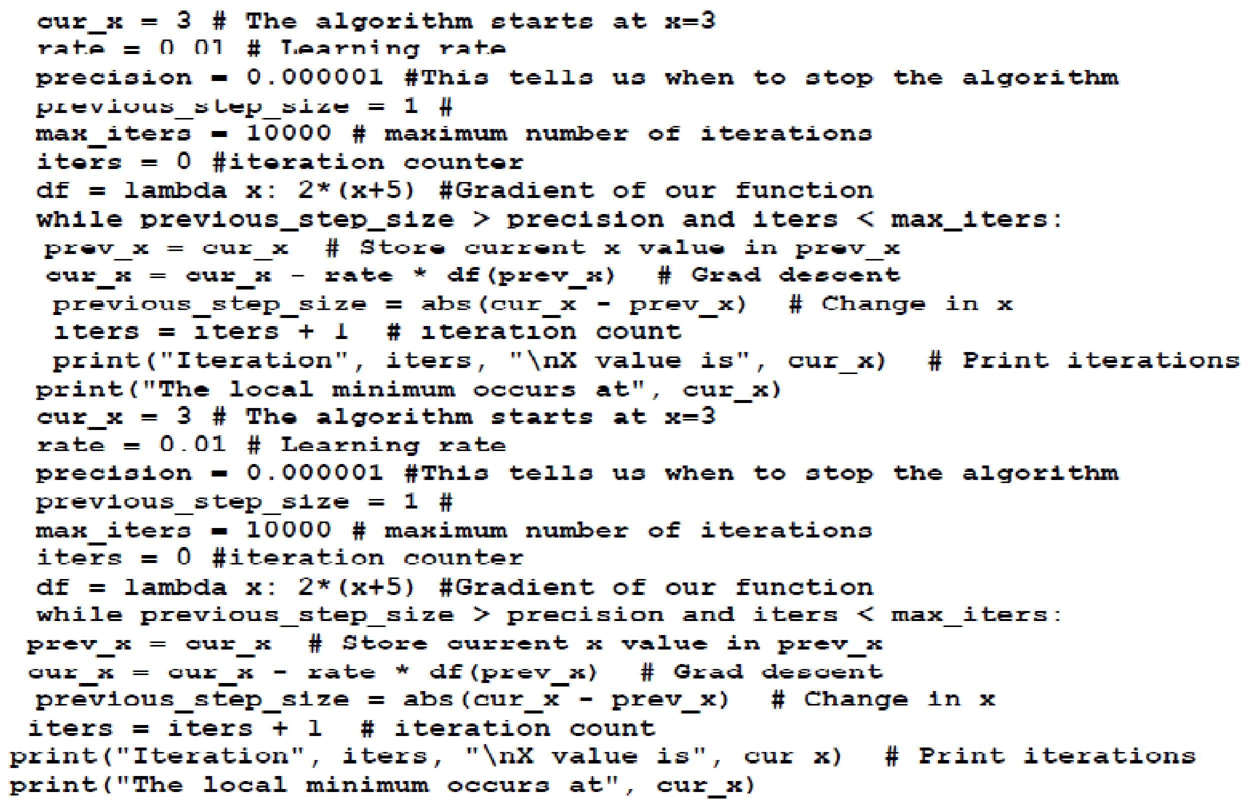

J(θ)
Iterations
A
B
$\frac{d}{d\theta_1}J(\theta_1)$
$\frac{d}{d\theta_1}J(\theta_1)$
θ_1

Figure 3.3: Gradient Descent Method

Gradient descent is used to locate the function's minimum. It starts from a random point and keeps calculating the first order derivative of the function. Then it steps in the direction of the gradient. Here is an example starting from two random points A and B [88].

Example 3.2.1. Find the local minimum of the function $y = (x + 5)^2$ starting from the point $x = 3$ and how to obtain the same numerically using gradient descent.

Code 4: Gradient Descent Method



```
cur_x = 3 # The algorithm starts at x=3
rate = 0.01 # Learning rate
precision = 0.000001 #This tells us when to stop the algorithm
previous_step_size = 1 #
max_iters = 10000 # maximum number of iterations
iters = 0 #iteration counter
df = lambda x: 2*(x+5) #Gradient of our function
while previous_step_size > precision and iters < max_iters:
 prev_x = cur_x  # Store current x value in prev_x
 cur_x = cur_x - rate * df(prev_x)  # Grad descent
  previous_step_size = abs(cur_x - prev_x)  # Change in x
  iters = iters + 1  # iteration count
  print("Iteration", iters, "\nX value is", cur_x)  # Print iterations
print("The local minimum occurs at", cur_x)
```

```
cur_x = 3 # The algorithm starts at x=3
rate = 0.01 # Learning rate
precision = 0.000001 #This tells us when to stop the algorithm
previous_step_size = 1 #
max_iters = 10000 # maximum number of iterations
iters = 0 #iteration counter
df = lambda x: 2*(x+5) #Gradient of our function
while previous_step_size > precision and iters < max_iters:
prev_x = cur_x  # Store current x value in prev_x
cur_x = cur_x - rate * df(prev_x)  # Grad descent
previous_step_size = abs(cur_x - prev_x)  # Change in x
iters = iters + 1  # iteration count
print("Iteration", iters, "\nX value is", cur x)  # Print iterations
print("The local minimum occurs at", cur_x)
```

Iteration	x
1	2.84
2	2.6832
3	2.529536
4	2.37894528
5	2.2313663744
6	2.0867390469119997
. . .	
594	-4.999950866358997
595	-4.9999518490318176
The local minimum occurs at -4.9999518490318176	

Table 3.1: **Our Results by Using the Gradient Descent Method**

3.3 One-Dimensional of Numerical Optimization

There are problems that are easiest to solve are first-order problems, while the hardest are third-order problems. In practice problems that are multidimensional are constrained It usually boils down application to unconstrained multidimensional problems which in turn means that for unconstrained one-dimensional problems [32]. The Nonlinear programming algorithms are available when minified function of one variable without restrictions.Therefore,if multidimensional optimization algorithms are effective. One-dimensional optimization algorithms are also effective.The must be built algorithms unconstrained and constrained.However, the issue with a One-Dimensional Optimization is is [51]

$$\begin{cases} \min \quad K = f(x) \\ \text{s.t} \quad x_l \leq x \leq x_u \end{cases}$$

where the function $f(x)$ with one variable. In some range if $f(x)$ is continuous of x there is solution to this problem , i.e., the x_l *and* x_u of the minimizer are the lower and upper bound and in some range $f(x)$ has only one minimum $x_l \leq x \leq x_u$. There are approximate methods and research methods used in one-dimensional optimization Methods. An intervals $[x_l, x_u]$ In search methods,containing a, Known as arc, is after they are created and based on job evaluations then they are downgraded again and again. So obtaining a lowered bracket $[x_l, k, x_u, k]$ which is small enough It can be assumed that the minifier is in the center of the time period $[\ x_l,\ k,\ x_u,\ k]$. To which function these operations are applied Repeating this process several times, the interval $[xl, xu]$ is narrowed until a precise enough value of is reached. In these procedures, it is necessary that $f(x)$ be differentiable and continuous. (i.e.,$f(x) \in C^1$) [76]

The one-dimensional optimization methods are .

1 - Golden-section search method

2 - Fibonacci search method

3 - Dichotomous search method

4 - Newton Method

5 - Quadratic Interpolation Method

6 - Cubic Interpolation Method

► These are research methods (first three) and approximation methods (last three).

3.3.1 Unconstraint Optimization

Golden-Section Search Method

[42]

▶ Suppose f is function on $[x_l, x_u]$ and let x_1 and x_2 be two points in $[x_l, x_u]$, with $x_1 \leq x_2$

▶ Choose two midpoints x_1 *and* x_2 such that

$x_2 = x_u - d, x_1 = x_l + d, \quad where \quad d = (x_u - x_l)$, the golden ratio $= \dfrac{\sqrt{5}-1}{2}$

▶ Evaluate $f(x_1)$ and$f(x_2)$

if $f(x_1) > f(x_2)$ then determine new x_l , x_1 , x_2 note the only new compute is done to determine the new x_1 ,

$x_2 = x_1 \quad x_u = x_u \quad x_l = x_2 \quad x_1 = x_l + \dfrac{\sqrt{5}-1}{2}(x_u - x_l)$

If $f(x_1) < f(x_2)$ then determine new x_l, x_1, x_2 note the only new

compute is done to determine the new x_2, $x_l = x_l \quad x_u = x_1 \quad x_1 = x_2 \quad x_2 = x_u + \dfrac{\sqrt{5}-1}{2}(x_u - x_l)$

▶ if $x_u - x_l < \varepsilon$ (a sufficiently small number) then the maximum occurs at $\dfrac{x_u - x_l}{2}$

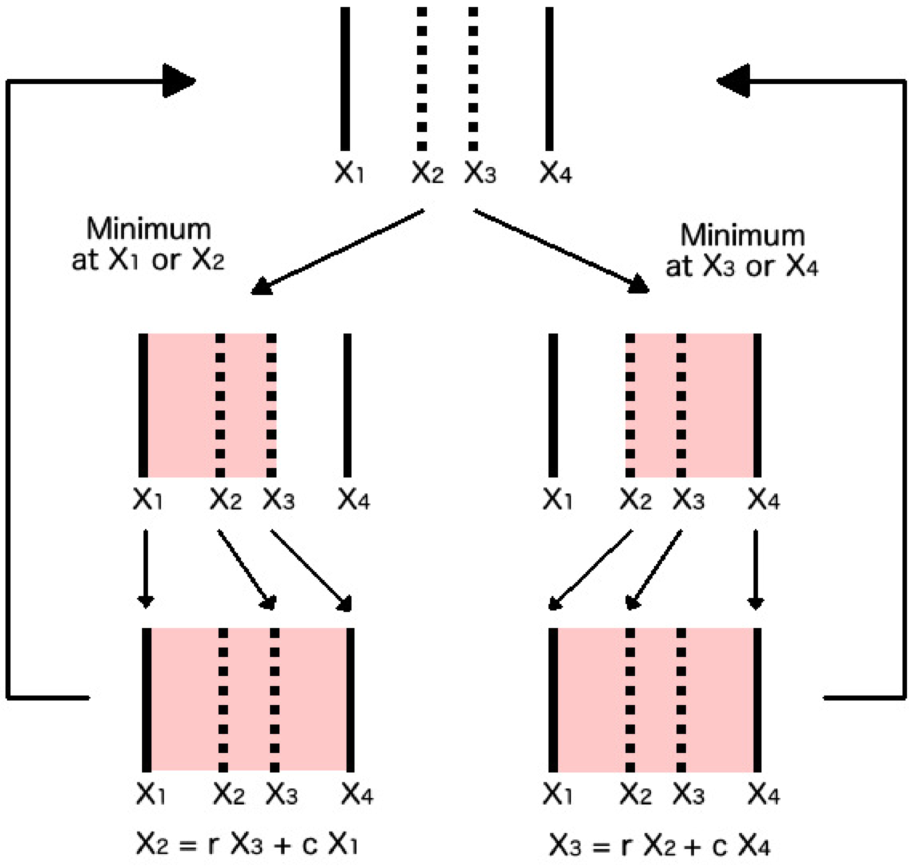


Figure 3.4: Diagram of Golden Section Search

A diagram of a golden section search showing the technique of narrowing the interval in x containing a minimum of a function $f(x)$, using the maximally efficient golden ratio.

Example 3.3.1. Solving the problem using the Golden Section Search to minimize

$$f(x) = 0.5 - x\exp(-x^2) \quad on \quad [0, 2]$$

x_1	f_1	x_2	f_2
0. 764	0. 074	1. 236	0. 074
0. 472	0. 122	0. 764	0. 074
0. 764	0. 074	0. 944	0. 113
0. 652	0. 074	0. 764	0. 074
0. 584	0. 085	0. 652	0. 074
0. 652	0. 074	0. 695	0. 071
0. 695	0. 071	0. 721	0. 071
0. 679	0. 072	0. 695	0. 071
0. 695	0. 071	0. 705	0. 071
0. 705	0. 071	0. 711	0. 071

Table 3.2: **The Results of Applying the Golden Section Search Method**

In the example to find value of x that minimizes the objective function in the range$[0, 2]$ for the given number of iterations ($N = 10$), we employed the golden search approach. After finding ($x_1 = 0.764, x_2 = 1.236, f_1 = 0.074, f_2 = 0.074$) in iteration one, we find ($x_1 = 0.705, x_2 = 0.711, f_1 = 0.071, f_2 = 0.071$) in last iteration values have become decreasing.

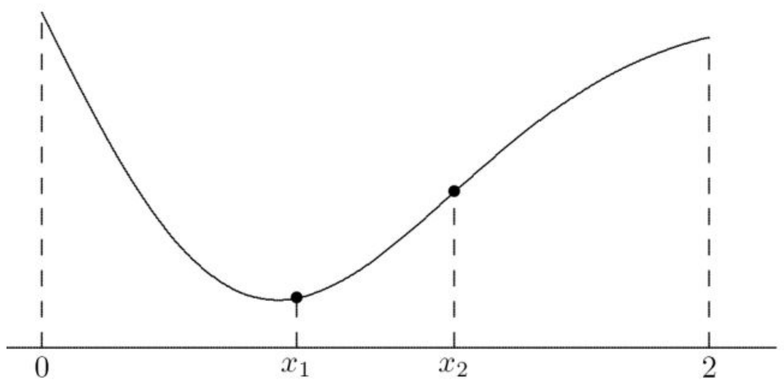


Figure 3.5: Numerical Result Base on Golden Search Method

Newton Method

To discover the roots of differentiation it use Newton's iterative method funection F, the solutions for the equation $F(x) = 0$. To determine the roots of the derivative it use Newton's technique (solutions $f'(x) = 0$),which are often referred to as the product's "critical points" [50]. When trying to find the most efficient way to do something, minimizing any unnecessary objective should be the top priority. Consider the results with just one true variable. It going to dive into a broader discussion of and a practical multivariate scenario that will be discussed later on. To this end, we have turned our attention to an problem of optimization $\min_{x \in R} f(x)$, $F : R \longrightarrow R$ is a twice-differentiable function. Address this problem through newton's technique using the following formula:

$$x_{k+1} = x_k - \frac{f'(x_k)}{f''(x_k)} \tag{3.1}$$

Solve the nonlinear equation $f'(x) = 0$ by Newton's method. Even though Newton's method for finding the minimum typically converges quadratically quickly, it requires a starting point relatively close to the solution in order to actually converge.

Example 3.3.2. Solving the problem by use Newton's method to minimize

$$f(x) = \frac{1}{4}x^3 + \frac{3}{4}x^2 - \frac{3}{2}x - 2$$

Code 5:Solving Newton Method by Python

```
from scipy import misc
def NewtonsMethod(f, x, tolerance=0.000001):
    while True:
        x1 = x - f(x) / misc.derivative(f, x)
        t = abs(x1 - x)
        if t < tolerance:
            break
        x = x1
    return x
def f(x):
    return (1.0/4.0)*x**3+(3.0/4.0)*x**2-(3.0/2.0)*x-2
x = 4
x0 = NewtonsMethod(f, x)
x1=NewtonsMethod(f, x)
x2= NewtonsMethod(f, x)
x3=  NewtonsMethod(f, x)
x4=  NewtonsMethod(f, x)
print('x: ', x)
print('x0: ', x0)
print("f(x0) = ", ((1.0/4.0)*x0**3+(3.0/4.0)*x0**2-(3.0/2.0)*x0-2 ))
print('x1: ', x1)
print("f(x1) = ", ((1.0/4.0)*x0**3+(3.0/4.0)*x0**2-(3.0/2.0)*x0-2 ))
print('x2: ', x2)
print("f(x2) = ", ((1.0/4.0)*x0**3+(3.0/4.0)*x0**2-(3.0/2.0)*x0-2 ))
print('x3: ', x3)
print("f(x3) = ", ((1.0/4.0)*x0**3+(3.0/4.0)*x0**2-(3.0/2.0)*x0-2 ))
print('x4: ', x4)
print("f(x4) = ", ((1.0/4.0)*x0**3+(3.0/4.0)*x0**2-(3.0/2.0)*x0-2 ))
#!/usr/bin/env python
from pylab import *
t = arange(-6.0, 4.0, 0.01)
s= t*t*t/4.0+3.0*t*t/4.0-3*t/2.0-2.0
ax = subplot(111)
ax.plot(t, s)
ax.scatter([-4,-1,2],[0,0,0])
ax.grid(True)
ax.spines['left'].set_position('zero')
ax.spines['right'].set_color('none')
ax.spines['bottom'].set_position('zero')
ax.spines['top'].set_color('none')
ax.set_xlim(-6,6)
ax.set_ylim(-20,20)
text(-3.0, 12,
     r"$f(x)=(1/4)*X^3+(3/4)*X^2-(3/2)*X-2$", horizontalalignment='center',
     fontsize=8)
plt.title("How to implement the Newton's method using python \n for finding the
zeroes of a real-valued function",fontsize=10)
plt.xticks(fontsize=8)
plt.yticks(fontsize=8)
plt.savefig('NewtonMethodExemple.png')
show()
```

It determined the maximum and minimum values of the functions and the roots using Newton's method. When the original function has roots of the value of the function equal

Iteration	x_n	$f(x_n)$
1	2.0009378	1.222079011e-3
2	2.0057142	1.257874138e-5
3	2.0864712	1.277242667e-5
4	2.0855236	1.279365157e-6
5	2.0855197	1.293261473e-6
The required root is 2.0855197		

Table 3.3: **The Results of Solving the Example are the Application of Newton's Method**

to zero, it use Newton's method up to the derivative function to find its origins, starting with the first approximation. Then, by combining the first and second derivatives, it create a series of approximations x_1, x_2, x_3, x_4, x_5 using the tangent lines of the f graph; To get four iterations. The value becomes an as the number of iterations increases that remains constant (2.0855197).

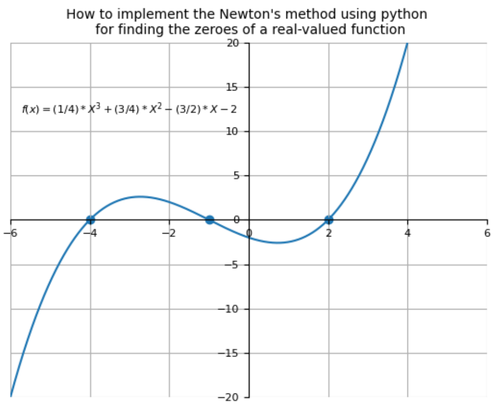


Figure 3.6: The Result Base on Newton Method

In the diagram,of the x-axis is computed the root value as $(4, -1, 2)$, and it choose the root is (2) .

3.3.2 Constraint Optimization

The general form optimization problems constraint with equality and inequality

$$\begin{cases} \text{minimize} & f(x) \\ \text{subject to} & g(x) = 0 \\ & h(x) \leq 0 \\ & x > 0 \end{cases} \tag{3.2}$$

The methods for constrained minimization problem are (Substitution method , Lagrange multiplier,Penalty Methods, Barrier Methods) [48].

Example 3.3.3. Solving the problem of quadratic programming

$$\begin{cases} \text{minimize} & f(x) = 0.5x_1^2 + 2.5x_2^2 \\ \text{subject to} & g(x) = x_1 - x_2 - 1 = 0 \end{cases}$$

The Lagrangian function is given by

$\Gamma(x, y) = f(x) + \lambda g(x) = 0.5x_1^2 + 2.5x_2^2 + \lambda(x_1 - x_2 - 1)$

Since

$$\nabla f(x) = \begin{bmatrix} x_1 \\ 5x_2 \end{bmatrix} \text{ and } J_g(x) = \begin{bmatrix} 1 & -1 \end{bmatrix}$$

we have

$$\bigtriangledown_x \Gamma(x, \lambda) = \bigtriangledown f(x) + J_g^t \lambda = \begin{bmatrix} x_1 \\ 5x_2 \end{bmatrix} + \lambda \begin{bmatrix} 1 \\ -1 \end{bmatrix}$$

So system to be solved for critical point of Lagrangian is

$x_1 + \lambda = 0$

$5x_2 - \lambda = 0$

$x_1 - x_2 = 1$

which in this case is linear system

$$\begin{bmatrix} 1 & 0 & 1 \\ 0 & 5 & -1 \\ 1 & -1 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \lambda \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

Solving this system, we obtain solution

$$x_1 = 0.833 \quad , \quad x_2 = -0.167, \quad \lambda = -0.833$$

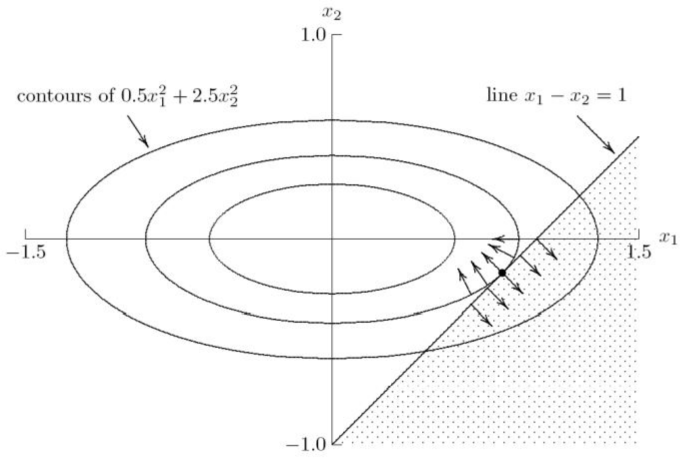


Figure 3.7: The Result Base on Lagrange Method

3.4 Numpy Package in Python

Python's NumPy library provides the foundation for a wide variety of scientific computing tasks. Quick operations on matrices, including mathematical and logical manipulation, shape, sort, selection, and discrete fourier transformations, basic linear algebra, basic statistical operations, stochastic simulation, and more, may be performed with the help of this python module [16]. An ndarray object is the backbone of the NumPy library. Many of the operations are carried out in compiled code for optimal speed, and thus encompasses homogenous n-dimensional arrays Data types. In contrast to regular Python sequences [41], NumPy arrays include a number of significant additional features. Unlike Python lists, newly formed NumPy arrays have a hard limit on their size (which can grow dynamically). The original array is overwritten when the size of an ndarray is modified. A NumPy array's elements must all have the same data type, which ensures that they all consume the same amount of memory [53]. To make up for this, Python (and NumPy) supports arrays of objects, which can include elements of varying sizes. NumPy arrays allow for sophisticated arithmetic and other operations to be performed on massive datasets. In most cases, custom Python sequences can do the same tasks with less code and more efficiency. NumPy arrays are used in many Python-based scientific and mathematical programs. To get the most out of modern Python-based scientific and mathematical software, familiarity with Python's in-built serialization module is essential. Understanding types alone is insufficient; familiarity with NumPy arrays is also required. In scientific computing, the size and speed of sequences are of importance [41]. In this in action by imagining a scenario in which a one-dimensional sequence is multiplied by the equivalent element of another sequence of the same length. If we divide the information into two Python lists, a and b, it easily go over each piece of data one by one using the for loop: Object-oriented programming is completely supported in NumPy, beginning with the ndarray data structure. As an illustration, ndarray is a file Class with a wide variety of features and characteristics [45]. Numerous its operations are mirrored by functions in the deepest NumPy namespace, giving the writer complete freedom to write in any style they see fit. Benefits of this adaptability include Through the use of the NumPy array dialect and the NumPy ndarray class, Python was able to become a true multidimensional language. Python and NumPy share data with one another [31]. This implicit behavior holds for all kinds of operations, not just arithmetic but also logical, bitwise, functional [46].

3.5 GEKKO Package in Python

Applied mathematics and optimization software written in Python, GEKKO is particularly adept at the dynamic optimization of systems of differential algebraic equations (DAE). Linear, quadratic, nonlinear, and mixed-integer programming solvers APOPT and IPOPT are included [10]. Applied mathematics, state space models (both continuous and discrete), simulation, estimation, and control are all possible. GEKKO Python can access solvers for continuous, discrete, and unconstrained problems, making it suitable for large-scale optimization [3]. Nonlinear optimization, systems of dynamic nonlinear equations, and multi-objective optimization are all areas where problems may be solved. The system can integrate optimization strategies into third-party modeling and analytic tools, as well as locate optimal solutions, conduct tradeoff assessments, and weigh different design choices [58]. Gekko models comprise of equations and variables to produce a symbolic representation of the situation. Over all data points and time horizons, the entire model is then generated by the solution modes. Gekko can handle many different kinds of issues, such as [25]:

- Methods of Linear Programming (LP)
- Calculating using Quadratics (QP)
- Dynamic Programming
- Programming with Mixed-Integers (MILP)

- Quadratic Programming using Mixed Integers (MIQP)
- The Nonlinear Programming with Mixed-Integers (MINLP)
- The Theory of Differential Equations (DAEs)
- The Complementarity-Constrained Mathematical Programming Problem (MPCCs)

In order to offer the solver with automated differentiation, Gekko translates the model into byte code and supplies it with sparse derivatives. Whether using Windows, Linux, MacOS, ARM processors, or another platform that supports Python, Gekko has the data purification tools and common tag actions for industrial-strength management and optimization.

3.6 Multivariable Optimization

A multivariate optimization problem is one in which there are a great number of independent variables to consider. [63]

$$Y = f(x_1, x_2, x_3, ..., x_n)$$

In general, the function Y can be a nonlinear function of the decision variables $x_1, x_2, x_3, ..., x_n$. Therefore, there are n variables in which one can optimization this Y function. Note that one can explain univariate optimization using two-dimensional images. The value of the decision variable,because in the x direction and in the y direction, the function's value.The use images in three dimensions if the optimization is multivariate,It is difficult to visualize if the decision variables are more than two. Multivariate optimization can be categorized into two parts Depending on the constraints [75].

1 - Optimization multivariable with equality constraint

2- Optimization multivariable with inequality constraint

3.6.1 Constraint Optimization

In this section we will discuss the basic concept of multivariable optimization problems

Optimization Multivariable With Inequality Constraints

The general form:

$$\begin{cases} \text{minimize (maximize)} & Z = f(x_i) = f(x_1, ..., x_n) \\ \text{subject to} & h_j(x_i) \leq 0, \quad (j = 1, 2, ...m) \\ & x_i > 0 \end{cases} \tag{3.3}$$

The Approximate Methods of solution:

(1)The Lagrange Multiples Method

(2)KKT conditions Method

(3)The Direct Method

(4)The Euler Method

Example 3.6.1. Solving optimization problem

$$\begin{cases} \text{minimize (maximize)} & Y = f(x_1, x_2) = 2x_1^2 - 3x_2^2 - 2x_1 \\ \text{subject to} & g_1(x_1) = x_1^2 + x_2^2 \leq 1 \\ & x_i > 0 \end{cases} \tag{3.4}$$

Use method [(KKT condition) +(Lagrange Multiples)].

Solution:

1) Slack variable

$M_j^2 = g_j(x_i) \geq 0$

$M_1^2 = 1 - x_1^2 - x_2^2 \geq 0$

2) (Kuhn-Tucker) +(Lagrange Multiples)

$L(x_1, x_2, \lambda, M) = 2x_1^2 - 3x_2^2 - 2x_1 + \lambda(M^2 - 1 + x_1^2 + x_2^2)$

3) Necessary condition

$\dfrac{\partial L}{\partial x_1} = 4x_1 - 2 + 2\lambda_1 x_1$

$\dfrac{\partial L}{\partial x_2} = -6x_2 + 2\lambda_1 x_2$

$\dfrac{\partial L}{\partial \lambda_1} = M^2 - 1 + x_1^2 + x_2^2 = 0$

$\dfrac{\partial L}{\partial M} = 2M\lambda_1$

4) First case : when$M_j^* = 0$

$4x_1 - 2 + 2\lambda x_1 = 0 \Rightarrow 2x_1 + \lambda x_1 = 1 \qquad (1)$

$-6x_2 + 2\lambda x_2 = 0 \Rightarrow -3x_2 + \lambda x_2 = 0 \qquad (2)$

$-1 + x_1^2 + x_2^2 = 0 \Rightarrow x_1^2 + x_2^2 = 1 \qquad (3)$

form (2) we get

$-3x_2 + \lambda x_2 = 0 \Rightarrow x_2(-3 + \lambda) = 0 \Rightarrow (x_2 = 0, \lambda = 3)$

i)$When\ \lambda\ = 3 : substitute\ \ in(1), weget$

$2x_1 + \lambda x_1 = 1 \Rightarrow 2x_1 + 3x_1 = 1 \Rightarrow 5x_1 = 1 \Rightarrow x_1 = 0.2$

substitute in (3), we get

$x_1^2 + x_2^2 = 1 \Rightarrow (0.2)^2 + x_2^2 = 1 \Rightarrow x_2^2 = 1 - 0.04 \Rightarrow x_2^2 = 0.96 \Rightarrow x_2 = \mp\sqrt{0.96}$

So, we have

$(x_1^*) = 0.2, \qquad x_2^* = \mp\sqrt{0.96}$

In the objective function when substituting these values we get

$z = f(x_1, x_2) = 2x_1^2 - 3x_2^2 - 2x_1 \Rightarrow f(0.2\ , \mp\sqrt{0.96}) = 0.08 - 3(\mp\sqrt{0.96})^2 - 0.4$

$= f(0.2\ , \mp\sqrt{0.96}) = 0.08 - 3(0.96) - 0.4 = 0.08 - 2.88 - 0.4 = -3.2$

so for $f(0.2\ ,\mp\sqrt{0.96}) \Rightarrow z = -3.2$

ii)$When\ \ x_2 = 0$

from (3) we get

$x_1^2 + x_2^2 = 1 \Rightarrow x_1^2 + (0) = 1 \Rightarrow x_1^2 = 1 \Rightarrow x_1 = \pm 1$

So, we have

$(x_1^* = 1\ ,\ \ x_2^* = 0)\ or(x_1^* = -1\ \ ,\ \ x_2^* = 0)$

and

$z = f(x_1, x_2) = 2x_1^2 - 3x_2^2 - 2x_1$

$f(1\ ,0) = 0$

$f(-1\ \ ,0) = 4$

$Second\ \ case : when\ \ \lambda_i^* = 0\ \ then$

$2x_1 - 1 = 0$ (1)

$-3x_2 = 0$ (2)

$M^2 - 1 + x_1^2 + x_2^2 = 0$ (3)

From (1) and (2) we get

$2x_1 - 1 = 0 \Rightarrow x_1 = 0.5$

$-3x_2 = 0 \Rightarrow x_2 = 0$

Substitute in (3) we ge

$M^2 = 1 - x_1^2 - x_2^2 \Rightarrow M^2 = 1 - (0.5)^2 - (0)^2 \Rightarrow M^2 = 0.75 \Rightarrow (+)$

$M\ is\ positive$

$z = f(x_1, x_2) = 2x_1^2 - 3x_2^2 - 2x_1$

$f(0.5\ ,\ 0) = 0.5$

5) Optimal solution

$z = f(0.2\ ,\pm\sqrt{0.96}) = -3.2 \quad \rightarrow optimal\ \ solution\ \ (min)$

$z = f(-1\ ,0) = 4 \quad \rightarrow optimal\ \ solution\ \ (max)$

Multivariable optimization with equality constraints

In mathematics, an assertion of equality is a statement that two quantities are equal or that two mathematical expressions represent the same mathematical object. We say "so" when we have an objective function with multiple decision variables and an equality constraint. [8].

The general problem is:

$$\begin{cases} \text{Minimize(Maximize)} \quad f(x) \\ \text{subject to} \qquad g_j(x) = 0, \qquad\qquad j = 1, 2, ..., m \\ \quad where \quad x = [x_1, x_2, ..., x_n]^T \end{cases} \tag{3.5}$$

Equality constraints, denoted by g, are a type of function. Only solutions that are able to work within these constraints can be considered acceptable. In mathematics, imposing constraints of equality leads to a reduction in the number of dimensions.
For a two-variable function $f(x)$ with an equality constraint $g_1(x)$, the solution space is reduced to two dimensions; for a two-variable function with two equality constraints $g_2(x)$, where both conditions are met when there is only one possible solution [49].

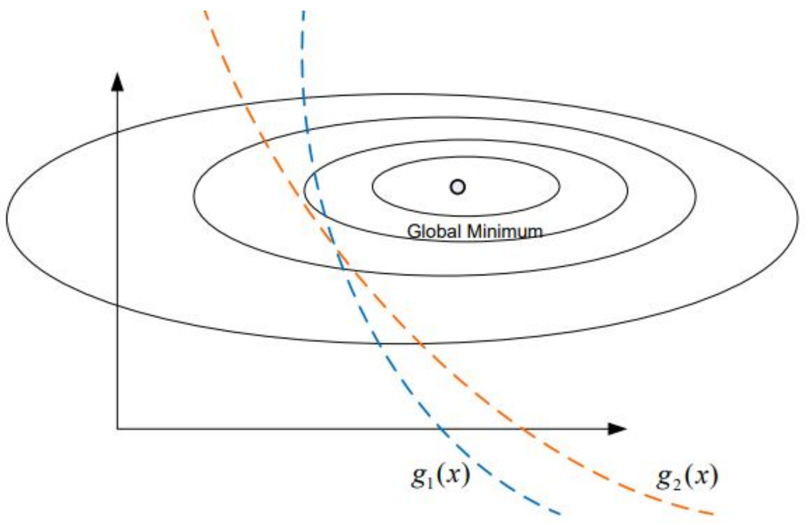


Figure 3.8: Multivariable Optimization with Equality Constraint

► By Lagrange Multiplier multivariate optimization with equality constraints will be solved.

Using Lagrange Multipliers to solve for equality constraints is a very ingenious strategy. The method is based on the equality constraints of the objective function. Conversely, we add an extra variable to the mix. When there are exactly as many equality constraints as lambda variables. Therefore, the current issue is [77].

$$\text{minimize } L(x, \lambda) = f(x) + \sum_{i=1}^{m} \lambda_j g_j(x))$$

From this point on, the techniques we developed will be used when solving multivariate optimization Unlimited problems from this point on. This means that we are looking for conditions in which the first derivative of L disappears.

Example 3.6.2. Solving optimization Problems

$$\begin{cases} \text{Minimize} \quad f(x_1, x_2) = x_1^2 + x_2^2 \\ \text{Subject to} \quad x_1 + x_2 - 5 = 0 \end{cases}$$

the transformed problem into ,

minimize $L(x_1, x_2, \lambda_1) = (x_1^2 + x_2^2) + \lambda(x_1 + x_2 - 5)$

This new problem has 3 *variables* (x_1, x_2, λ) Differentiating

$$\nabla L = \begin{Bmatrix} 2x_1 + \lambda_1 \\ 2x_2 + \lambda_1 \\ x_1 + x_2 - 5 \end{Bmatrix} = \begin{Bmatrix} 0 \\ 0 \\ 0 \end{Bmatrix}$$

The equations are in general nonlinear. However, the third equation is the constraint equality in the derivatives as $\dfrac{\partial L}{\partial \lambda_1}$

$$x_1 = -\frac{\lambda_1}{2} x_2 = -\frac{\lambda_1}{2}$$

when the substitute into the third equation we will find$\lambda_1 = -5$

$$x_1 = x_2 = 2.5$$

3.7 Control Optimization

Mathematical optimization is the process of selecting the best member of the group based on some criteria. Today, optimization is one of the most popular techniques used by modern control theory practitioners. In order to obtain the closed-loop characteristic, one must define the control law, adjust the controller parameters, fit the model, and determine the optimal operating conditions. Reducing or increasing the value of a job by systematically limiting its inputs is the simplest form of optimization. Valid input matrix elements are used to specify the output of the function. Algorithms with a finite number of steps, when dealing with optimization issues, both iterative and approximate methods can be used within a particular class of problems that are close to a solution. In some cases (although their frequency does not necessarily converge). work actively When posing the challenge of designing a control system, it is important to consider the wide variety of situations, physical limitations, and product quality constraints that must be met. current difficulties Implementation of optimization algorithm, especially for problem solving [17].

3.8 Euler's method

Euler's method is a numerical approach for solving ordinary differential equations (ODEs) with a known initial value. The method is named after the mathematician Leonhard Euler. The approximation of solutions to certain differential equations can be achieved by using Euler's approach. It does this by coming up with an approximate solution curve using linear segments. Using different numerical techniques equations in differential geometry that are difficult to solve can always be approximated [33]. The Euler method is going to be discussed here as an example of a numerical method. Euler method It begins at the point (x_0, y_0) and uses publicly available slope information to advance of a given differential equationalong the polygon approximation of the graph moving from one point to another until it reaches the final point. This process continues until it reaches the final point (x_n, y_n). The value of (y_n) at the endpoint of the differential equation is frequently of the utmost importance, notwithstanding his interest in calculating along the differential equation at all points. Differential equations are common in physics, chemistry, and economics. However, differential equations typically cannot be solved explicitly, thus it is necessary to make an estimate of the solutions. This is where Euler's approach comes in handy. For instance, one can use the Euler method to approximate the path that an object will take as it travels through a viscous fluid as it falls, the rate at which a reaction will occur over time, and the flow of traffic on a busy road.

the general of solution the differential equation be $y = f(x)$

$$\frac{dy}{dx} = g(x, y), \quad y(x_o) = y_0$$

for $x_o \leq x \leq x_n$ and let $x_{i+1} = x_i + h$, where $h = \frac{x_n - x_o}{n}$ and

$$\begin{cases} y_{i+1} = y_i + g(x_i, y_i)h \end{cases} \tag{3.6}$$

for $0 \leq i \leq n-1$, then $f(x_{i+1}) \approx y_{i+1}$

Example 3.8.1. Use Euler method solving

$$\frac{dy}{dx} = x + y + 6$$
$$\text{with initial value } y = 1, x = 0, y(0) = 1$$

Code 6:Euler Method

```
 # Euler method python program
# function to be solved
def f(x, y):
    return x + y +6
# or
# f = lambda x: x+y +6
# Euler method
def euler(x0, y0, xn, n):
    # Calculating step size
    h = (xn - x0) / n
    print('\n-----------SOLUTION-----------')
    print('------------------------------')
    print('x0\ty0\tslope\tyn')
    print('------------------------------')
    for i in range(n):
        slope = f(x0, y0)
        yn = y0 + h * slope
        print('%.4f\t%.4f\t%0.4f\t%.4f' % (x0, y0, slope, yn))
        print('------------------------------')
        y0 = yn
        x0 = x0 + h
    print('\nAt x=%.4f, y=%.4f' % (xn, yn))
# Inputs
print('Enter initial conditions:')
x0 = float(input('x0 = '))
y0 = float(input('y0 = '))
print('Enter calculation point: ')
xn = float(input('xn = '))
print('Enter number of steps:')
step = int(input('Number of steps = '))
# Euler method call
euler(x0, y0, xn, step)
```

x_0	y_0	slope	y_n
0.0000	1.0000	7.0000	2.4000
0.2000	2.4000	8.6000	4.1200
0.4000	4.1200	10.5200	6.2240
0.6000	6.2240	12.8240	8.7888
0.8000	8.7888	15.5888	15.5888
x= 1.0000		y = 11.9066	

Table 3.4: **Numerical Result Euler Method**

In the above example it given an initial value $y = 1, x = 0, y(0) = 1$,and By using the formula $y_{i+1} = y_i + g(x_i, y_i)h$, and $x_{i+1} = x_i + h$, where $h = \dfrac{x_n - x_o}{n}$ It was valuable $x_0 = 0.0000, y_0 = 1.0000, slope = 7.0000$ and $y_n = 2.4000$ to the fifth iteration $x_0 = 0.8000, y_0 = 8.7888, slope = 15.5888$ *and* $y_n = 11.9066$ and values were found $x = 1.0000, y = 11.9066$

3.9 Dynamic Optimization with Linear and Non Linear Models

The Optimization problems for both linear and nonlinear dynamic programming (DP) are a common mathematical approach. The word "dynamic" is coined because the method is typically used to create a series of optimal judgments that may adapt dynamically to changes in circumstances over time [57]. Different from linear programming, dynamic programming does not assume a linear relationship between variables. This means that a variety of issues can be addressed with this technology. It's great that there is some difference. However, this comes at a cost because it requires a very special problem framing the optimization problem as a dynamic program.

Therefore, dynamic programming structures are usually considered works of art. As a matter of control, modeling the system is a challenge. The goal is to provide a mathematical description so simple that it accurately predicts the reaction. [57].

$$Y = f_{ode}(x, t, u, d)$$

Here,$t \in R$ stands for the independent variable, usually called time, $u(t) \in U$ Represents control variables (input or processing) at time t. The vector $Y \in R^n$ at any time instant t the $n \geq 1$, represents the state (or phase) variables, which characterize the behavior of the system. for initial condition $Y(t_0) = Y_0$, a solution $x(t; x_0, u(0))$ of the above equation is called a response of the system, corresponding to the control $u(do)$. The optimal control problem can be made subject to many different types of limitations. Consequently, the control and case variables can only take on certain values within this framework.. Optimal control problems can have restrictions that are analogous to point constraints and path constraints. It's possible that all of these constraints are unequal. a limiting of points. Regularly employed in optimum control problems, in particular as terminal constraints. Optimal control implies the optimization of a dynamical system [78]. Typically, this takes the form of a trajectory in which the states of the system evolve with time. The evolution of the states is typically governed by an ordinary differential equation (ODE) or a differential algebraic equation (DAE). In Dymos, the characterize all dynamics as an ODE, although solving systems of DAEs is also possible. In order to impact the behavior of such systems, the need controls. Controls may be allowed to vary with time. And Dymos is a library for the optimal control of multidisciplinary dynamic systems. While it can improve typical control optimization problems, its main advantage is the ability to optimize systems where the path is just one part of the overall optimization. Other optimization programs frequently rely on parameterizing hardware models to approximate, e.g., engine mass as a function of thrust level. such as the angle of attack of an aircraft during flight [26]. It called as "dynamic controls." [28].

The general form :

$$\begin{cases} Minimize\ J = f_{obj}(x,t,u,d) \\ s.t\ Dynamic\ Constraints\ z = f_{ode}(x,t,u,d) \\ \qquad Time\ t_{lb} \leq t \leq t_{ub} \\ \qquad State\ variables\ x_{lb} \leq x \leq x_{ub} \\ \qquad Dynamic\ Controls\ u_{lb} \leq u \leq u_{ub} \\ \qquad Design\ Parameters\ d_{lb} \leq d \leq d_{ub} \\ \qquad Initial\ Boundary\ Constraints\ g_{0,lb}^{-} \leq g_0(x_0,t_0,u_0,d_0) \leq g_{0,ub}^{-} \\ \qquad Final\ Boundary\ Constraints\ g_{f,lb}^{-} \leq g_f(x_f,t_f,u_f,d_f) \leq g_{f,ub}^{-} \\ \qquad Path\ Constraints\ p_{f,lb}^{-} \leq p_f(x,t,u,d) \leq p_{f,ub}^{-} \end{cases}$$

Example 3.9.1. Solving optimization problem

$$\begin{cases} \min\limits_{u} & 5 + x_1(t_f) \\ \text{s.t} & \dfrac{dx_1}{dt} = u \\ & 1 \leq u(t) \leq 2 \\ & t_f = \{0, 0.1, 0.2, 0.3, ..., 0.9, 1\} \end{cases}$$

Code 7:Solving Dynamic Optimization by Python

```
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
x1 = m.Var(value=1)
u = m.Var(value=0,lb=1,ub=2)
p = np.zeros(nt)
p[-1] = 1.0
final = m.Param(value=p)
# Equations
m.Equation(x1.dt()==u)
# Objective Function
m.Obj(5+x1*final)
m.options.IMODE = 6
m.solve()
# m.solve(remote=False)  # for local solve
plt.figure(1)
plt.plot(m.time,x1.value,'k-',lw=2,label=r'$x_1$')
plt.plot(m.time,u.value,'r--',lw=2,label=r'$u$')
plt.legend(loc='best')
plt.xlabel('Time')
plt.ylabel('Value')
plt.show()
```

	scaled	unscaled
Objective	5.0300004508232968e+02	5.0300004508232968e+02
Dual infeasibility	3.8477740524497060e-09	3.8477740524497060e-09
Constraint violation	4.4408920985006262e-16	4.4408920985006262e-16
Complementarity	4.5102342809243662e-07	4.5102342809243662e-07

Table 3.5: **Numerical Result Dynamic Optimization**

The final value of the objective function	503.000045082330
Solution time	2.199999999720603E-002 sec
Objective	503.000045082330

Table 3.6: **Numerical Result Dynamic Optimization**

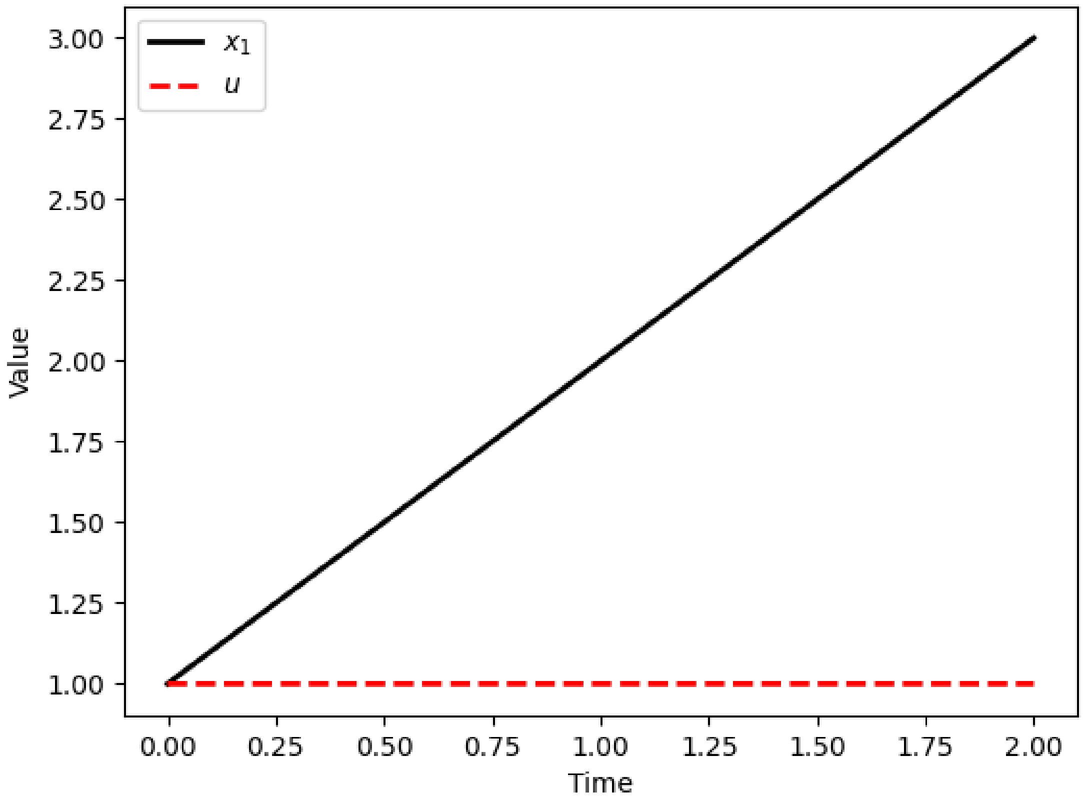


Figure 3.9: The Dynamic Optimization

The implementation we obtained based on two variables (x_1) depends on $u(t)$ changes, which represented the control variables at the time t. As can be seen in the figure, the x-axis represents the time and the y-axis represents the value of the objective function relationship.

CHAPTER 4

SOLVING MATHEMATICAL MODELS USING NUMERICAL OPTIMIZATION METHODS

4.1 Introduction

In this chapter, we will discussing numerical optimization (constrained problems, unconstrained problems) and the concepts and terms we used. Moreover, Python packages, solve mathematical models using numerical optimization methods, and mathematical models are One-dimensional of numerical optimization , multivariate optimization problems (large scale) and These models contain Dynamic, Control optimization and its applications.

4.2 Numerical Optimization

The definition numerical optimization is the process of maximizing or reducing an evaluation scale of a ”objective function” according to a number of constraints. Among the most important methods in machine learning, it plays a crucial role. Setting up a loss function to gauge the quality of the answer and then reducing the parameters of that function to discover the solution is a common approach to solving situations where finding the best solution directly is challenging [43]. Mathematical and computational studies include solving issues where several alternatives must be considered before the optimal one can be determined. If trying to tackle a design challenge,might want to look at optimization, which is a type of computational scenario in which trying to discover the best potential answers. The model incorporates data on the constraints of the current design and the hidden costs of ad hoc rules and policy choices, allowing for more accurate comparisons between them. What-if analysis is aided by a well-designed optimization model, which shows where improvements are achieved or where trade-offs may be necessary.Multiple fields of engineering make use of optimization techniques [52].

4.3 Python Software

Python can be used in a variety of projects due to its high level and versatility. Large indentation and other design options make the code easier to read. Python is an interpreted, dynamically written, and garbage-collecting programming language [66]. It is friendly to a number of programming styles, including structural (particularly procedural), object-oriented, and functional. The large standard library that comes with it has earned it the nickname "battery included" among language enthusiasts. As an alternative to ABC, Guido van Rossum began developing python in the late 1980s. The first version, Python 0.9.0, was published in 1991. The year 2000 saw the introduction of Python 2.0, which had a number of improvements over its predecessor. Python is still the number one competitor among programming languages [71]. It can also be used for more than one programming paradigm. Many of its properties are valid for both functional and object-oriented programming, and it fully supports both structured and object-oriented programming (including meta- and object-descriptive programming). Extensions allow the use of many other paradigms, such as design by nodes and logic programming. Python stands out from other scripting tools due to its emphasis on readability, coherence, and overall program quality. Python code can be read, modified, and reused more easily than that of other scripting languages. Python code is standardized, it can understand it even if you don't write it. More complex strategies for reusing programs in python are also supported, including object-oriented programming (OOP). The vast majority of Python applications run unmodified on all major computer systems. For example, copying the script code is often all that is needed for a Python port between Linux and Windows. In addition, Python provides a variety of libraries and frameworks for developing web-based systems, database access applications, and graphical user interfaces. Python portability extends to operating system interfaces such as program startup and directory handling [44].

The Python Standard Library is a large set of pre-made portable features. From network scripting to in-text pattern matching, this library can handle it all. In addition, Python may be augmented with a large number of third-party application support tools and custom libraries. There are many useful tools for creating websites, working with numbers, connecting to serial ports, making games, and much more in the third-party realm in python. For example, the NumPy add-on has been hailed as a free and more efficient alternative to matlab's numerical programming environment [56].

4.4 Package Ipopt

Ipopt is a software package for large scale nonlinear optimization. It is designed to find (local) solutions to problems of mathematical optimization of the model.

$$\begin{cases} \min\limits_{x \in R^n} f(x) \\ subject \ \ to \ \ g^L \leq g(x) \leq g^U \\ \qquad\qquad\qquad x^L \leq x \leq x^U \end{cases}$$

where $x \in R^n$ are the optimization variables,the vectors x^L and x^U are the bounds on the variables x, $g : R^n \rightarrow R^m$ are the general nonlinear constraints and $f : R^n \rightarrow R$ is the objective function [59].The functions $g(x)$ and $f(x)$ can be nonlinear (linear), non-convex(convex),(but should be twice continuously differentiable).The constraint functions, $g(x)$, Contains g^L , g^U lower and upper bounds. When the parameters in a problem change, the Ipopt project provides a set of tools for working with natural language processing sensitivity theory to rapidly generate working solutions. In any optimization study we consider sensitivity to nonlinear programming problems as an essential step. For parametric nonlinear programs, sensitivity yields first-order estimates and sheds light on the regularity and curvature of (KKT)scores; it also determines which variables are most important to the optimization process. Sensitivity can be integrated into processing solutions with minimal additional computation compared to (NLP) algorithms that rely on exact second derivatives, and it can provide valuable information. The functionality of this program grants ipopt the ability to compute sensitivities and approximate perturbed solutions. The Applications,General Nonlinear optimization and Process Engineering , DAE/ PDE systems process Design and operations ,Nonlinear model Predictive control , Design Under Uncertainty [29].

4.5 Developing Mathematical Models Based on Numerical Analysis

4.5.1 Solving NLP of Unconstrained Optimization Problem

Consider the optimization problem

let $f : S \longrightarrow R$ where S is convex set and $S \subseteq R^n$, consider the

$$\begin{cases} \min\ 3x - 1.7x^2 + 2.2x^3 - 1.5x^4 \\ \qquad x \geq 0, \quad x \in S \end{cases}$$

```
Algorithm 1 : Golden-Section Search Method
step 1 : Given interval $[a, b]$ , tolerance
step 2 : The distance $d = (b - a)$
step 3 : compute $x_1 = a + d,\ x_2 = b - d$ and compute $f(x_1),\ f(x_2)$
step 4 : $iff(x_1) > f(x_2)$
then $x_2 = a,\ x_1 = x_2\ ,\ x_2 = b - d,\ x_1 = a + d$
step 5 : Repeat steps until find $x^*, f(x^*)$
```

Code 8:Use Golden-Section Search Method to Solve Unconstrained Optimization Problems

```python
# Golden-section search method
import math
def cal_f(_x):  # 3x - 1.7x^2 + 2.2x^3 - 1.5x^4
    return 3 * _x - 1.7 * _x ** 2 + 2.2 * _x ** 3 - 1.5 * _x ** 4
def cal_d(_b, _a):
    return R * (_b - _a)
def check_e(_b, _a, _xopt, _e):
    _ea = (1 - R) * abs((b - _a) / _xopt)

    if _ea < _e:
        return 1
    return 0
def search(_b, _a, _e, _n):
    _x = [0, 0]
    _f = [0, 0]
    _result = 0
    for i in range(0, _n):
        _d = cal_d(_b, _a)
        _x[0] = _a + _d
        _x[1] = _b - _d
        _f[0] = cal_f(_x[0])
        _f[1] = cal_f(_x[1])
        if _f[0] > _f[1]:
            _a = _x[1]
            _result = _x[0]
            if check_e(_b, _a, _x[0], _e):
                break
        else:
            _b = _x[0]
            _result = _x[1]
            if check_e(_b, _a, _x[1], _e):
                break
        if i % 1 == 0:
            print("Iteration {}: a = {}\t b = {}\t x1 = {}
\t fx1 = {}\t x2 = {}\t fx2 = {}\t d = {}\t"
                  .format(i, round(_a, 5), round(_b, 5),
round(_x[0], 5), round(_f[0], 5), round(_x[1], 5),
                          round(_f[1], 5), round(_d, 5)))
    return _result
R = (math.sqrt(5) - 1) * 0.5
b = 4
a = 2
eS = 1 / 100
n = 15
result = search(b, a, eS, n)
print("x = {}, fx = {}".format(result, cal_f(result)))
```

Iteration	a	b	x_1	$f(x_1)$	x_2	$f(x_2)$	d
0	2	3.23607	3.23607	-98.03808	2.76393	-45.78183	1.23607
1	2	2.76393	2.76393	-45.78183	2.47214	-25.75954	0.76393
2	2	2.47214	2.47214	-25.75954	2.2918	-16.95203	0.47214
3	2	2.2918	2.2918	-16.95203	2.18034	-12.6365	0.2918
4	2	2.18034	2.18034	-12.6365	2.11146	-10.34914	0.18034
5	2	2.11146	2.11146	-10.34914	2.06888	-9.06913	0.11146
6	2	2.06888	2.06888	-9.06913	2.04257	-8.32651	0.06888
7	2	2.04257	2.04257	-8.32651	2.02631	-7.88545	0.04257
8	2	2.02631	2.02631	-7.88545	2.01626	-7.61956	0.02631
9	2	2.01626	2.01626	-7.61956	2.01005	-7.45775	0.01626
10	2	2.01005	2.01005	-7.45775	2.00621	-7.35871	0.01005
11	2	2.00621	2.00621	-7.35871	2.00384	-7.29787	0.00621
12	2	2.00384	2.00384	-7.29787	2.00237	-7.2604	0.00384
13	2	2.00237	2.00237	-7.2604	2.00147	-7.2373	0.00237
14	2	2.00147	2.00147	-7.2373	2.00091	-7.22304	0.00147

Table 4.1: **Our Results Using the Golden Section Search Method**

The model above using the golden section search method to find the value of x that minimize of objective function in the range $[2, 3.23607]$ and the distance $d = (b - a)$ as part of its calculation. By using the Python program it was found ($x_1 = 3.23607, f(x_1) = -98.03808, x_2 = 2.76393, f(x_2) = -45.78183$). This , the value of x that minimizer f is located in the interval $[2, 2.00147]$.

4.5.2 Solving Dynamic Optimization Model With Multivariable Based on Constraint Differential Equation

Consider the dynamic optimization model which include the large scale of variables

$$\begin{cases} \min\limits_{u(t)} & x_2 + 4(t_f) \\ \text{s.t} & \dfrac{dx_1}{dt} = u(t) + 3 \\ & \dfrac{dx_2}{dt} = x_1^2 + u^2(t) - 5 \\ & x(0) = [1, 0]^T \\ & t_f = \{0, 0.1, 0.2, 0.3, ..., 0.9, 1\} \end{cases}$$

Algorithm 2 : Solving Dynamic Optimization Model With Multivariable Based on Constraint Differential Equation

---

Initialization step

step 1: choose initial condition $x_1(0) = 1, x_2(0) = 0$

step 2: x_1, x_2 two variable

step 3: Dynamic optimization with parameter u

step 4: $t_f \longrightarrow$ finial time

step 5: Using lagrange method to find optimal solution

$$\text{minimize } L(x, \lambda) = f(x) + \sum_{i=1}^{m} \lambda_j g_j(x))$$

Code 9: Solving Dynamic Optimization Model With Multivariable Based on Constraint Differential Equation

```
import numpy as np
import matplotlib.pyplot as plt
from gekko import GEKKO
m = GEKKO()
nt = 60
m.time = np.linspace(0,1,nt)
# Variables
x1 = m.Var(value=1)
x2 = m.Var(value=0)
u = m.Var(value=-0.75)
p = np.zeros(nt)
p[-1] = 1.0
final = m.Param(value=p)
# Equations
m.Equation(x1.dt()==u+3)
m.Equation(x2.dt()==x1**2 + u**2-5)
# Objective Function
m.Obj(x2+4*final)
m.options.IMODE = 6
m.solve()
plt.figure(1)
plt.plot(m.time,x1.value,'k:',lw=2,label=r'$x_1$')
plt.plot(m.time,x2.value,'b-',lw=2,label=r'$x_2$')
plt.plot(m.time,u.value,'r--',lw=2,label=r'$u$')
plt.legend(loc='best')
plt.xlabel('Time')
plt.ylabel('Value')
plt.show()
```

Jacobian of the equality constraint with nonzero indices	647
Jacobian of the inequality constraint with nonzero indices	0.0
Lagrangian Hessian number of non-zero elements	118
The final number of variables	295
Lower-bounded variables	0.0
Lower and upper bounded variables	0.0
Upper-bounded variables	0.0
The final numbers of equality constraints	236
The final numbers of inequality constraints	0.0

Table 4.2: **Numerical Result For Dynamic Optimization Model With Multivariable Based on Constraint Differential Equation**

final value CPU sec in IPOPT	0.0050
CPU secs final in NLP function evaluation	0.009
The objective function final value	-26.8919075309659
Solution time	1.920000000245636E-002 sec
Objective	-26.891907530965
Solver	IPOPT

Table 4.3: **Numerical Result For Dynamic Optimization Model With Multivariable Based on Constraint Differential Equation**

iter	objective	inf_pr	inf_du	Lg(mu)	$\|d\|$	Alpha_du	Alpha_pr
0	3.1000000e+000	1.04e+00	1.80e-01	0.0	0.00e+000	0.00e+0	0.00e+000
1	3.6030873e+010	2.13e+00	5.35e-01	-11.0	1.15e+000	1.00e+0	1.00e+000
2	3.7234280e+010	3.12e-01	1.22e-14	-11.0	1.25e+000	1.00e+0	1.00e+000
3	3.9891908e+010	4.00e-15	1.41e-14	-11.0	1.14e-010	1.00e+0	1.00e+000
4	4.0091908e+010	5.22e-15	1.52e-14	-11.0	2.14e-010	1.00e+0	1.00e+000
5	-5.7811908e+010	5.21e-15	1.56e-14	-11.0	2.16e-010	1.00e+0	1.00e+000
6	-5.6791708e+010	5.35e-15	1.62e-14	-11.0	2.24e-010	1.00e+0	1.00e+000
7	-5.4371608e+010	5.47e-15	1.61e-14	-11.0	2.34e-010	1.00e+0	1.00e+000
8	-5.3524908e+010	5.59e-15	1.65e-14	-11.0	3.14e-010	1.00e+0	1.00e+000
9	-2.0001908e+010	6.02e-15	1.72e-14	-11.0	3.24e-010	1.00e+0	1.00e+000
10	-2.9241908e+010	6.43e-15	1.75e-14	-11.0	3.54e-010	1.00e+0	1.00e+000
11	-2.7352908e+010	6.63e-15	1.77e-14	-11.0	4.13e-010	1.00e+0	1.00e+000
12	-2.6891908e+010	6.71e-15	1.79e-14	-11.0	4.14e-010	1.00e+0	1.00e+000

Table 4.4: **Our Results The Dynamic Optimization Model Using The Lagrange Method**

- **Iter** : The numbers of iteration and regular repetition during the recovery phase.
- **Objective** : The value of objective function in itreation point.
- inf $-pr$:Unscaled constraint violation at current point. This quantity is the infinity (maximum) of the (unscaled) constraints ($g^L \leq g(x) \leq g^U$)in (NLP)).
- inf $-du$: The scaled dual infeasibility at the current point.
- **Lg(mu)** : $\log_{10}$ of the value of parameter μ.
- $\|d\|$: The Infinity norm or max of the step primal
- **alpha -du** : the stepsize for the dual variables.
- **alpha -pr** : The size of the scores for the initial variables .

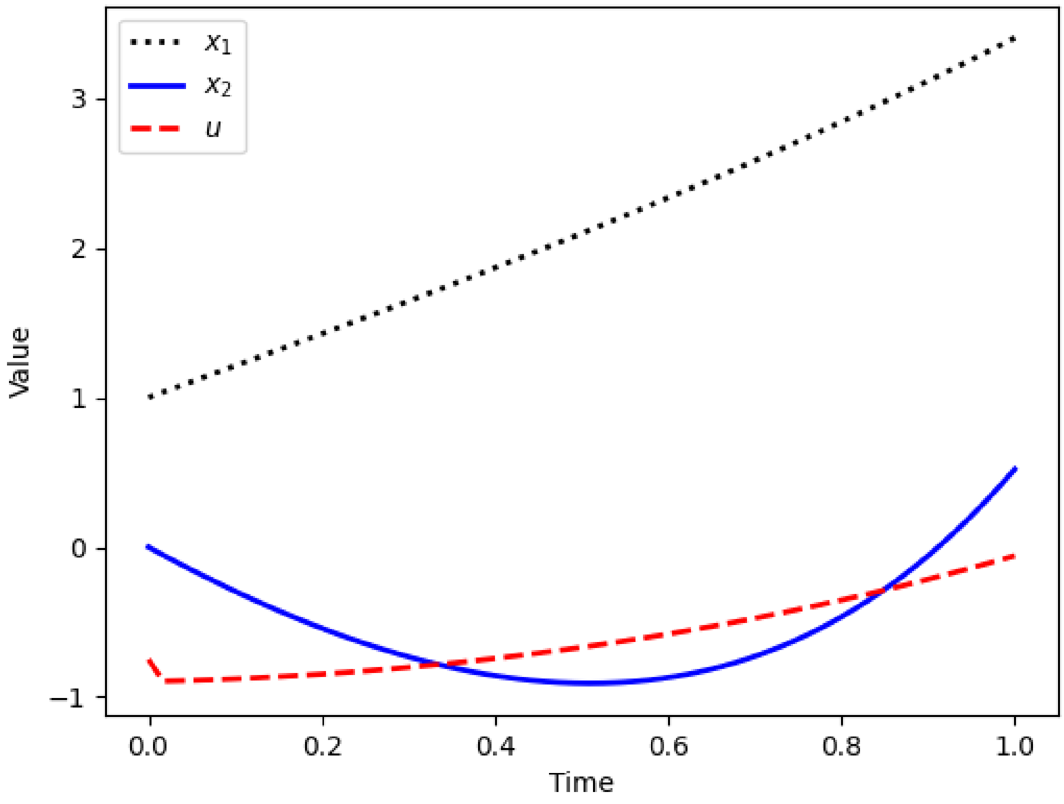


Figure 4.1: The Optimal Solution With Time Via value

As we can see in the figure the x-axis represents the time and the y-axis represents the value of the objective function relationship, in this mathematical analysis model the implementation we obtained based on two variables (x_1, x_2) depends on $u(t)$ changes which represented the control variables at the time t.

4.5.3 Solving Dynamic Optimization Model With Initial Value Problem Constraint

Consider the dynamic optimization model with initial value problem constraint

$$\begin{cases} \min\limits_{z(t)} & y+5(t_f) \\ \text{s.t} & \dfrac{dx}{dt}=z(t)+8 \\ & \dfrac{dy}{dt}=x^2+z^2(t)+10 \\ & x(0)=[1,0]^T \\ & t_f=\{0,0.1,0.2,0.3,...,0.9,1\} \end{cases}$$

Algorithm 3 : Solving Dynamic Optimization Model With Initial Value Problem Constraint

```
Initialization step
step 1 : $x, y, z$ three variables
step 2 : Dynamic optimization with parameter $z$
step 3 : choose initial condition $x(0) = 1, y(0) = 0$
step 4 : $t_f \longrightarrow$ finial time
step 5 :$x(t_f) = 1$
step 6 : Using Euler method to find optimal solution
```

Code 10:Solving Dynamic Optimization Model With Initial Value Problem Constraint

```python
import numpy as np
import matplotlib.pyplot as plt
from gekko import GEKKO
m = GEKKO()
nt = 60
m.time = np.linspace(0,1,nt)
# Variables
x = m.Var(value=1)
y = m.Var(value=0)
z = m.Var(value=-0.48)
p = np.zeros(nt)
p[-1] = 1.0
final = m.Param(value=p)
# Equations
m.Equation(x.dt()==z+8)
m.Equation(y.dt()==x**2 +z**2+10)
m.Equation(final*(x-1)==0)
# Objective Function
m.Obj(y+5*final)
m.options.IMODE = 6
m.solve()
plt.figure(1)
plt.plot(m.time,x.value,'k:',lw=2,label=r'$x$')
plt.plot(m.time,y.value,'b-',lw=2,label=r'$y$')
plt.plot(m.time,z.value,'r--',lw=2,label=r'$z$')
plt.legend(loc='best')
plt.xlabel('Time')
plt.ylabel('Value')
plt.show()
```

Jacobian of the equality constraint with nonzero indices	706
Jacobian of the inequality constraint with nonzero indices	0.0
The elements non-zero number in Lagrangian Hessian	118
the final number of variables	295
Lower-bounded variables	0.0
Lower and upper bounded variables	0.0
Upper-bounded variables	0.0
The final numbers of equality constraints	295
The final numbers of inequality constraints	0.0

Table 4.5: **Numerical Result for Solving Dynamic Optimization Model With Initial Value Problem Constraint**

iter	objective	inf_pr	inf_du	Lg(mu)	$\|\|d\|\|$	Alpha_du	Alpha_pr
0	5.0000010e+000	1.12e+01	1.00e+00	0.0	0.00e+100	0.00e+100	0.00e+000
1	5.1200000e+000	1.22e+01	1.00e+00	-11.0	1.00e+000	1.00e+000	1.00e+010
2	5.2450000e+000	1.23e+01	1.00e+00	-11.0	1.10e+000	1.00e+000	1.20e+010
3	7.1072000e+000	1.24e+01	1.00e+00	-11.0	1.22e+000	1.00e+000	1.27e+010
4	7.2893000e+000	1.26e+01	1.00e+00	-11.0	2.10e+000	1.00e+000	1.20e+010
5	7.1972948e+010	1.66e+01	8.80e-01	-11.0	3.07e+010	1.00e+000	1.25e-010
6	7.2892948e+010	1.62e+01	8.10e-01	-11.0	3.09e+010	1.00e+000	1.28e-010
7	2.0172948e+001	1.16e+01	8.80e-01	-11.0	3.07e+010	1.00e+000	1.27e-010
8	2.1774193e+030	2.17e-05	1.51e+01	-11.0	9.55e+010	1.00e+000	1.00e+100
9	1.3667014e+030	1.65e-04	1.75e-10	-11.0	3.81e+030	1.00e+000	1.00e+100
10	1.3667014e+030	9.009e-13	5.68e-14	-11.0	1.65e-040	1.00e+000	1.00e+000

Table 4.6: **Our Results the Dynamic Optimization Model Using the Euler Method**

Total CPU secs in IPOPT (function evaluations)	0.021
Total CPU secs in NLP function evaluations	0.017
The objective function final value	1366.70136317053
Solution time	4.370000000926666E-002 sec
Objective	1366.70136317053
Solver	IPOPT

Table 4.7: **Numerical Result for Solving Dynamic Optimization Model With Initial Value Problem Constraint**

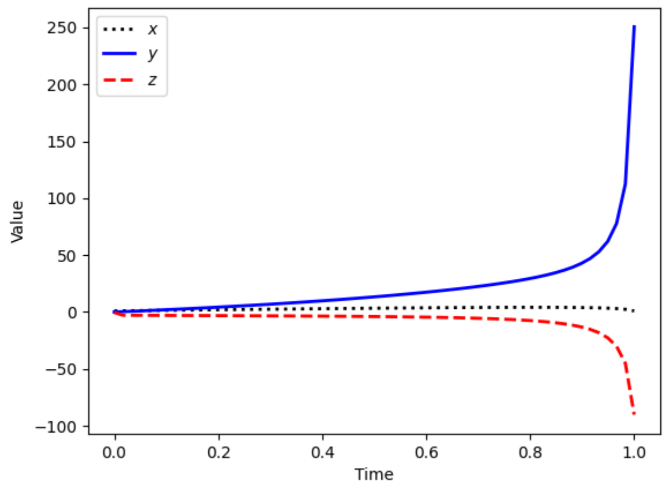


Figure 4.2: The Optimal Solution of the Dynamic Model Based on (Time Via Objective Function)

As we can see in the figure the y-axis represents the value of the objective function relationship and the x-axis represents the time , in this mathematical analysis model the implementation we obtained based on two variables (x,y) depends on z(t) changes which represented the control variables at the time t.

4.5.4 Solving Nonlinear Numerical Optimization Model with Large Scale Variable

Consider the nonlinear numerical optimization model with large scale

$$\begin{cases} \min & x_2x_4(x_1x_2x_4)x_3 \\ \text{s.t} & x_1x_2x_3x_4 \geq 30 \\ & x_1^2 + x_2^2 + x_3^2 + x_4^2 = 50 \\ & 1 \leq x_1, x_2, x_3, x_4 \leqslant 5 \\ & x_0 = (1,3,3,2) \end{cases}$$

Algorithm 4 : Solving Nonlinear Numerical Optimization Model with Large Scale Variable

---

Initialization step

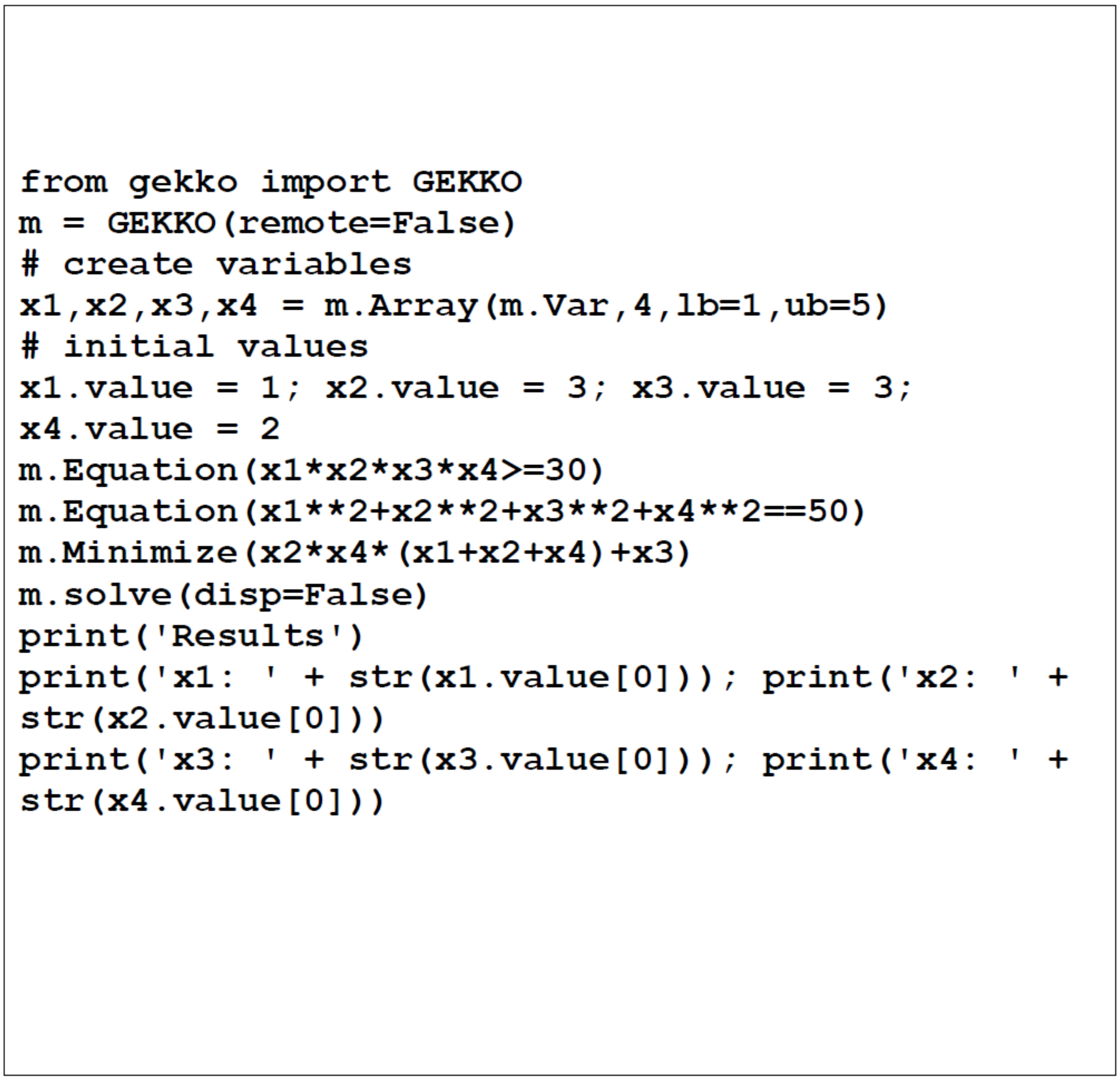
step 1 : x_1, x_2, x_3, x_4 four variables

step 2 : choose initial value $x_0 = (1,3,3,2)$

step 3 :$1 \leq x_1, x_2, x_3, x_4 \leq 5$

step 4 : Using KKT with inequality constraint method to find optimal solution

Code 11:Solving Nonlinear Numerical Optimization Model With Large Scale Variable



```
from gekko import GEKKO
m = GEKKO(remote=False)
# create variables
x1,x2,x3,x4 = m.Array(m.Var,4,lb=1,ub=5)
# initial values
x1.value = 1; x2.value = 3; x3.value = 3;
x4.value = 2
m.Equation(x1*x2*x3*x4>=30)
m.Equation(x1**2+x2**2+x3**2+x4**2==50)
m.Minimize(x2*x4*(x1+x2+x4)+x3)
m.solve(disp=False)
print('Results')
print('x1: ' + str(x1.value[0])); print('x2: ' +
str(x2.value[0]))
print('x3: ' + str(x3.value[0])); print('x4: ' +
str(x4.value[0]))
```

The numerical result for the model [4] are:

$x_1 =$ 4.7400842791 , $x_2 =$ 1.125077893 $x_3 = 5$, $x_4 = 1.125077893$

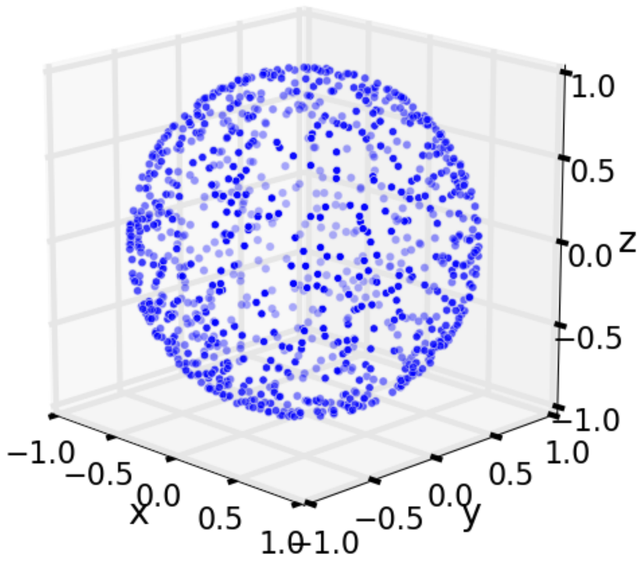


Figure 4.3: The Optimal Solution of the Nonlinear Model With Large Scale

4.5.5 The Large Scale Variables in Control Optimization Problem

Consider the large scale variables in control optimization problem

$$\begin{cases} \min\limits_{u(t)} \quad x_4 + 12(t_f) \\ \text{s.t} \quad \dfrac{dx_1}{dt} = x_2 \\ \qquad \dfrac{dx_2}{dt} = x_3 u(t) + 15t - 8 \\ \qquad \dfrac{dx_3}{dt} = u(t) \\ \qquad \dfrac{dx_4}{dt} = x_1^2 + x_2^2 + 0.007(x_2 + 15t - 8 - 0.1x_3 u^2(t))^2 \\ \qquad x(0) = [0 \ \ -1 \ \ -\sqrt{5} \ \ 0]^T \\ \qquad -3 \le u \le 9 \\ \qquad t_f = \{0, 0.1, 0.2, 0.3, ..., 0.9, 1\} \end{cases}$$

Algorithm 5 : The Large Scale Variables in Control Optimization Problem

---

Initialization step

step 1 :The initial condition $x_1 = 0$, $x_2 = -1$, $x_3 = -\sqrt{5}$, $x_4 = 0$

step 2 : Control optimization problem with parameter u , $-3 \le u \le 9$

step 3 : x_1, x_2, x_3, x_4 four variables

step 4 : The time $(t = 0)$, $(t_f = 1)$ finial time

step 5 : Using lagrange method to find optimal solution

Code 12: The Large Scale Variables in Control Optimization Problem

```
import numpy as np
import matplotlib.pyplot as plt
from gekko import GEKKO
m = GEKKO()
nt = 70
m.time = np.linspace(0,1,nt)
# Parameters
u = m.MV(value=2,lb=-3,ub=9)
u.STATUS = 1
u.DCOST = 0
# Variables
t = m.Var(value=0)
x1 = m.Var(value=0)
x2 = m.Var(value=-1)
x3 = m.Var(value=-np.sqrt(5))
x4 = m.Var(value=0)
p = np.zeros(nt)
p[-1] = 1.0
final = m.Param(value=p)
# Equations
m.Equation(t.dt()==1)
m.Equation(x1.dt()==x2)
m.Equation(x2.dt()==-x3*u+16*t-8)
m.Equation(x3.dt()==u)
m.Equation(x4.dt()==x1**2+x2**2 +0.008*(x2+15*t-8-0.1*x3*(u**2))**2)
# Objective Function
m.Obj(x4+12*final)
m.options.IMODE = 6
m.options.NODES = 4
m.options.MV_TYPE = 1
m.options.SOLVER = 3
m.solve()
print(m.path)
print('Objective = min x4+12(tf): ' + str(x4[-1]))
plt.figure(1)
plt.subplot(2,1,1)
plt.plot(m.time,u,'r-',lw=2,label=r'$u$')
plt.legend(loc='best')
plt.subplot(2,1,2)
plt.plot(m.time,x1.value,'r--',lw=2,label=r'$x_1$')
plt.plot(m.time,x2.value,'g:',lw=2,label=r'$x_2$')
plt.plot(m.time,x3.value,'k-',lw=2,label=r'$x_3$')
plt.plot(m.time,x4.value,'b-',lw=2,label=r'$x_4$')
plt.legend(loc='best')
plt.xlabel('Time')
plt.ylabel('Value')
plt.show()
```

Jacobian of the equality constraint with nonzero indices	9021
Jacobian of the inequality constraint with nonzero indices	276
The elements non-zero number in Lagrangian Hessian	2277
the final number of variables	2553
Lower-bounded variables	138
Lower and upper bounded variables	207
Upper-bounded variables	0.0
The final numbers of equality constraints	2346
The final numbers of inequality constraints	138

Table 4.8: **Numerical Results For the Large Scale Variables in Control Optimization Problem**

Total CPU secs in IPOPT (function evaluations)	0.199
Total CPU secs in NLP function evaluations	0.343
The objective function final value	35.9776469406198
Solution time	0.573799999983748 sec
Objective	35.9776469406198
Solver	IPOPT

objective	inf_pr	inf_du	Lg(mu)	$\|d\|$	Alpha_du	Alpha_pr
1.2000000e+010	3.53e+000	1.00e+00	0.0	0.00e+00	0.00e+000	0.00e+000
-5.9536729e+010	2.24e+010	1.49e+01	-0.8	2.06e+01	6.22e-020	1.00e+000
-2.4214955e+010	1.10e+010	4.24e+00	-0.9	2.51e+02	6.04e-010	1.00e+000
3.8627094e+010	1.75e-010	1.22e+00	-2.2	1.14e+02	7.06e-010	1.00e+000
3.7772392e+010	1.16e-010	3.88e-01	-1.8	5.74e+01	7.45e-010	1.00e+000
3.6379894e+010	1.63e-010	1.42e-01	-2.4	3.69e+01	7.11e-010	1.00e+000
3.5928771e+010	2.50e-010	5.71e-02	-3.1	1.83e+02	7.92e-010	1.00e+000
3.5960007e+010	2.56e-010	4.69e-02	-3.4	1.70e+02	7.99e-010	1.00e+000
3.5982288e+010	1.19e-010	1.55e-02	-3.6	1.36e+02	6.78e-010	1.00e+000
3.5981228e+010	5.48e-020	9.91e-03	-3.7	2.61e+02	6.87e-010	1.00e+000
3.5977896e+010	1.04e-020	5.21e-03	-9.5	1.20e+02	7.82e-010	1.00e+000
3.5977719e+010	2.19e-030	1.87e-03	-5.2	9.22e+01	8.81e-010	1.00e+000
3.5977650e+010	4.03e-040	3.39e-04	-6.4	3.55e+01	9.46e-010	9.91e-010
3.5977648e+010	3.91e-050	3.91e-05	-8.1	1.14e+01	9.88e-010	1.00e+000
3.5977646e+010	2.88e-060	2.62e-06	-11.0	2.15e+00	9.98e-010	9.86e-010
3.5977647e+010	3.62e-080	3.00e-06	-11.0	7.76e+00	6.01e-010	1.00e+000
3.5977642e+010	6.18e-120	3.09e-05	-11.0	1.98e+01	5.47e-010	1.00e+000
3.5977649e+010	5.33e-150	2.10e-05	-11.0	4.19e+01	6.08e-010	1.00e+000
3.5977637e+010	4.44e-150	1.21e-04	-10.7	2.30e+02	3.88e-010	1.00e+000
3.5977657e+010	3.55e-150	6.06e-06	-10.7	3.71e+02	6.21e-010	1.00e+000
3.5977667e+010	3.55e-150	2.53e-06	-10.7	9.70e+02	5.83e-010	1.00e+000
3.5977647e+010	5.33e-150	1.02e-06	-10.7	2.27e+03	5.96e-010	1.00e+000
3.5977647e+010	5.33e-150	1.08e-07	-11.0	2.47e+03	7.68e-010	1.00e+000

Table 4.9: **Our Results the Control Optimization Model Using the Lagrange Method , where we get the optimal solution by 22 iteration**

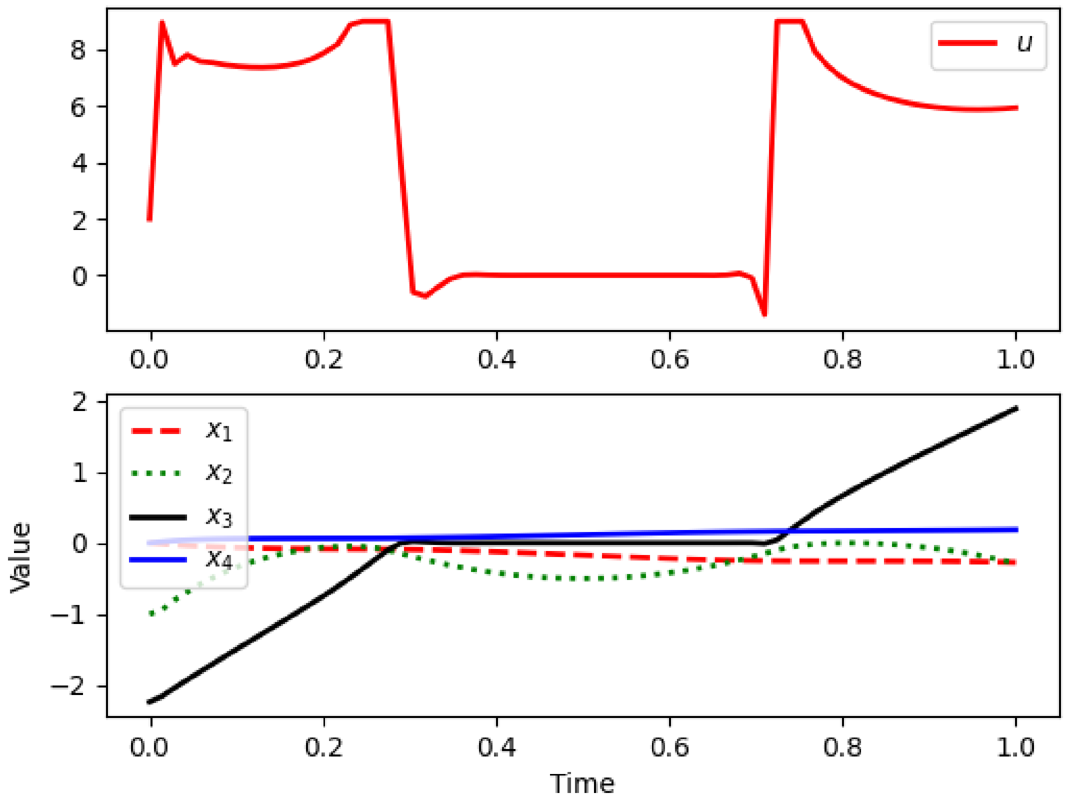


Figure 4.4: The Optimal Solution With the Large Scale Variables in Control Optimization Problem

As we can see in the figure the x-axis represents the time and the y-axis represents the value of the objective function relationship, in this mathematical analysis model the implementation based on four variables (x_1, x_2, x_3, x_4) Where the red color symbolizes the variable x_1, the green color the variable x_2, the black color the variable x_3 and the blue color the variable x_4 , $u(t)$ changes which represented the control variables at the time t.

4.5.6 The Large Scale Variables in Control Optimization Problem

Consider the large scale variables in Control optimization problem

$$\begin{cases} \max\limits_{u(t)} & x_2(t_f) \\ \text{s.t} & \dfrac{dx_1}{dt} = -(u(t) + 0.3u^2)x_1 \\ & \dfrac{dx_2}{dt} = u(t)\ x_1 \\ & x(0) = [1, 0]^T \\ & 0 \leq u \leq 7 \\ & t_f = \{0, 0.1, 0.2, 0.3, ..., 0.9, 1\} \end{cases}$$

Algorithm 6 : Control Optimization Problem With Large Scale Variables

---

Initialization step

step 1 : Control optimization problem with parameter u , $0 \leq u \leq 7$

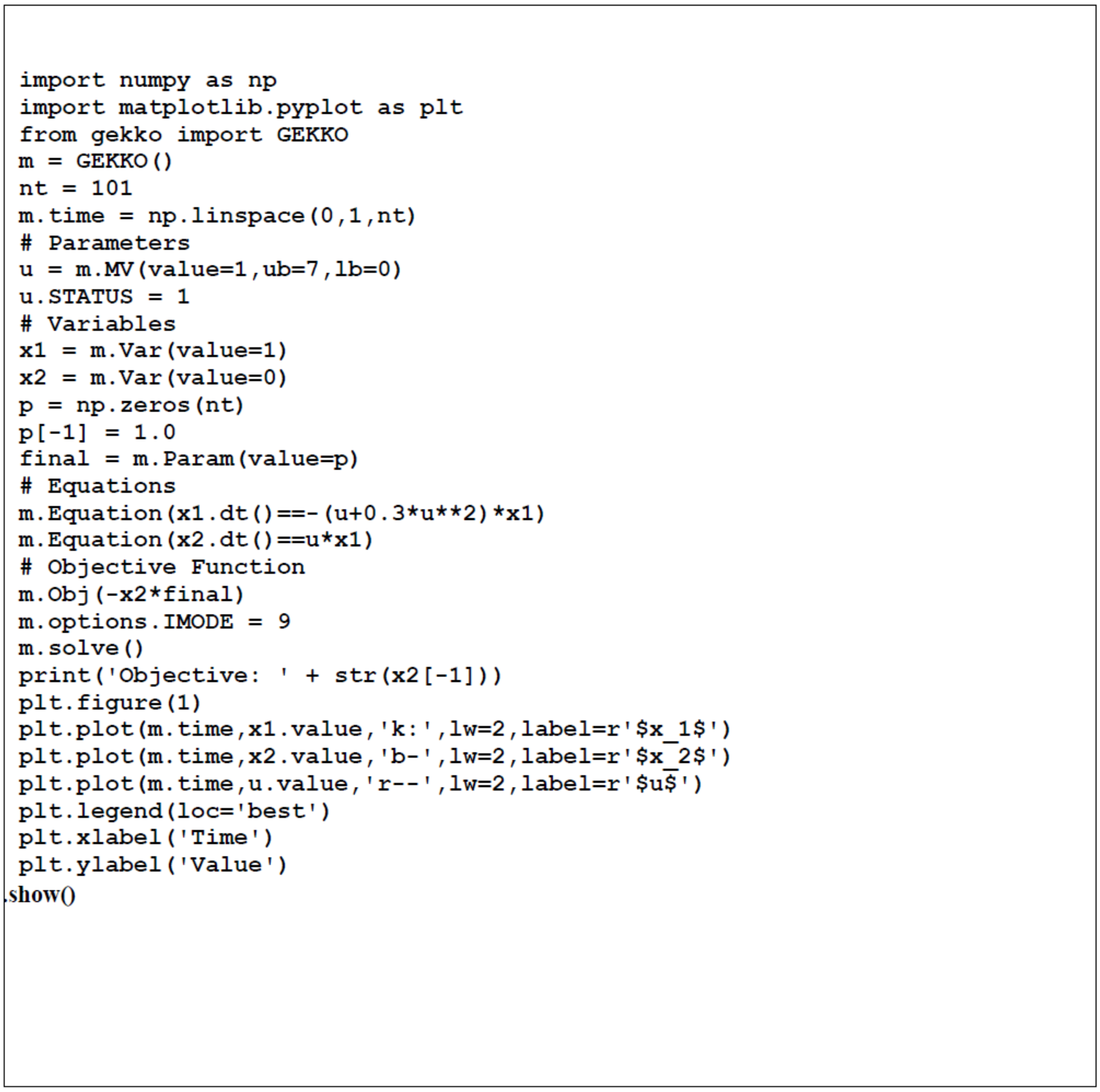
step 2 : The initial condition $x_1 = 1$, $x_2 = 0$

step 3 : The time $(t = 0)$, $(t_f = 1)$ finial time

step 4 : x_1, x_2 two variables

step 5 : Using KKT condition method to find optimal solution

Code 13:Solving Control optimization problem with large scale Variables



```
import numpy as np
import matplotlib.pyplot as plt
from gekko import GEKKO
m = GEKKO()
nt = 101
m.time = np.linspace(0,1,nt)
# Parameters
u = m.MV(value=1,ub=7,lb=0)
u.STATUS = 1
# Variables
x1 = m.Var(value=1)
x2 = m.Var(value=0)
p = np.zeros(nt)
p[-1] = 1.0
final = m.Param(value=p)
# Equations
m.Equation(x1.dt()==-(u+0.3*u**2)*x1)
m.Equation(x2.dt()==u*x1)
# Objective Function
m.Obj(-x2*final)
m.options.IMODE = 9
m.solve()
print('Objective: ' + str(x2[-1]))
plt.figure(1)
plt.plot(m.time,x1.value,'k:',lw=2,label=r'$x_1$')
plt.plot(m.time,x2.value,'b-',lw=2,label=r'$x_2$')
plt.plot(m.time,u.value,'r--',lw=2,label=r'$u$')
plt.legend(loc='best')
plt.xlabel('Time')
plt.ylabel('Value')
.show()
```

Jacobian of equality constraint with nonzero indices	1298
Jacobian of inequality constraint with nonzero indices	400
The elements non-zero number in Lagrangian Hessian	0.0
the final number of variables	800
Lower-bounded variables	200
Lower and upper bounded variables	0.0
Upper-bounded variables	0.0
The final numbers of equality constraints	600
The final numbers of inequality constraints	200

Total CPU secs in IPOPT (function evaluations)	0.012
Total CPU secs in NLP function evaluations	0.018
The objective function final value	-0.557740968627238
Solution time	8.040000000255532E-002 sec
Objective	-0.558157637615552
Solver	IPOPT

Table 4.10: **Numerical Results For the Large Scale Variables in Control Optimization Problem**

objective	(inf_pr)	(inf_du)	(Lg(mu))	($\|d\|$)	(alpha_du)	(alpha_pr)
1.1899988e-040	1.30e+000	1.010e+00	0.0	0.00e+000	0.00e+000	0.00e+0000
-5.5770830e-010	2.22e-160	1.00e-02	-8.0	1.28e+000	9.90e-010	1.00e+000
-5.5770874e-010	2.22e-160	9.50e-05	-10.0	5.42e-005	9.91e-010	1.00e+000
-5.5774097e-010	1.11e-160	1.11e-16	-11.0	3.63e-003	1.00e+000	1.00e+000
-5.5815068e-010	2.22e-160	6.30e-02	-8.0	1.016e-001	9.40e-010	1.00e+000
-5.5815075e-010	2.22e-160	1.09e-03	-9.2	9.09e-006	9.90e-010	1.00e+000
-5.5815764e-010	1.11e-160	1.00e-16	-11.0	8.40e-004	1.00e+000	1.00e+000

Table 4.11: **Our Results the Control Optimization Model Using the KKT Condtion Method ,where we get the optimal solution by 6 iteration**

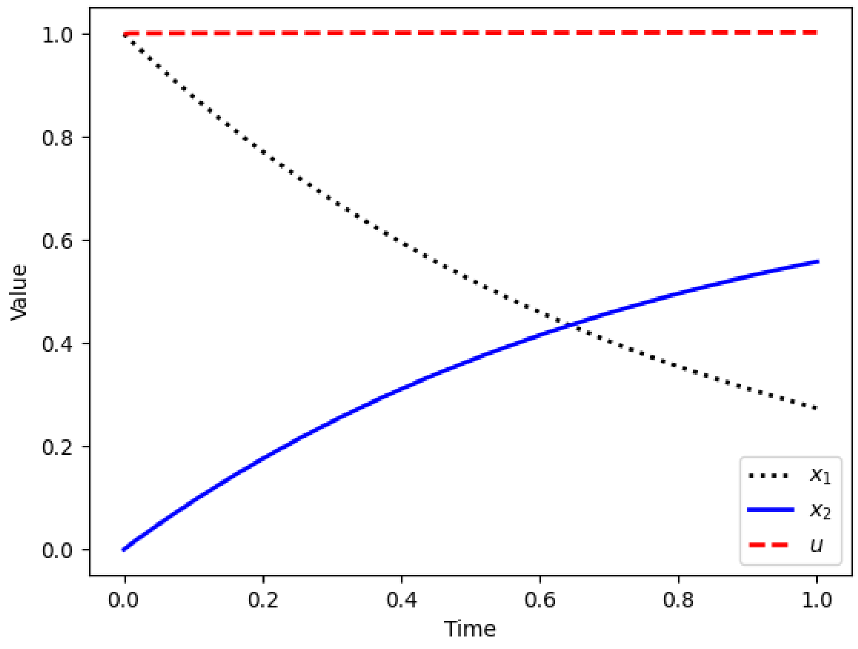


Figure 4.5: The Optimal Solution With the Large Scale Variables in Control Optimization Using KKT Condtion

As we can see in the figure the x-axis represents the time and the y-axis represents the value of the objective function relationship, in this mathematical analysis model the implementation we obtained based on four variables (x_1, x_2) depends on $u(t)$ changes which represented the control variables at the time t.

4.5.7 The Large Scale Variables in Control Optimization Problem

Consider the large scale variables in Control optimization problem

$$\begin{cases} \max\limits_{T(t)} & x_2(t_f) \\ \text{s.t} & \dfrac{dx_1}{dt} = -k_1 x_1^2 \\ & \dfrac{dx_2}{dt} = k_1 x_1^2 - k_2 x_2 \\ & k_1 = 3000 \exp(-\dfrac{3000}{T}) \\ & k_2 = 6.235 \exp(-\dfrac{6000}{T}) \\ & x(0) = [1, 0]^T \\ & 250 \leq T \leq 320 \\ & t_f = \{0, 0.1, 0.2, 0.3, ..., 0.9, 1\} \end{cases}$$

Algorithm 7 :The Large Scale Variables in Control Optimization Problem

---

Initialization step

step 1 : x_1, x_2 two variables

step 2 : Intermediates (Algebraic Equation)

$k_1 = 3000 \exp(-\dfrac{3000}{T})$

$k_2 = 6.235 \exp(-\dfrac{6000}{T})$

step 3 : The time $(t = 0)$, $(t_f = 1)$ finial time

step 4 : Control optimization problem with parameter T , $250 \leq T \leq 320$

step 5 : The initial condition $x_1 = 1$, $x_2 = 0$

step 6 : Using KKT condition method to find optimal solution

Code14:Solving Control optimization problems with large scale variables1

```
import numpy as np
import matplotlib.pyplot as plt
from gekko import GEKKO
m = GEKKO()
nt = 101
m.time = np.linspace(0, 1, nt)
# Parameters
T = m.MV(value=320, ub=398, lb=250)
T.STATUS = 1
T.DCOST = 0
# Variables
x1 = m.Var(value=1)
x2 = m.Var(value=0)
p = np.zeros(nt)
p[-1] = 1.0
final = m.Param(value=p)
# Intermediates
k1 = m.Intermediate(3000 * m.exp(-3000 / T))
k2 = m.Intermediate(6.235 * m.exp(-6000 / T))
# Equations
m.Equation(x1.dt() == -k1 * x1 ** 2)
m.Equation(x2.dt() == k1 * x1 ** 2 - k2 * x2)
# Objective Function
m.Obj(-x2 * final)
m.options.IMODE = 6
m.solve()
print('Objective: ' + str(x2[-1]))
plt.figure(1)
plt.subplot(2, 1, 1)
plt.plot(m.time, x1.value, 'k:', lw=2, label=r'$x_1$')
plt.plot(m.time, x2.value, 'b-', lw=2, label=r'$x_2$')
plt.ylabel('Value')
plt.legend(loc='best')
plt.subplot(2, 1, 2)
plt.plot(m.time, T.value, 'r--', lw=2, label=r'$T$')
plt.legend(loc='best')
plt.xlabel('Time')
plt.ylabel('Value')
plt.show()
```

Jacobian of equality constraint with nonzero indices	1797
Jacobian of inequality constraint with nonzero indices	400
Lagrangian Hessian number of non-zero elements	400
The final number of variables	900
Lower-bounded variables	200
Lower and upper bounded variables	100
Upper-bounded variables	0.0
The number final of equality constraints	600
The number final of inequality constraints	200

Table 4.12: **Numerical Results For Solving the Large Scale Variables in Control Optimization Problem**

final value CPU sec in IPOPT	0.062
CPU secs final in NLP function evaluation	0.141
The objective function final value	-0.612822209411815
Solution time	0.218300000000454 sec
Objective	-0.612822209411815
Solver	IPOPT

Table 4.13: **Numerical Results For Solving the Large Scale Variables in Control Optimization Problem**

objective	inf_pr	inf_du	lg(mu)	$\|d\|$	alpha_du	alpha_pr
0.0000000e+000	2.54e-010	1.00e+000	0.0	0.00e+00	0.00e+000	0.00e+000
-1.9904609e-010	1.01e-020	1.23e-020	-7.1	2.53e-01	9.88e-010	1.00e+000
-2.0310998e-010	3.22e-060	2.46e-030	-9.0	3.97e-01	8.01e-010	1.00e+000
-2.1597165e-010	3.87e-040	2.41e-030	-9.7	3.02e+01	3.98e-020	1.00e+000
-2.1701576e-010	1.37e-060	6.69e-050	-5.3	4.90e-01	9.86e-010	1.00e+000
-2.1947302e-010	1.57e-050	4.90e-050	-7.4	1.47e+00	8.19e-010	1.00e+000
-2.2709786e-010	1.49e-040	4.97e-050	-7.1	4.47e+00	8.28e-010	1.00e+000
-2.5203042e-010	1.52e-030	5.20e-050	-7.6	1.40e+01	5.03e-010	1.00e+000
-3.4766202e-010	1.89e-020	6.93e-040	-6.6	4.75e+01	6.92e-010	1.00e+000
-4.9336176e-010	3.58e-020	3.21e-030	-4.6	1.43e+02	1.50e-020	3.77e-010
-4.9564590e-010	9.90e-020	3.32e-030	-3.8	1.81e+02	1.00e+000	9.68e-010
-5.2557064e-010	5.90e-020	8.48e-04 0	-3.6	2.04e+02	1.00e+000	9.99e-010
-2.2187385e-010	1.16e+000	2.83e-02 0	-1.6	1.03e+04	1.35e-010	9.48e-020
-2.4898348e-010	2.24e+000	1.44e-01 0	-1.9	2.78e+03	9.59e-010	3.20e-010
-2.4495356e-010	1.72e+000	1.01e-010	-1.9	5.72e+03	3.82e-010	1.00e+000
-2.8662833e-010	1.04e-010	3.43e-020	-1.9	5.93e+03	6.40e-010	1.00e+000
-3.4493460e-010	9.86e-030	2.14e-030	-2.8	1.60e+03	9.31e-010	1.00e+000
-5.3951251e-010	2.67e-010	6.52e-030	-3.3	5.78e+03	8.43e-010	8.78e-010
-5.9608487e-010	2.75e-020	8.15e-040	-4.1	4.46e+03	9.03e-010	1.00e+000
-6.1160816e-010	2.96e-030	7.65e-050	-5.7	7.62e+02	9.78e-010	1.00e+000
-6.1281384e-010	2.80e-050	5.30e-070	-7.5	3.29e+02	9.96e-010	1.00e+000
-6.1282221e-010	1.26e-090	8.85e-110	-11.01	1.31e+001	1.00e+010	1.00e+001

Table 4.14: **Our Results the Control Optimization Problem Using the KKT Condtion Method, where we get the optimal solution by 21 iteration**

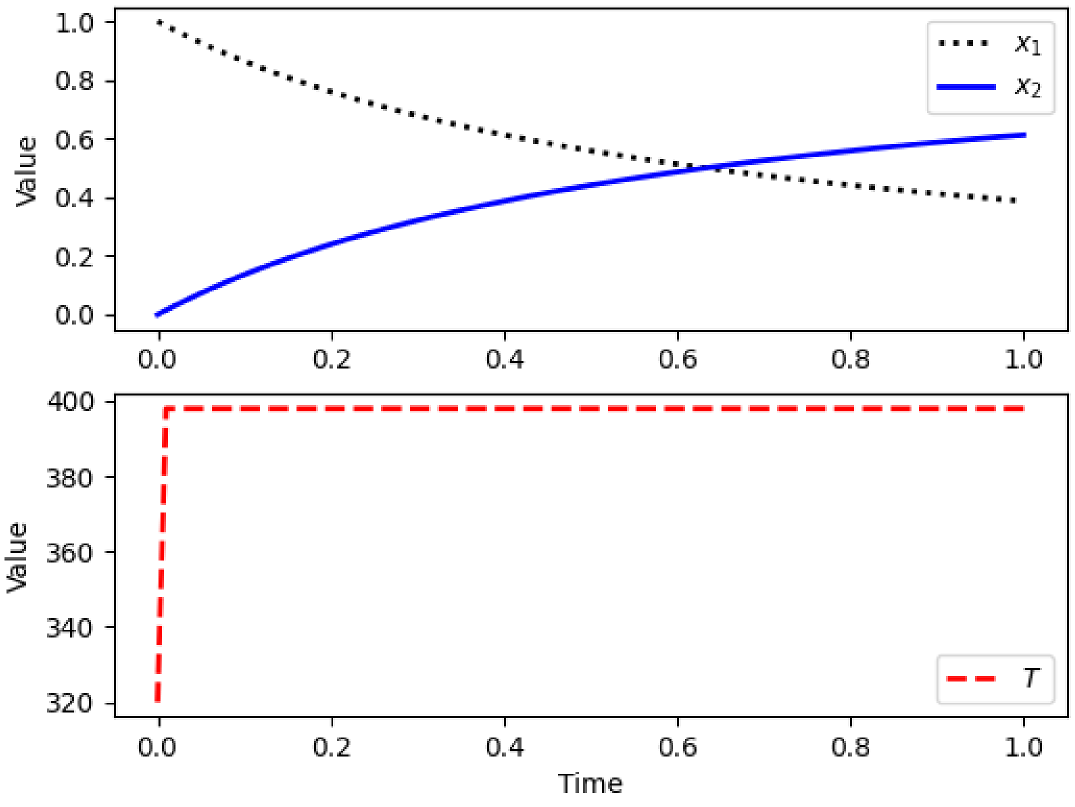


Figure 4.6: The Optimal Solution With Control Optimization Problme Wih Large Scale Variables

As we can see in the figure the y-axis represents the value of the objective function relationship and the x-axis represents the time ,in this mathematical analysis model the implementation we obtained based on two variables (x_1, x_2) depends in time t by Intermediates (Algebraic Equations) (k_1, k_2).

4.5.8 The Large Scale Variables in Control Optimization Problem

Consider the large scale variables in Control optimization problem

$$\begin{cases} \max\limits_{u(t)} & (1 - x_1(t_f) - x_2(t_f)) \\ \text{s.t} & \dfrac{dx_1}{dt} = u(t)(10x_2 - x_1) \\ & \dfrac{dx_2}{dt} = -u(t)(10x_2 - x_1) - (1 - u(t))x_2 \\ & x(0) = [1, 0]^T \\ & 0 \leq u \leq 1 \\ & t_f = \{0, 0.1, 0.2, 0.3, ..., 0.9, 1\} \end{cases}$$

Algorithm 8 :Control Optimization Problem With Large Scale Variables

Initialization step

step 1 : Control optimization problem with parameter u , $0 \leq u \leq 1$

step 2 : The initial condition $x_1 = 1$, $x_2 = 0$

step 3 : x_1, x_2 two variables

step 4 : The time $(t = 0)$, $(t_f = 1)$ finial time

step 5 : Using gradient Descent method to find optimal solution

Code 15:Solving the Large Scale Variables in Control Optimization Problem2

```
import numpy as np
import matplotlib.pyplot as plt
from gekko import GEKKO
m = GEKKO()
nt = 101
m.time = np.linspace(0,12,nt)
# Parameters
u = m.MV(value=1,ub=1,lb=0)
u.STATUS = 1
u.DCOST = 0
# Variables
x1 = m.Var(value=1)
x2 = m.Var(value=0)
p = np.zeros(nt)
p[-1] = 1.0
final = m.Param(value=p)
# Equations
m.Equation(x1.dt()==u*(10*x2-x1))
m.Equation(x2.dt()==-u*(10*x2-x1)-(1-u)*x2)
# Objective Function
m.Obj(-final*(1-x1-x2))
m.options.IMODE = 6
m.solve()
print('Objective: ' + str(1-x1[-1]-x2[-1]))
plt.figure(1)
plt.subplot(2,1,1)
plt.plot(m.time,x1.value,'k:',lw=2,label=r'$x_1$')
plt.plot(m.time,x2.value,'b-',lw=2,label=r'$x_2$')
plt.ylabel('Value')
plt.legend(loc='best')
plt.subplot(2,1,2)
plt.plot(m.time,u.value,'r-',lw=2,label=r'$u$')
plt.legend(loc='best')
plt.xlabel('Time')
plt.ylabel('Value')
show()
```

Jacobian of the equality constraint with nonzero indices	1897
Jacobian of the inequality constraint with nonzero indices	400
Lagrangian Hessian number of non-zero elements	200
The final number of variables	900
Lower-bounded variables	200
Lower and upper bounded variables	100
Upper-bounded variables	0.0
The number final of equality constraints	600
The number final of inequality constraints	200

Table 4.15: **Numerical Results For the Large Scale Variables in Control Optimization Problem**

final value CPU sec in IPOPT	0.0320
CPU secs final in NLP function evaluation	0.0660
1. The objective function final value	-0.474801689458803
Solution time	0.107200000000375 sec
Objective	-0.474801689458803
Solver	IPOPT

Table 4.16: **Numerical Results For the Large Scale Variables in Control Optimization Problem**

ite	objective	inf_pr	inf_du	lg(mu)	$\|d\|$	alpha_du	alpha_pr
0	0.0000000e+000	9.90e-01	1.00e+00	0.0	0.00e+00	0.00e+00	0.000e+00
1	-1.0757282e-020	2.14e-03	3.22e-01	-6.8	4.31e-01	7.55e-01	1.000e+00
2	-1.1303691e-020	2.37e-07	2.53e-02	-7.4	1.04e-03	9.47e-01	1.000e+00
3	-1.9157949e-010	2.53e-03	3.89e-02	-3.5	2.70e-01	2.86e-01	1.000e+00
4	-5.0159531e-010	1.81e-02	2.21e-02	-3.7	7.08e-01	5.21e-01	1.000e+00
5	-4.8879425e-010	2.38e-02	1.26e-02	-3.6	1.11e+00	7.62e-01	1.000e+00
6	-4.7992153e-010	6.12e-03	3.91e-03	-5.0	3.64e-01	8.80e-01	1.000e+00
7	-4.7494411e-010	2.51e-03	1.10e-03	-6.3	6.33e-01	8.71e-01	1.000e+00
8	-4.7479697e-010	4.05e-04	2.85e-04	-7.2	1.730e-01	9.68e-001	1.000e+00
9	-4.7480051e-010	7.12e-05	5.68e-05	-8.3	1.14e-01	9.69e-01	1.000e+00
10	-4.7480158e-010	3.69e-06	3.73e-06	-11.0	2.84e-02	9.89e-001	1.000e+00
11	-4.7480169e-010	1.40e-08	4.23e-07	-11.0	5.77e-02	8.81e-01	1.000e+00

Table 4.17: **Our Results the Control Optimization Problem Using the Gradient Descent Method**

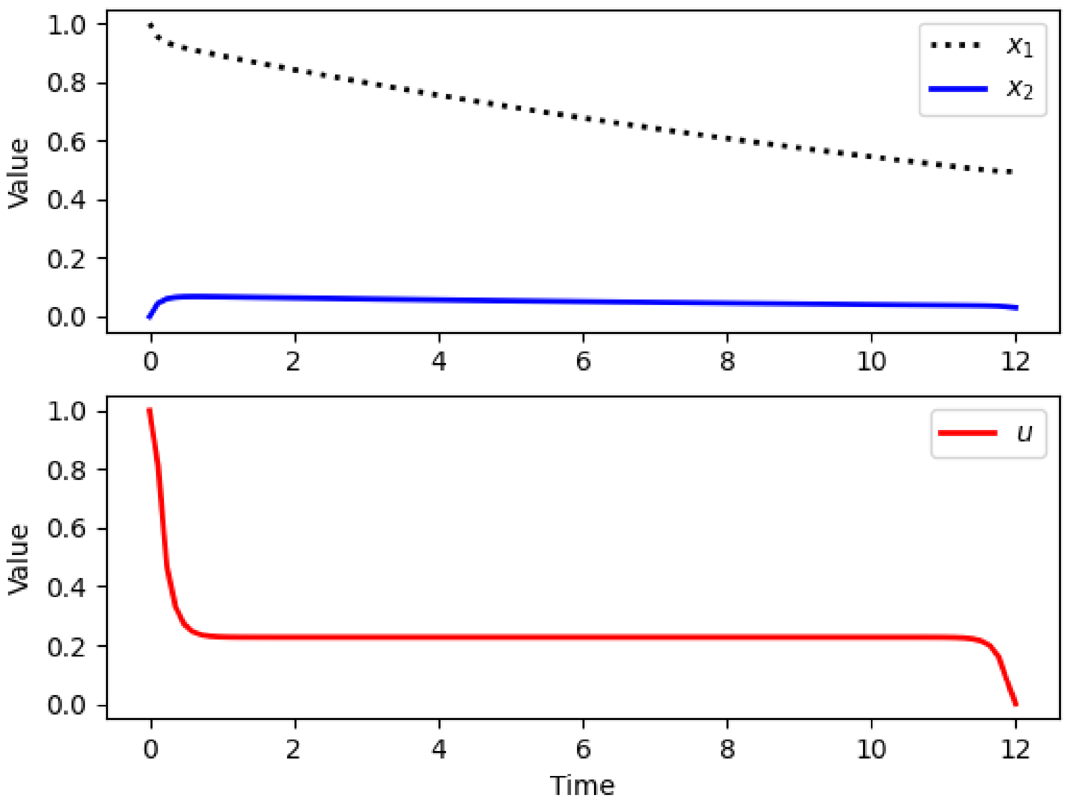


Figure 4.7: The Optimal Solution With Control Optimization Problem Using the Gradient Descent Method

The implementation we obtained based on two variables (x_1, x_2) depends on $u(t)$ changes, which represented the control variables at the time t. As we can see in the figure, the x-axis represents the time and the y-axis represents the value of the objective function relationship.

4.6 Mathematical models applications

The application in this models is in life using Dynamic optimization

▶ **Dynamic of Floating Offshore Wind Turbines**

Floating offshore wind turbines have the potential to harness the vast deepwater wind resource and provide renewable energy to a sizable portion of the global population. Dynamical models were required to determine the viability of various concepts for floating wind turbines by taking into account wind turbine dynamics (including wind flow, aerodynamics, elasticity, and control), incident wave dynamics, sea current dynamics, hydrodynamics, and mooring dynamics of the floating platform. Finance and technology. These dynamic turbines are equipped with the features needed to conduct load analyses for different rotor assemblies, tower, support platform, and mooring system configurations. Excitation of the incident wave from linear diffraction in normal or irregular seas; linear hydrostatic recovery process; nonlinear viscous clouds; linear wave radiation contributes mass and damping, and there are also free-surface memory effects. In response to the interaction between the mooring lines and the sea floor, a semi-fixed mooring line unit has been developed. You can count this as one of the dynamic programs [72].

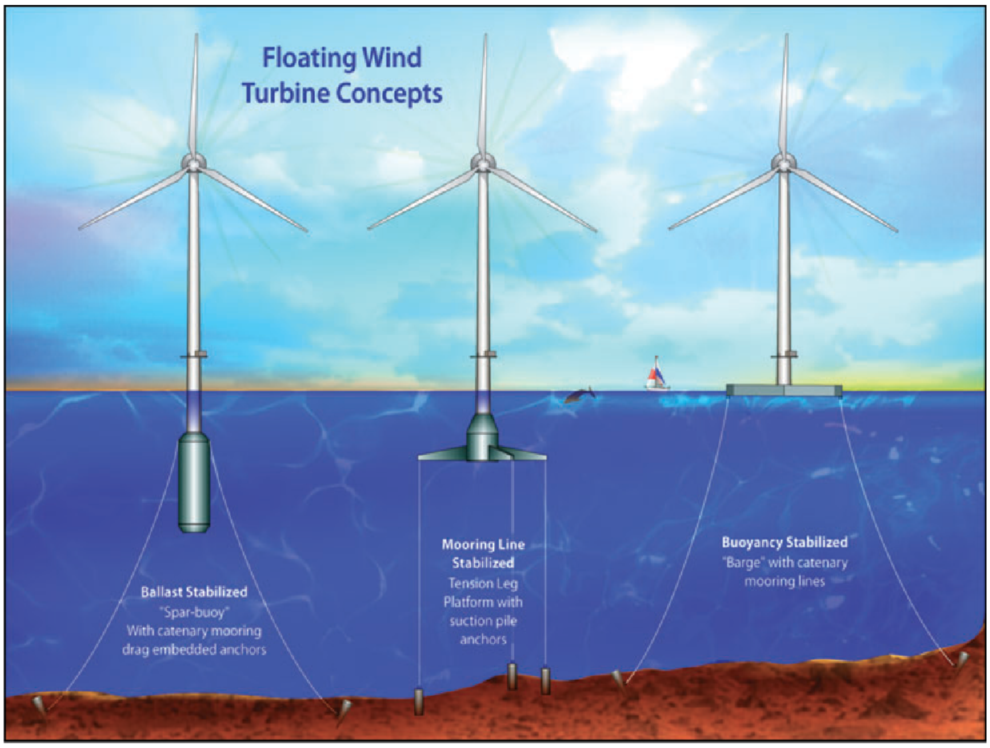

Floating Wind Turbine Concepts
Ballast Stabilized "Spar-buoy" With catenary mooring drag embedded anchors
Mooring Line Stabilized Tension Leg Platform with suction pile anchors
Buoyancy Stabilized "Barge" with catenary mooring lines

Figure 4.8: Dynamic of Floating Offshore Wind Turbines

▶Movement of Cars

The engine in a car provides the power necessary for the car to accelerate and move forward. The dynamic force exerted by the car engine as it moves from one location to another over time is the driving force. Forces that change position over time are called dynamic forces [34].

▶Hitting a Cricket Ball

A cricket ball is hurled at the batsman by the bowler. As the incoming ball collides with the bat, the batsman alters the ball's trajectory and velocity. Dynamic force best describes the pressure a batsman applies to the ball [80].

The application in this model is in life using control optimization

▶COVID-19

The COVID-19 pandemic has started an unprecedented series of international choices. Without any reliable vaccines or drug treatments, governments have resorted to non-drug methods including isolation, quarantine, and lockdowns to slow the spread of the disease. Although these measures are useful in reducing the spread of the virus and buying the healthcare system some time to adjust, they can be very costly economically and socially, and over longer periods the public is less likely to comply. Pharmaceutical interventions, such as vaccination or therapy, have been extensively explored for their potential to shed light on optimal control problems for systems governed by the SIR or the SEIR model. Non-drug interventions have been extensively researched for a control consisting of isolates acting only on infected subjects in the context of optimal control problems. Given that there is an undetected incubation period, non-drug therapies can range from a mild mitigation policy to a strong suppression policy. The suppression effort "seeks to halt the spread of the epidemic by reducing the number of new cases and ensuring that the current low number of cases remains in place indefinitely." Restrictions on travel, closures of schools and businesses, bans on social gatherings, and "shelter in place" orders are all tools of oppression. We use an optimal control approach to demonstrate that a single DOI, that is, a very short period of isolation, is the best course of action.he evolution of the system is governed by the following set of coupled nonlinear ordinary differential equations [24].

►Energy Management

Powerful optimization received a great deal of focus in the area of energy management. From crude oil production to refined product demand and market prices, the oil industry as a whole can benefit from a strategic planning approach. Reducing local cost requires attention to both unit commitment and economic phases of transportation. Within the organisation, production uses most of the energy. The importance of production planning and management increases accordingly. Plan, control and manage all activities involved in creating a product, including those relating to time, space, quantity and operation. The manufacturing planner is responsible for organizing production processes to maximize and optimize the use of resources. As the structure of energy production changes in the near future, there will be a greater need for storage facilities. Limited energy storage is a concern that must be considered in production planning and control. Energy can theoretically be stored in three different ways: electrically, mechanically, and chemically [20].

4.7 Conclusions

The mathematical optimization become one of the most important tools in modern optimal control of computing the control law.In this study,we developed the numerical methods of the solution to linear and nonlinear optimization problems. We have presented different numerical processes to find the good approximate solution depend on the feasible reign of optimal control problem. different Mathematical Models Based on Numerical Analysis,to solve nonlinear optimization problems based on dynamical and control constrained,we present the iterative algorithms which end in finite number of steps for the optimal control problem, on the other hand the iterative methods which converge to the solution in specific class of models.We present the Dynamic optimization model with multivariable based on constraint differential equation The new scheme is an relaxation the algorithms technique, the new scheme applied for nonlinear numerical optimization model with large scale variable and applied to the Dynamic optimization model with initial value problem constraint.The implemented of results depend on different methods, the Golden Section Search method, the Lagrange method, the KKT with inequality constraint method, the Euler method and the Gradient Descent method. The methods was explained, and applied to some different test models to see how they perform. Our implementation of optimization methods using the Python optimization Tool Box with different packages.

4.8 Future Work

Future studies of this project can be explained in the following developments:

1. Formulating a new algorithm and mathematical models that contain large dimensions in numerical optimization.

2. Using different methods to solve one-dimensional and multi-dimensional numerical optimization problems using different programming languages.

3. Different model can be tested, and results can be extracted to provide a better understanding of the algorithm performance.

4. The bounding procedure can be further understood by investigating theoretical convergence properties.

REFERENCES

[1] Alekh Agarwal and Leon Bottou. A lower bound for the optimization of finite sums. In *International conference on machine learning*, pages 78–86. PMLR, 2015.

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

[3] Ahmed Hasan ALRIDHA, Abbas Musleh Salman, and Ahmed Sabah Al-Jilawi. Numerical optimization approach for solving production planning problem using python language. *CENTRAL ASIAN JOURNAL OF MATHEMATICAL THEORY AND COMPUTER SCIENCES*, 3(6):6–15, 2022.

[4] Andreas Antoniou and Wu-Sheng Lu. General nonlinear optimization problems. *Practical Optimization: Algorithms and Engineering Applications*, pages 501–531, 2007.

[5] Andreas Antoniou and Wu-Sheng Lu. *Practical optimization.* Springer, 2007.

[6] Rajesh Kumar Arora. *Optimization: algorithms and applications.* CRC press, 2015.

[7] Kendall Atkinson. *An introduction to numerical analysis.* John wiley & sons, 1991.

[8] JR Bar-On and KA Grasse. Global optimization of a quadratic functional with quadratic equality constraints. *Journal of Optimization Theory and Applications*, 82:379–386, 1994.

[9] Jonathan Baxter and Peter L Bartlett. Direct gradient-based reinforcement learning: I. gradient estimation algorithms. Technical report, Citeseer, 1999.

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

[11] Dimitri P Bertsekas and JN Tsitsiklis. Nonlinear programming. athena scientific optimization and computation series, athena scientific, belmont, ma, 1999.

[12] Joseph-Frédéric Bonnans, Jean Charles Gilbert, Claude Lemaréchal, and Claudia A Sagastizábal. *Numerical optimization: theoretical and practical aspects.* Springer Science & Business Media, 2006.

[13] Cinzia Bonotto. Realistic mathematical modeling and problem posing. *Modeling Students' Mathematical Modeling Competencies: ICTMA 13*, pages 399–408, 2010.

[14] Stephen Boyd, Stephen P Boyd, and Lieven Vandenberghe. *Convex optimization.* Cambridge university press, 2004.

[15] Richard P Brent. *Algorithms for minimization without derivatives.* Courier Corporation, 2013.

[16] Eli Bressert. Scipy and numpy: an overview for developers. 2012.

[17] Arthur Earl Bryson. *Applied optimal control: optimization, estimation and control.* CRC Press, 1975.

[18] Richard L Burden, J Douglas Faires, and Annette M Burden. *Numerical analysis.* Cengage learning, 2015.

[19] John Charles Butcher. *Numerical methods for ordinary differential equations.* John Wiley & Sons, 2016.

[20] Barney L Capehart, Wayne C Turner, and William J Kennedy. *Guide to energy management.* Crc Press, 2006.

[21] Olivier Chapelle and Mingrui Wu. Gradient descent optimization of smoothed information retrieval metrics. *Information retrieval*, 13:216–235, 2010.

[22] Wei-Yu Chiu and Bor-Sen Chen. Mobile location estimation in urban areas using mixed manhattan/euclidean norm and convex optimization. *IEEE Transactions on Wireless Communications*, 8(1):414–423, 2009.

[23] Edwin KP Chong and Stanislaw H Zak. *An introduction to optimization*, volume 75. John Wiley & Sons, 2013.

[24] Marco Ciotti, Massimo Ciccozzi, Alessandro Terrinoni, Wen-Can Jiang, Cheng-Bin Wang, and Sergio Bernardini. The covid-19 pandemic. *Critical reviews in clinical laboratory sciences*, 57(6):365–388, 2020.

[25] Christopher Clinch. *Final Year Project Report: Gekko Application.* PhD thesis, Dublin, National College of Ireland, 2018.

[26] Robert P Cook and Insup Lee. Dymos: A dynamic modification system. In *Proceedings of the Symposium on High-level Debugging*, pages 201–202, 1983.

[27] Mahmoud Mahmoud El-Alem. *A global convergence theory for a class of trust region algorithms for constrained optimization.* Rice University, 1988.

[28] Robert Falck, Justin S Gray, Kaushik Ponnapalli, and Ted Wright. dymos: A python package for optimal control of multidisciplinary systems. *Journal of Open Source Software*, 6(59):2809, 2021.

[29] Z Foroozandeh, M Shamsi, et al. On numerical methods for singular optimal control problems: An application to an auv problem. *Discrete & Continuous Dynamical Systems-Series B*, 24(5), 2019.

[30] Zoltán Füredi and János Komlós. The eigenvalues of random symmetric matrices. *Combinatorica*, 1:233–241, 1981.

[31] Alex Gezerlis. *Numerical methods in physics with Python.* Cambridge University Press, 2020.

[32] Atthakorn Glankwahmdee, Judith S Liebman, and Gary L Hogg. Unconstrained discrete nonlinear programming. *Engineering Optimization*, 4(2):95–107, 1979.

[33] David F Griffiths, Desmond J Higham, David F Griffiths, and Desmond J Higham. Euler's method. *Numerical methods for ordinary differential equations: Initial value problems*, pages 19–31, 2010.

[34] Geoffrey Grime and Ian Shore Jones. Car collisions—the movement of cars and their occupants in accidents. *Proceedings of the Institution of Mechanical Engineers: Automobile Division*, 184(1):87–136, 1969.

[35] Nyoman Gunantara. A review of multi-objective optimization: Methods and its applications. *Cogent Engineering*, 5(1):1502242, 2018.

[36] Richard W Hamming. *Introduction to applied numerical analysis.* Courier Corporation, 2012.

[37] Bryan M Hennelly and John T Sheridan. Generalizing, optimizing, and inventing numerical algorithms for the fractional fourier, fresnel, and linear canonical transforms. *JOSA A*, 22(5):917–927, 2005.

[38] Warren Hoburg and Pieter Abbeel. Geometric programming for aircraft design optimization. *AIAA Journal*, 52(11):2414–2426, 2014.

[39] TE Hull, WH Enright, BM Fellen, and AE Sedgwick. Comparing numerical methods for ordinary differential equations. *SIAM Journal on Numerical Analysis*, 9(4):603–637, 1972.

[40] Alan T James. Zonal polynomials of the real positive definite symmetric matrices. *Annals of Mathematics*, pages 456–469, 1961.

[41] Robert Johansson, Robert Johansson, and Suresh John. *Numerical Python*, volume 1. Springer, 2019.

[42] David Jones and Robert D Grisso. Golden section search as an optimization tool for spreadsheets. *Computers and Electronics in Agriculture*, 7(4):323–335, 1992.

[43] Nocedal Jorge and J Wright Stephen. Numerical optimization, 2006.

[44] Mokbel Karam and Tony Saad. Buckinghampy: A python software for dimensional analysis. *SoftwareX*, 16:100851, 2021.

[45] Jaan Kiusalaas. *Numerical methods in engineering with Python 3*. Cambridge university press, 2013.

[46] Qingkai Kong, Timmy Siauw, and Alexandre Bayen. *Python programming and numerical methods: A guide for engineers and scientists*. Academic Press, 2020.

[47] Vangipuram Lakshmikantham and V Trigiante. *Theory of difference equations numerical methods and applications*. CRC Press, 2002.

[48] Evgeny S Levitin and Boris T Polyak. Constrained minimization methods. *USSR Computational mathematics and mathematical physics*, 6(5):1–50, 1966.

[49] Jian Li, Mingzhi Mao, Frank Uhlig, and Yunong Zhang. Z-type neural-dynamics for time-varying nonlinear optimization under a linear equality constraint with robot application. *Journal of Computational and Applied Mathematics*, 327:155–166, 2018.

[50] OL Mangasarian. A newton method for linear programming. *Journal of Optimization Theory and Applications*, 121:1–18, 2004.

[51] Brendan McMahan and Jacob Abernethy. Minimax optimal algorithms for unconstrained linear optimization. *Advances in Neural Information Processing Systems*, 26, 2013.

[52] Zbigniew Michalewicz. Genetic algorithms, numerical optimization, and constraints. In *Proceedings of the sixth international conference on genetic algorithms*, volume 195, pages 151–158. Citeseer, 1995.

[53] Fabio Nelli and Fabio Nelli. The numpy library. *Python Data Analytics: With Pandas, NumPy, and Matplotlib*, pages 49–85, 2018.

[54] Jorge Nocedal and Stephen J Wright. Quadratic programming. *Numerical optimization*, pages 448–492, 2006.

[55] Mathew M Noel. A new gradient based particle swarm optimization algorithm for accurate computation of global minimum. *Applied Soft Computing*, 12(1):353–359, 2012.

[56] Ruben E Perez, Peter W Jansen, and Joaquim RRA Martins. pyopt: a python-based object-oriented framework for nonlinear constrained optimization. *Structural and Multidisciplinary Optimization*, 45:101–118, 2012.

[57] 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.

[58] Roberto G Proaño. Optimization of the attack angle of the alabe injector of a michell banki turbine in python with gekko package. In *2022 IEEE Sixth Ecuador Technical Chapters Meeting (ETCM)*, pages 1–4. IEEE, 2022.

[59] Arvind U Raghunathan, Devesh K Jha, and Diego Romeres. Pyrobocop: Python-based robotic control & optimization package for manipulation. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 985–991. IEEE, 2022.

[60] 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.

[61] Michael Renardy and Robert C Rogers. *An introduction to partial differential equations*, volume 13. Springer Science & Business Media, 2006.

[62] Brian D Ripley. The second-order analysis of stationary point processes. *Journal of applied probability*, 13(2):255–266, 1976.

[63] Harrison Ritz, Xiamin Leng, and Amitai Shenhav. Cognitive control as a multivariate optimization problem. *Journal of Cognitive Neuroscience*, 34(4):569–591, 2022.

[64] Steven Roman, S Axler, and FW Gehring. *Advanced linear algebra*, volume 3. Springer, 2005.

[65] Sebastian Ruder. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016.

[66] Michel F Sanner et al. Python: a programming language for software integration and development. *J Mol Graph Model*, 17(1):57–61, 1999.

[67] Gilles Savard and Jacques Gauvin. The steepest descent direction for the nonlinear bilevel programming problem. *Operations Research Letters*, 15(5):265–272, 1994.

[68] Marc Schoenauer and Spyros Xanthakis. Constrained ga optimization. In *Proc. 5th International Conference on Genetic Algorithms*, pages 573–580. Morgan Kaufmann, 1993.

[69] Naum Zuselevich Shor. *Minimization methods for non-differentiable functions*, volume 3. Springer Science & Business Media, 2012.

[70] Ankur Sinha, Tharo Soun, and Kalyanmoy Deb. Using karush-kuhn-tucker proximity measure for solving bilevel optimization problems. *Swarm and evolutionary computation*, 44:496–510, 2019.

[71] Abel Soares Siqueira, Raniere Costa da Silva, and Luiz-Rafael Santos. Perprof-py: A python package for performance profile of mathematical optimization software. *Journal of Open Research Software*, 4(1), 2016.

[72] Bjørn Skaare, Tor David Hanson, Finn Gunnar Nielsen, Rune Yttervik, Anders Melchior Hansen, Kenneth Thomsen, and Torben Juul Larsen. Integrated dynamic analysis of floating offshore wind turbines. In *European wind energy conference and exhibition*, volume 3, pages 1929–1939. Hamburg, Germany, 2007.

[73] Aivar Sootla, Anders Rantzer, and Georgios Kotsalis. Multivariable optimization-based model reduction. *IEEE Transactions on Automatic Control*, 54(10):2477–2480, 2009.

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

[75] Jos MF Ten Berge. *Least squares optimization in multivariate analysis.* DSWO Press, Leiden University Leiden, 1993.

[76] Johan Thim. *Continuous nowhere differentiable functions.* 2003.

[77] I Thng, A Cantoni, and YH Leung. Analytical solutions to the optimization of a quadratic cost function subject to linear and quadratic equality constraints. *Applied Mathematics and Optimization*, 34:161–182, 1996.

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

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

[80] Juanita R Weissensteiner, Bruce Abernethy, and Damian Farrow. Hitting a cricket ball: what components of the interceptive action are most linked to expertise? *Sports Biomechanics*, 10(4):324–338, 2011.

[81] Angelika Wiegele. Biq mac library—a collection of max-cut and quadratic 0-1 programming instances of medium size. *Preprint*, 51, 2007.

[82] Shmuel Winograd. A new algorithm for inner product. *IEEE Transactions on Computers*, 100(7):693–694, 1968.

[83] Stephen Wright, Jorge Nocedal, et al. Numerical optimization. *Springer Science*, 35(67-68):7, 1999.

[84] J SunL W ZhangY Wu. Properties of the augmented lagrangian in nonlinear semidefinite optimization. *Journal of optimization theory and applications*, 129(3):437, 2006.

[85] Dachuan Xu*, Yinyu Ye, and Jiawei Zhang. Approximating the 2-catalog segmentation problem using semidefinite programming relaxations. *Optimization Methods and Software*, 18(6):705–719, 2003.

[86] X Yang. Introduction to mathematical optimization. *From linear programming to metaheuristics*, 2008.

[87] Sanyou Y Zeng, Lishan S Kang, and Lixin X Ding. An orthogonal multi-objective evolutionary algorithm for multi-objective optimization problems with constraints. *Evolutionary computation*, 12(1):77–98, 2004.

[88] N Zhang, D Lei, and JF Zhao. An improved adagrad gradient descent optimization algorithm. In *2018 Chinese Automation Congress (CAC)*, pages 2359–2362. IEEE, 2018.

الملخص

تعتبر تقنيات التحسين العددي أداة مهمة يمكن أن تحدد الحل الامثل. تتمثل المهمة الرئيسية لهذه الدراسة في تطوير خوارزميات جديدة للجانبين (التحسين غير الخطي المقيد وغير المقيد). دراسة هذا البحث التفاعل بين النماذج المختلفة بناءً على التحكم والتحسين الديناميكي. تتطلب مشاكل التحسين العددي المختلفة فئات مختلفة من خوارزميات التحسين الرقمي لحل فعال. تركز الدراسة على نماذج التحسين ذات البعد الواحد ونماذج التحسين متعددة الأبعاد مع تطبيقاتها وتحسين خصائص التقارب النظري, ويستخدم(APMintor solver Python) للحساب العددي، وهو برنامج التحسين المجاني.الدراسة طورت فئات مختلفة من النمذجة لمشاكل التحسين التفاضلي التي تعتمد على النظم الديناميكية والتحكم , ذات المشاكل البعد الواحد والابعاد كثيرة المتغيرات .

جمهورية العراق

وزارة التعليم العالي والبحث العلمي

جامعة بابل

كلية التربية للعلوم الصرفة

قسم الرياضيات

نهج جديد لمشاكل التحسين على الوجهين في التحليل العددي

أطروحـة

مقدمة الى مجلس كلية التربية للعلوم الصرفة في جامعة بابل كجزء من متطلبات نيل

درجة الدكتوراه فلسفة في التربية / رياضيات

من قبل الطالب

عمار عماد ناظم نومي الجنابي

بأشراف

أ . م .د احمد صباح احمد الجيلاوي

1444 هـ

2023 م