Republic of Iraq

Ministry of Higher Education & Scientific Research

University of Babylon

College of Education for Pure Sciences

Department of Mathematics

A New Approach of Numerical Optimization with Differential Equations

A Thesis

Submitted to the Council of the College of Education for Pure Sciences,

University of Babylon in Partial Fulfillment of the Requirements for

the Degree of Master in Education / Mathematics.

By

Rabab Abdul Sattar Mohsin Jawad

Supervised by

Assist prof. Dr. Ahmed Sabah Al-Jilawi

2022 A.D. **1444 A.H.**

بِسۡــــمِ ٱللَّهِ ٱلرَّحۡمَٰنِ ٱلرَّحِيــــمِ

وَيَرَى الَّذِينَ أُوتُوا الْعِلْمَ الَّذِي أُنْزِلَ إِلَيْكَ مِنْ رَبِّكَ هُوَ الْحَقَّ وَيَهْدِي إِلى صِرَاطِ الْعَزِيزِ الْحَمِيدِ

صَدَقَ ٱللَّهُ ٱلْعَظِيمُ

سورة سبأ- الآية 6

The Supervisor Certificate

I certify that this thesis entitled " A New Approach of Numerical Optimization with Differential Equations " was prepared by the student " Rabab Abdul Sattar Mohsin Jawad " under my supervision at the University of Babylon, College of Education for Pure Sciences as a partial fulfillment of the requirement the Degree of Master in Education /Mathematics.

Signature: [illegible]

Name: Dr. Ahmed Sabah Al-Jilawi

Scientific Grade: Assistant Professor

Date: / / 2022

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

Signature: [illegible]

Name: Dr. Azal Jaafar Musa

Scientific Grade: Assistant Professor

Address: Head of Mathematics Department

Date: / / 2022

Linguistic Supervisor's Certification

This is to certify that I have read this thesis entitled " A New Approach of Numerical Optimization with Differential Equations" and I found it is qualified for debate.

Signature: [illegible]

Name: Dr. Lihadh Abdul Ameer Kareem Mubarak

Title: Assistant Professor

Address: English Department, University of Babylon

Date: / / 2022

Scientific Supervisor's Certification

This is to certify that I have read this thesis entitled " A New Approach of Numerical Optimization with Differential Equations" and I found it is qualified for debate.

Signature: [illegible]

Name: Dr. Mushtaq K.Abd Al-Rahem

Title: Assistant Professor

Address: College of Administration and Economics, University of Karbala

Date: / / 2022

Scientific Supervisor’s Certification

This is to certify that I have read this thesis entitled " A New Approach of Numerical Optimization with Differential Equations" and I found it is qualified for debate.

Signature: [illegible]

Name: Dr. Bassem Abbas Hassan

Title: Professor

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

Date: / / 2022

Examining Committee Certificate

We certify that we have read this thesis entitled " A New Approach of Numerical Optimization with Differential Equations " as an examination committee, examined the student " Rabab Abdul Sattar Mohsin Jawad " in its contents and that in our opinion it meets the standard of a dissertation for the Degree of Master in Education/ Mathematics.

Signature:	Signature:
Name: Dr.Ahmed Abbas Al-Adilee	Name: Dr.Hussien Ali Mohammed Al-Sharifi
Title: Professor	Title: Assist. Professor
Date : / / 2022	Date : / / 2022
Chairman	Member
Signature:	Signature:
Name: Mustafa Hasan Hadi	Name: Dr.Ahmed Sabah Al-Jilawi
Title: Assist. Professor	Title: Assist. Professor
Date : / / 2022	Date : / / 2022
Member	Member / Supervisor

Approved by the Dean of College

Signature:

Name: Dr. Bahaa Hussein Salih Rabee

Title: Professor

Address: Dean of the College of Education for Pure Sciences

Date : / / 2022

Dedication

To those who have lived in hearts .... And who dwelt in heaven

To those who sacrificed their lives .... To live safely

For every national martyr .... Previous and subsequent ones

And to the soul of my beloved father .... May he finally see the light

Mohammed ... The God of the Immaculate

May Allah's prayers and peace be upon them all

I raise this modest product ...

Rabab A.S. AlJawad

Acknowledgements

In the name of Allah, the Most Gracious, the Most Merciful First and above all, Praise be to Allah who gave me the health and patience to conduct this work. And then, I would like to thank prophet Muhammad and Ahl Al-Bayt (peace be upon them). I am especially grateful for the help provided by my wonderful and respected supervisor with high morals, Dr. Ahmed Sabah Al-jilawi, to encourage, guide and support him during this work. Deep appreciation and love to all my family and everyone who gave me a hand of support.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

Table 1: Notations used in the Thesis

$f(\chi)$ and $\chi(t_f)$	The Objective function
$Minf(\chi)$	The Minimum of Objective Function
$Maxf(\chi)$	The Maximum of Objective Function
$g(\chi)$	The Inequality Constraint
$h(\chi)$	The Equality Constraint
QP	Quadratic Programming
SQP	Sequential Quadratic Programming
LP	Linear Programming or (Problem)
NLP	Non-Linear Programming (problem)
DE	Differential Equation
ODE	Ordinary Differential Equation
PDE	Partial Differential Equation
du, df and $y^{'}$	First Order Derivatives
d^2u, d^2f and $y^{''}$	Second Order Derivative
∂u	Partial Differential
IVP	Initial Value Problem
BVP	Boundary Value Problem
min	Minimum
max	Maximum
ISP	Internet Service Provider
∇f	Gradient the function
$\nabla^2 f$	Hessian the function

$F(\chi)$	Integral function
C^1	One-time derivable
C^2	Two-time derivable
MPC	Model Predicative Control
RTO	Real Time Optimization
CVP	Control Vector Parameter
IDP	Iterative Dynamic Programming
DOP	Dynamic Optimization Problem
DAE	Dynamic Algebra Equation
KKT	Karush Kuhn Tucker
MILP	Mixed Integer Linear programming
MINLP	Mixed Integer Non-Linear programming
s.t	subject to

Abstract

In this study, the problems of numerical analysis and optimization in the category of functional differential equations, which are important tools in applied mathematics, were presented. Where the mathematical model SIR (S = Susceptible, I = Infection, R = Recovered or Removed) used with COVID-19 was developed as an Objective function with constraints imposed in Iraq according to data taken from the Karbala health department for 2021, as well as the Euler method was used to find the results from the speed of the spread of the epidemic to the speed of recovery from it. A new numerical method was introduced based on new approaches to optimization algorithms, general dynamic optimization problems, optimal control, where both the algorithm of the interior point method and the gray wolves optimization algorithm (GWO) were used to optimize systems in chemical processes adopted. The main programming language used in this study was Python, where its observed the results obtained in general accurate, robust and achieved the desired goals to be reached compared to the results of previous studies

CHAPTER 1

INTRODUCTION

Introduction

Optimization has evolved into a flexible technique with applications spanning industry and information technology to the social sciences. Methods for addressing optimization issues are many. There are so many different approaches that it is impossible to fully use them. They are specified in several technical languages and implemented in various software programs. Many are never implemented. It is difficult to determine which one is best for a specific situation, and there are far too few opportunities to mix strategies with complementary qualities. The ideal situation would be to combine various strategies under one roof so that they and their combinations could all be used to address an issue. As it turns out, many of them have a similar problem solving style on some level [52]. The Optimization as a mathematical topic has the following distinguishing characteristics:

1. In the past, most mathematics was dedicated to defining how systems behaved, such as in the laws of physics. Today's mathematics is committed to prescribing how systems should act. The introduction of computers, on the other hand, has placed a greater focus on the use of mathematics to cause systems to act in certain ways, which in turn leads to concerns of optimization.

2. Inequalities predominate over equations: Because optimization often deals with variables that must be within specified ranges, imposed by the restrictions of what options are permissible, inequalities predominate over equations in most cases.

3. Linear and nonlinear vs convex and non convex: In optimization, the distinction between linearity and non linearity is less relevant than the distinction between convexity and non convexity, which requires a new investment in conceptual understanding [29].

4. Differential calculus versus sub differential calculus: The prevalence of inequalities,

as well as the special properties of the operations "max" and "min," necessitate the development of a methodology that does not rely as heavily as classical mathematics on supposing surfaces to be smooth and functions upon to be differentiable everywhere, as does classical mathematics [78].

Emphasis is placed on the fundamental problems of constrained and unconstrained optimization, linear and convex programming, fundamental iterative solution algorithms, gradient methods, Newton-Raphson and its variants, and more general iterative optimization methods, as well as on the necessary mathematical tools and results. One reason optimization is valuable in applied mathematics is because nature often optimizes, or maybe a more accurate term is that nature economizes [68]. The patterns and sizes of tree branches generate effective communication networks; the hexagonal structures of honeycombs efficiently occupy space; the shape of a soap bubble reduces the potential energy contained in the surface tension. Optimization is the process of optimizing a function of one or more often multiple variables. The Objective function is the function that has to be optimized. There are two main sorts of applications that result in optimization issues, which will refer to as optimization problems and inference problems. On the one hand, there are optimization issues, which may also refer to as natural optimization problems, in which maximizing a given function is the primary and natural aim. The primary objective, maximization of profits or shortest commute, is unquestionable, albeit the particular function involved will depend on the simplifications used to convert the real-world situation to mathematics. A factory wanting to maximize earnings, subject to whatever constraints the circumstances imposes, or a commuter seeking to reduce the time required to get to work, subject to speed limitations, are examples of such situations. The manufacturer may or may not neglect non linearity such as economies of scale when converting the real-world issue to a mathematical problem. The commuter may or may not use probabilistic models of traffic density when turning the real-world problem into a mathematical problem. While the final mathematical optimization problem will rely on these choices, the underlying real-world issue is still one of optimization [8, 24].

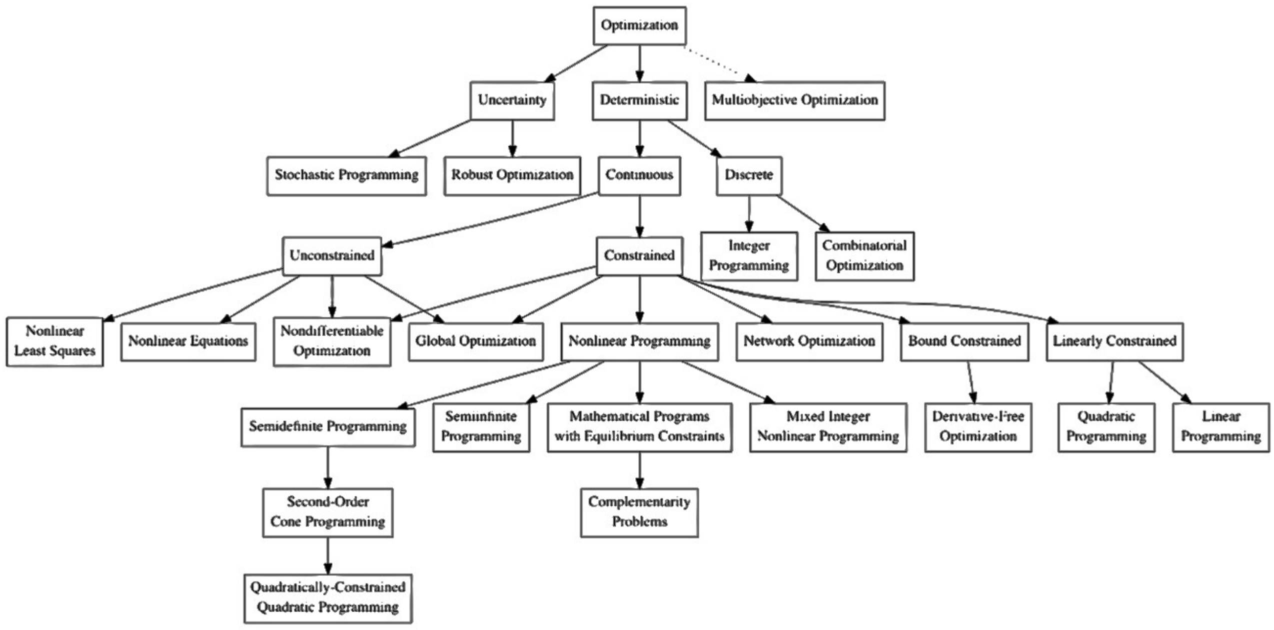


Figure 1.1: Classification of Optimization

1.1 Related Work

1. Numerical Optimization
2. Differential Equations
3. Topology Optimization
4. Dynamic Optimization and Optimal Control
5. Optimization Algorithms

1.1.1 Numerical Optimization

People optimize. Airlines arrange employees and planes to save costs. Investors aim to construct portfolios that minimize risk while maximizing profit. Manufacturers strive to maximize the efficiency of their manufacturing processes via design and operation. Nature optimize. Physical systems naturally gravitate toward a condition of minimal energy. In an isolated chemical system, molecules react with one another until the total potential energy of their electrons is reduced to a minimum. Light rays choose the shortest possible route. Optimization is a critical technique in decision science and physical system analysis [100]. To apply it, first establish a goal, a quantifiable measure of the studied system's performance. Profit, time, potential energy, or any amount or combination of quantities that a single number can express may be used as the target. The purpose is contingent upon some system features, referred to as variables or unknowns. Our purpose is to determine the variable values that maximize the objective. Frequently, the variables are confined or restricted in some manner. For example, a molecule's electron density and the interest rate on loan cannot be negative [68]. Modelling is the process of defining the aim, variables, and constraints for a specific situation. The optimization process begins with developing an adequate model, which is often the most critical phase. If the model is oversimplified, it will not provide valuable insights into the real situation, but the problem may become unsolvable if it is oversimplified. Once the model has been defined, it may be solved using an optimization procedure. Typically, the method and model are complex enough that this operation requires the use of a computer. There is no general algorithm for optimization. Rather than that, there are several methods, each one optimized for a certain sort of optimization issue. It is often up to the user to choose an algorithm suited for their particular application [38]. This is a critical option; it may decide whether the issue is solved quickly or slowly, or even if the problem is addressed at all. After applying an optimization technique to the model, which is able to determine if the approach was successful in finding a solution [68].

1.1.2 Differential Equations

Differential equations may be used to model practically any system that is changing. They pervade all fields of study, including science and engineering, economics, social science, biology, business, and health care. Several mathematicians have spent hundreds of years studying the nature of these equations, and there are numerous well-developed solution approaches [50]. Often, the systems represented by differential equations are so complicated, or the systems described by them are so massive that a purely analytical solution to the equations is impractical . Computer simulations and numerical approaches are advantageous in these complicated systems. Prior to the advent of programmable computers, numerical approximation methods for solving differential equations were developed [53] . Throughout World War II, rooms filled with workers (typically women) working on mechanical calculators to solve systems of differential equations for military calculations numerically were ubiquitous. Prior to the advent of programmable computers, analogue computers were often used to investigate mechanical, thermal, or chemical systems. As the speed and affordability of programmable computers have grown, more complicated systems of differential equations may be solved using simple programs developed to run on a standard personal computer. Today, the computer on your desk is capable of tackling issues that were previously unreachable to even the best supercomputers just five or ten years ago [51].

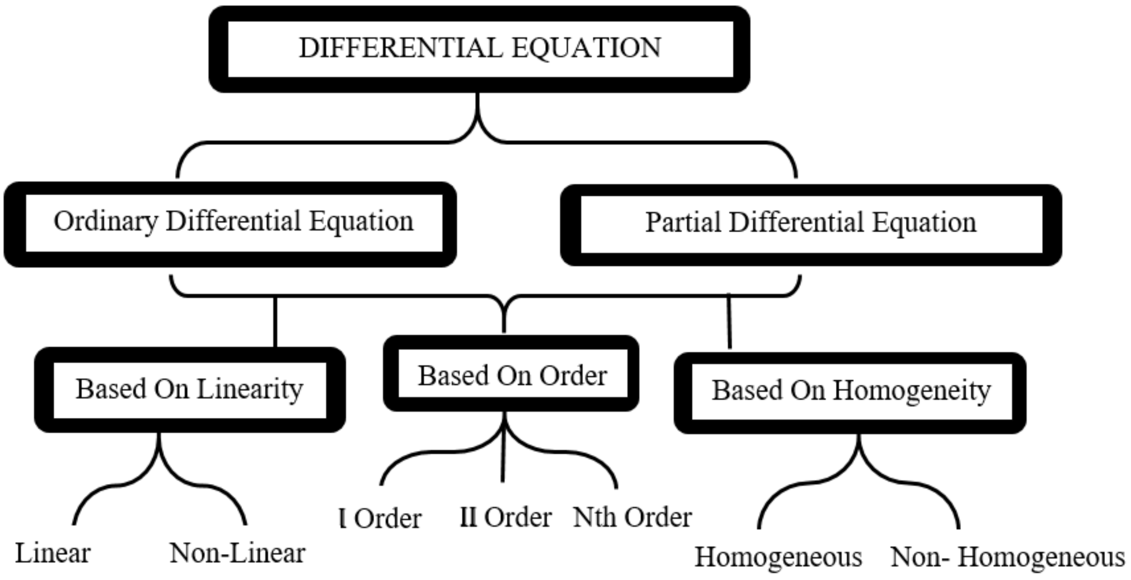


Figure 1.2: Classification of Differential Equations

1.1.3 Topology Optimization

Since the start of the homogenization approach, topology optimization has become a well-known research topic (Bendsoe and Kikuchi 1988). Researchers have used the method to study steady-state heat conduction. For example, Bendsoe and Sigmund (2004) use a two-dimensional (2-D) finite element framework to get a branching, "tree-like" structure. Gersborg-Hansen et al. (2006) solved a similar planar heat conduction problem using the finite volume method and topology optimization. They found that topology optimization for heat conduction led to designs that were similar to branching, "five-finger" plans for optimal traffic patterns, where constant population density was like constant volumetric heating in a domain. As de Kruijf et al. showed, similar branching or dendritic characteristics still show up in 2-D structural design problems with multiple goals when heat conduction is given more weight (2007). The problem of planar heat conduction can also be solved by combining the level set method with topological derivatives, as Zhuang et al. (2007) did when they looked at multiple load cases. As Li et al. explain, you could also use evolutionary algorithms to design the 2-D topology of heat-conducting fields (2004). Extensions to three-dimensional (3-D) design for heat conduction (Dede 2009; Chen et al. 2010; Burger FH et al. 2013) show that the size of the solution space does not limit tree-like properties. Not only have problems with isolated heat conduction been solved, but also those with convective boundary conditions (Bruns 2007; Iga et al. 2009; Yoon 2010; Dede et al. 2015). The problem of both conduction and convection at the same time also favors tree-like structures. So, topology optimization has been used to solve the steady-state heat conduction problem using a variety of computer algorithms, and most studies agree that dendritic topologies are a class of designs that work best. Even though the reviewed methods tend to make similar designs, it is well known that the efficiency and accuracy of structural topology optimization methods can depend a lot on the assumed starting material distribution, analysis mesh, and filtering techniques that are used (Svanberg and Svard 2013; Sigmund and Maute 2013) [58].

The selected method for dynamic scaling was Topology Optimization (TO). TO seeks to identify the optimal material arrangement inside a design domain in order to minimize/maximize an objective under a set of constraints and boundary conditions. This approach is based on iterative analysis and design update processes, which are frequently led by gradient computation. Since 1988, it has been utilized and developed within the engineering area. The design domain is discretized into finite elements, and the optimizer calculates which elements should be filled with material and which should be empty. Topology optimization has numerical instabilities, such as checkerboard patterns and mesh dependencies, despite its widespread application in structural optimization. In the field of TO, one of the most commonly used method is inferior point line search filter method which will be the chosen approach in this thesis [103]

A material indicator function, $\sigma : \Omega \subseteq R^d \to \{0, 1\}$, determines whether there is material present in the design domain ($\sigma = 1$) or not ($\sigma = 0$) in material distribution topology optimization (Bendsøe and Sigmund 2003). The domain (Ω) is often divided into n

components so that the topology optimization issue may be solved numerically. Determine if a particular element includes material or not by using the element values $\sigma_i \in \{0,1\}, i \in \{1,\ldots,n\}$ that are produced as a result of the optimization. Typically, the resultant nonlinear integer optimization problem is softened by letting $\sigma_i \in [0,1], i \in \{1,\ldots,n\}$. This relaxation permits the application of gradient-based optimization methods that are suited for tackling issues on a large scale. In the continuous context, such relaxation is frequently required to ensure the presence of solutions. We employ penalization strategies to press for binary designs. Penalization, on the other hand, usually has the effect of making the solutions to the optimization issue mesh-dependent. It is well knowledge that mesh dependence can at times be traced back to the absence of viable solutions to the continuous problem that it corresponds to. Borrvall (2001) gives a thorough analysis of numerous popular ways to overcome the issue of mesh-dependence and presence of solutions. Several strategies have been offered to handle the issue of mesh-dependence and existence of solutions. Utilizing a filtering method is one of the most common approaches utilized to accomplish the goal of creating mesh-independent designs. It is usual practice to classify filtering processes as either sensitivity filtering (Sigmund 1994) or density filtering (Bourdin 2001, Bruns and Tortorelli 2001), where the objective function's derivatives are filtered or where the design variables are filtered The design variables in a density filtering technique are no longer the "physical density," that is, the coefficients that enter the governing equation. Bourdin (2001) showed the existence of solutions to a continuous variant of the linearly filtered penalized minimal compliance issue. The linear filter's main disadvantage is that it compensates for penalization by providing physical designs with relatively extensive zones of intermediate density. Recently, a set of nonlinear filters aiming at lowering intermediate densities while maintaining mesh independence has been proposed (Guest et al. 2004, 2011, Sigmund 2007, Svanberg and Svard 2013). We just developed the class the line search methods to gain a new approach to harmonize the use of filters [45]. To identify the best solution for the objective function inside the code of the interior point method that incorporates the majority of topology optimization filters and the code utilizing the ipopt solver, the method and codes can be explained in the fourth and fifth chapters of this thesis.

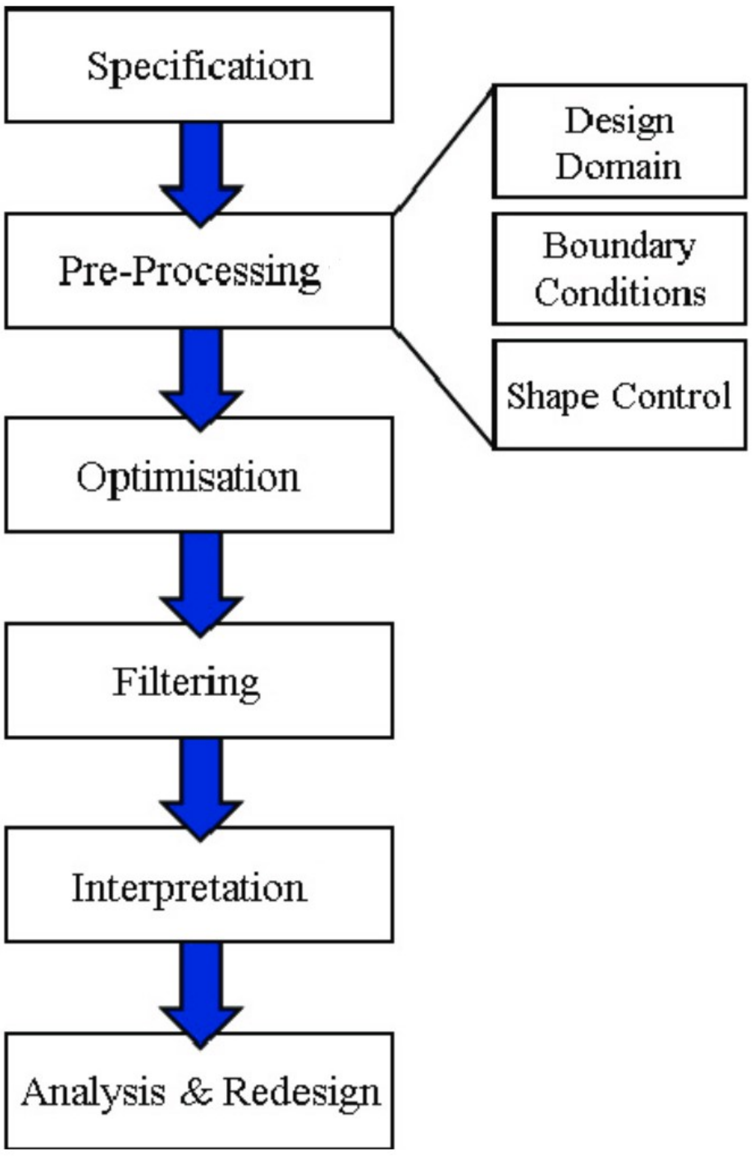


Figure 1.3: Workflow for Topology Optimization

1.1.4 Dynamic Optimization and Optimal Control

Optimal control theory is a part of mathematical optimization that deals with creating a control for a dynamical system over time that optimizes an objective function. Numerous applications exist in the fields of science, engineering, and operations research. For instance, the dynamical system may be a spaceship with controls that correspond to rocket thrusters, and the main objective could be to reach the moon with the minimum amount of fuel [88]. Alternatively, the dynamical system may be a country's economy, with the goal of minimizing unemployment; in this case, the controls could be fiscal and monetary policy [36]. Optimal control is a mathematical optimization approach for formulating control strategies that is an extension of calculus of variations [88]. The method is largely due to the work of Lev Pontryagin and Richard Bellman in the 1950s, after contributions to calculus of variations by Edward J. McShane. Optimal control can be seen as a control strategy in control theory [41]. Optimal control is concerned with the challenge of determining a control law for a given system that meets a specific optimality condition. A cost functional that is a function of state and control variables is included in a control issue. A set of differential equations defining the pathways of the control variables that minimize the cost function is referred to as an optimum control. Pontryagin's maximum principle (a necessary condition also known as Pontryagin's minimal principle or simply Pontryagin's principle) can be used to locate the optimal control [19]. otherwise, by finding a solution to the Hamilton–Jacobi–Bellman equation (a sufficient condition) by way of illustration Consider a vehicle going in a straight path on a road with hills. How should the driver depress the accelerator pedal to save trip time? In this instance, control law relates directly to the manner in which the driver applies the accelerator and switches gears. The system consists of both the automobile and the road, and the optimality criteria is the minimization of total journey time. Control issues typically include auxiliary restrictions. For instance, the amount of accessible gasoline may be limited, the accelerator pedal may not be depressed all the way to the bottom of the vehicle, and there may be speed restrictions. A suitable cost function is a mathematical equation that describes the journey time as a function of speed, geometrical factors, and beginning system circumstances. Frequently, constraints are interchangeable with the cost function. Another related optimal control problem could be to figure out how to drive the car so that it uses the least amount of gas, given that it has to finish a given course in a certain amount of time. Another related control problem could be to make the trip as cheap as possible in terms of money, given the prices for time and fuel [36]. A more abstract framework goes as follows. Minimize the continuous-time cost functional:

$$J\left[\chi(\cdot), u(\cdot), t_0, t_f\right]$$

Subject to the first-order dynamic constraints (the state equation)

$$\dot{\chi}(t) = \mathrm{f}[\chi(t), \mathbf{u}(t), t]$$

The algebraic path constraints

$$\mathrm{h}[\chi(t), \mathrm{u}(t), t] \leq 0$$

Where $\chi(t)$ is the state, $\mathrm{u}(t)$ is the control, t is the independent variable (generally speaking, time), t_0 is the initial time, and t_f is the terminal time. Furthermore, it is noted that the path constraints are in general inequality constraints and thus may not be active (i.e., equal to zero) at the optimal solution. It is also noted that the optimal control problem as stated above may have multiple solutions (i.e., the solution may not be unique). Thus, it is most often the case that any solution $\left[\chi^*(t), \mathrm{u}^*(t), t_0^*, t_f^*\right]$ to the optimal control problem is locally minimizing. With the escalation of environmental and energy challenges, industry and academics have turned their attention to lowering energy consumption, enhancing production efficiency, and optimizing economic gains in the chemical process [57]. The chemical production model, on the other hand, has a complicated structure, a high degree of nonlinearity, and unpredictability [85]. classified chemical process optimization control as a kind of dynamic optimization, indicating that dynamic optimization technology has developed into an effective tool for resolving chemical process control challenges. Control Vector Parameterization (CVP) [75] and Iterative Dynamic Programming (IDP) [60] , [54] are two established methodologies for solving chemical dynamic optimization issues.

Numerous real-world optimization problems have dynamic search spaces due to the objective function, the number of variables, and the presence of constraints [66]. Optimizing problems that change over time and must be handled online using an optimization approach are called dynamic optimization problems (DOPs) [67]. It is not enough for algorithms to discover excellent solutions when solving DOPs; they must also be capable of adapting to changing conditions to swiftly find a new key when the prior one is no longer good enough [66, 67, 99]:

1.1.5 Optimization Algorithms

The topic of whether a problem can be solved efficiently, which is comparable to the present concept of an algorithm, garnered much attention in the early twentieth century, particularly during the thirties. During that time period, the focus was on effectively classifying issues as solvable or unsolvable. To this end, the need emerged for a computational model capable of classifying a problem as solvable if an algorithm to solve it can be constructed using that model. Several of these models include Goedel's recursive functions, Church's -calculus, Post's Post machines, and Turing's Turing machines. As a theoretical complement to current computer equipment, the RAM model of computation was presented. According to Church Thesis, all of these models are equivalent in that if a problem can be solved on one of them, it can be solved on all of them. Surprisingly, "nearly all" issues turn out to be intractable [74]. This is readily defended as follows. The collection of functions to calculate is uncountable since each method may be seen as a function having a domain of non-negative integers and a range of real values. Due to the fact that each method, or more precisely, each program, may be encoded as a binary string corresponding to a unique positive integer, the number of functions that can be calculated is countable [15]. Thus, the countable set of integers is identical to the number of solved issues, but the number of insoluble issues is equal to the countable set of real numbers (which is uncountable). To demonstrate, there is no way for detecting if the decimal expansion has seven consecutive. This is consistent with the algorithm definition, which requires that the time of an algorithm's execution be limited. Another example is determining whether an equation with n variables $\chi_1, \chi_2, ..., \chi_n$ has integer solutions. This problem is unsolvable regardless of the amount of computing power used. The subject of computability theory or theory of computing is concerned with the decidability and solvability of problems; nevertheless, some computer scientists contend that the present area of algorithms should be included in this study. The advent of digital computers forced a review of those previously resolved concerns. At first, without regard for the program's resource needs, most notably time, one was pleased with a basic program that handled a specific issue. Then, as a consequence of scarce resources and the increasing demand for complex algorithms, the necessity for resource-efficient programs became obvious. As a result, a new computer science called computational complexity was formed [7]. The following is a formal definition of the sorting problem: A series of n integers $(a1, a2, ..., an)$ as input.

Output: The input sequence $(a1, a2, ..., an)$ is reordered in such a way that $a1,' a2,' an'$. Sorting is a basic operation in computer science since it is used by so many applications as an intermediate step. As a consequence, now have access to a huge number of effective sorting algorithms. Which algorithm is optimal for a given application is determined by a number of factors, comprising the number of things to sort, the degree to which the items are already sorted, any limits on the item values, the computer's architecture, and the kind of storage devices to be used: main memory, disks, or even tapes. Accuracy is defined as the ability of an algorithm to end with the correct output for each input instance. An appropriate algorithm is one that resolves the computer problem at hand. In certain input conditions, a faulty algorithm may fail to terminate at all or may terminate with an incorrect answer. Contrary to common assumptions, flawed algorithms may sometimes be advantageous if their error rate is kept under control. Normally, though, will only work on algorithms that are correct. Only one thing is needed, the specification must accurately describe the computing method that will be used [28]. Optimization employs iterative processes. They begin with an educated approximation of the ideal variable values and iteratively revise their estimates until they get a solution. What distinguishes one algorithm from another is the approach used to transition from one iteration to the next. In the majority of methods, the values of the objective function, the constraints, and perhaps their first and second derivatives are employed. Certain algorithms employ data from previous iterations, while others utilize just data from the current iteration. Regardless of these details, all superior algorithms should possess the following characteristics [10]:

- **Sturdiness:** Perform wonderfully on a wide variety of tasks in their class, assuming that any relevant starting variables are selected.
- **Efficient operation:** Refrain from using an excessive quantity of computer time or storage space.
- **Exactitude:** Capable of accurately identifying a solution without being too sensitive to data imperfections or arithmetic rounding mistakes induced by computer implementations. These aims may conflict. For instance, a rapid convergent solution for nonlinear programming may need an excessive amount of computer storage for large problems. On the other side, the most time-consuming technique may be the most resilient. Numerical optimization is fundamentally about making trade-offs between convergence rate and storage requirements, as well as between robustness and speed. The mathematical theory of optimization is utilized to define optimal locations and serves as the basis for the bulk of

algorithms. Without a firm grasp of the underlying theory, it is hard to comprehend numerical optimization completely [32].

Different Algorithms may differ in their efficiency and resource needs. Newton's gradient-based approach, for example, is very efficient at solving smooth goal functions but may get stuck in local modes if the objective function is extremely multimodal. If the target is not smooth or has a kink, the Nelder–Mead simplex approach may be employed since it is a gradient-free technique that works well in discontinuous situations, albeit it may become sluggish and stuck in a local mode. Although nonlinear optimization strategies come in a variety of flavours, including the trust-region method and the interior-point method, all rely on local search. They may be quite effective in some circumstances, such as when the objective becomes convex. Quadratic programming (QP) and sequential quadratic programming (SQP) both take advantage of these properties of convexity. While the simplex approach is effective in practice, it presupposes linearity for all problem functions. However, when solving a (LP) issue involving integer variables, the simplex approach cannot be used alone; it must be used in combination with branch and bind [96].

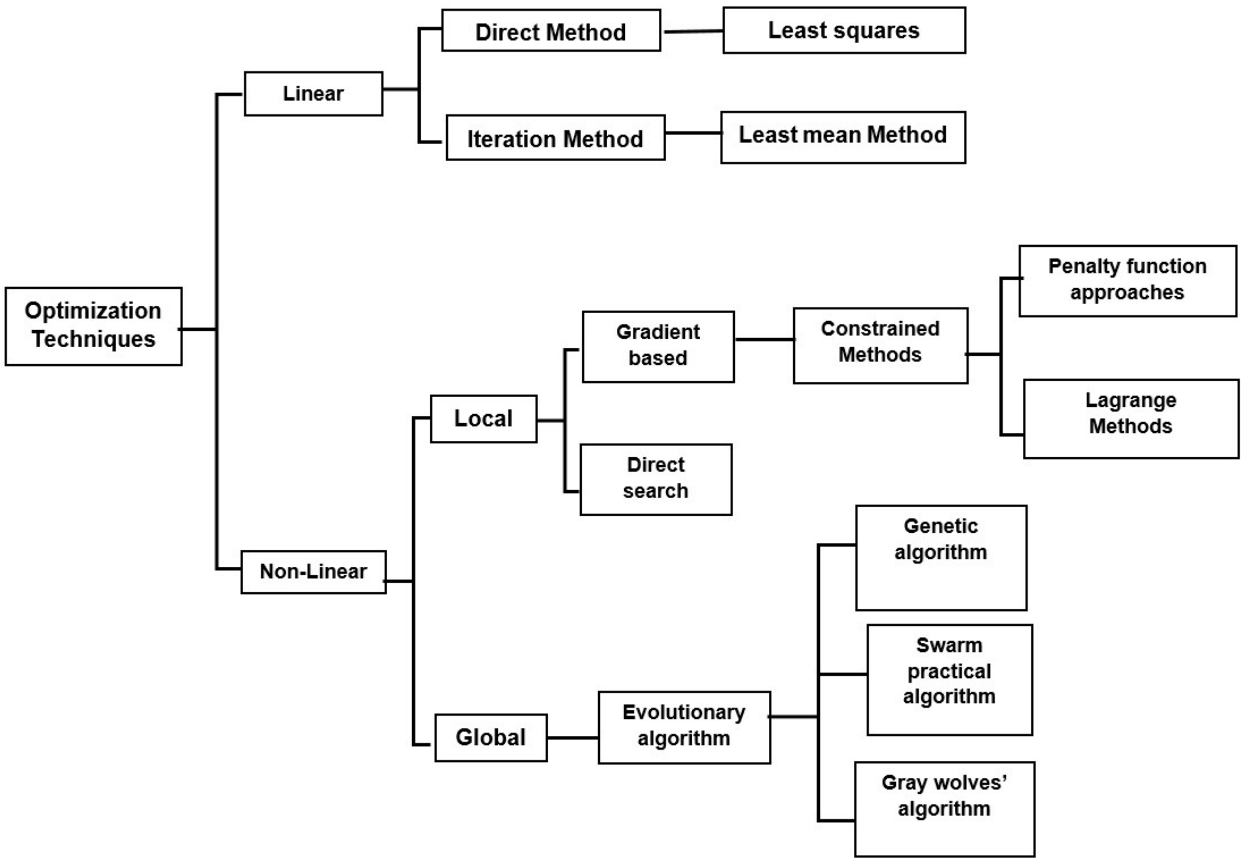


Figure 1.4: Classification of The Optimization Algorithms (Techniques)

CHAPTER 2

MATHEMATICAL BACKGROUND

Introduction

Some significant mathematical ideas will be presented in this chapter so that may apply them in the next chapters.

2.1 Classify Differential Equations

❇ **Ordinary and Partial Differential Equations** [51]:

1. The equation is called an **Ordinary Differential Equations (ODEs)** if the unknown function has just one independent variable, such as the time constant t. Example of ODE:

$$\frac{du}{dt} = 3u \tag{2.1}$$

$$a\frac{d^2u}{dt^2} + b\frac{du}{dt} + cu = f(\chi) \tag{2.2}$$

2. **Partial Differential Equations (PDEs)** are used when there are several independent variables in an unknown function. For example:

$$\frac{\partial^2 u}{\partial \chi^2} + \frac{\partial^2 u}{\partial y^2} = 0 \tag{2.3}$$

$$\frac{\partial^2 u}{\partial t^2} - c\frac{\partial^2 u}{\partial \chi^2} = 0 \tag{2.4}$$

❇ **Order**: Differential Equation Order is determined by how many differentiable terms occur in the equation.

❇ **Linear and Non linear**: A Differential Equation of the form:

$$a_n(t)y^n(t) + a_{n-1}(t)y^{n-1}(t) + ... + a_1(t)y^{'}(t) + a_0(t)y(t) = f(t) \tag{2.5}$$

A linear DE of order n is one in which the functions $a_n(t), \ldots, a_0(t)$ are specified and operate as coefficients of the derivatives of $y^n(t)$. The unknown function, and $f(t)$. Non-linear DEs are DEs that don't follow a straight line [16]. Note the next example:

$$(t^3+1)y^{''}(t)+\sin(t)y^{'}(t)-5y(t)=e^t \tag{2.6}$$

is a linear DE of order 2. while,

$$(t^3+1)y^{''}(t)+\sin(y^{'}(t))-5y(t)=e^t \tag{2.7}$$

Is a non-linear DE of order 2.

❇ **Homogeneous and Non-Homogeneous**: A linear nth-order differential equation is said to be homogeneous when $f(t)$ in equation 2.5 equal to zero, otherwise called non- homogeneous [104].

❇ **Initial Condition(s) , Initial Value and Boundary Value Problem**:
Initial Conditions are a condition, or combination of conditions, on the solution that will help us to identify the solution are looking for. The following are the initial conditions: $y(t_0)=y_0$ or $y^{(k)}(t_0)=y_k$. In other words, beginning conditions specify the solution's and (or) its derivative's values. The differential equations have unique solutions, which means that only one will meet the set restrictions. As will see, the number of beginning conditions required for a specific differential equation is proportional to its order.
An Initial Value Problem (IVP) is a differential equation with a set of initial circumstances. An IVP is provided below:

$$4\chi^2y^{''}+12\chi y^{'}+3y=0 \quad , \quad y(4)=\frac{1}{8} \quad , \quad y^{'}(4)=\frac{-3}{64} \tag{2.8}$$

Boundary Value Problem (BVP) is a differential equation together with a set of additional constraints, called the boundary conditions. A solution to a boundary value problem is a solution to the differential equation which also satisfies the boundary conditions. Boundary value problems arise in several branches of physics and chemistry as any physical or chemical differential equations will have them [21].

❈ **General and Actual Solution**:

The general solution to a DE is the solution in its most general form, taking no initial conditions into consideration. For example:

$$y(t) = \frac{3}{4} + \frac{c}{t^2} \tag{2.9}$$

is a general solution to:

$$2ty' + 4y = 3 \tag{2.10}$$

In a differential equation, **the actual solution** is the particular solution that fulfills both the differential equation and the provided starting condition(s). For example: What is the actual solution to the following IVP in Eq. 2.10 with $y(1) = -4$.
The solution is considerably simpler to do than it seems at first glance. Which is already know that all solutions to the differential equation are of the form 2.9.
All that remains is to identify the value of c that will provide the desired result. All need to do is apply the following condition in the first condition:

$$y(1) = -4 = \frac{3}{4} + \frac{c}{1^2} \tag{2.11}$$

$$c = -4 - \frac{3}{4} = -\frac{19}{4} \tag{2.12}$$

Then, the actual solution to the IVP is.

$$y(t) = \frac{3}{4} - \frac{19}{4t^2} \tag{2.13}$$

❇ **Implicit and Explicit Solution**: An Explicit solution is one that is expressed in the form $y = y(t)$. In other words, y appears just once on the left side and is only increased to the first power. Any answer that is not expressed explicitly is an implicit solution. Note that:

$$y^2 = t^2 - 3 \quad , \tag{2.14}$$

Is the actual implicit solution to:

$$y^{'} = \frac{t}{y} \qquad with \qquad y(2) = -1\,. \tag{2.15}$$

To get the explicit answer, only need to solve for $y(t)$

$$y(t) = \pm\sqrt{t^2 - 3}\,. \tag{2.16}$$

Now, have the problem. There are two functions here, and want just one of them to be accurate. By reapplying the original condition, can find the proper function. Each of them will meet the original requirement in some way. In this scenario, the [$-$] answer is the right one [30]. The actual explicit solution is:

$$y(t) = -\sqrt{t^2 - 3}\,. \tag{2.17}$$

2.2 Formula of Optimization Problem

The following is an example of a general optimization issue [83]:

$$(P_1)\begin{cases} \textit{Optimize [min or max]} & z_\chi = f(\chi) \\ \textit{subject to} & h_j(\chi) = 0 \\ & g_j(\chi) \geq 0 \\ & g_j(\chi) \leq 0 \\ & \chi \geq 0 \end{cases} \tag{2.18}$$

Non-Linear Programming, Mathematical Programming, and Numerical Optimization are used to describe Mathematical Optimization .

An Optimization Problem's $f : R^n \rightarrow R$ purpose is to identify a collection of choice variables χ that maximizes or reduces the objective function z while maintaining the model's h and g constraints. Figure 2.1 shows the iterative approach used illustrates a numerical optimization approach. The picture shows that the optimizer executes the model using a set of decision variable values for χ. The model is based on what is happening in the real world and determines the goal function and associated limitations.

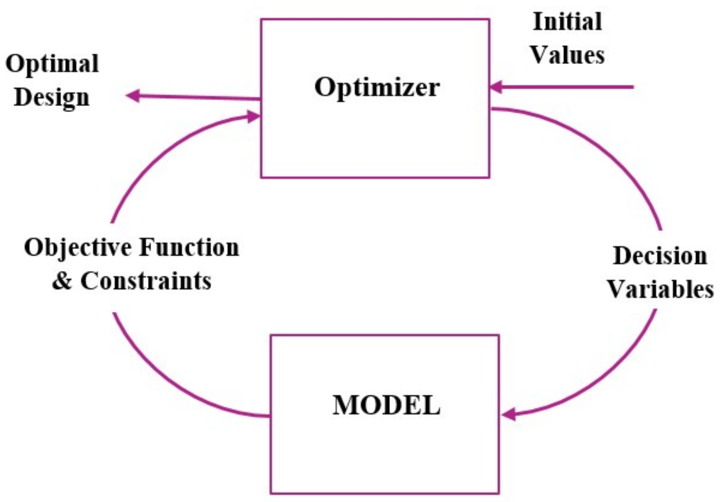


Figure 2.1: A Visual Illustration of The Framework for Numerical Optimization

The optimizer makes use of this information to create a new set of decision variables. This iterative process is repeated until the optimization algorithm's optimization conditions are fulfilled. Numerous software programs are available for numerical optimization, popular software applications like EXCEL, MATLAB, and Python all include optimization tools. Software packages such as Omega and Evolver have spreadsheet interfaces [33].
Systematic Optimization may be achieved via the use of differential equations:

$$(p_2)\begin{cases} \textit{Optimize [min or max]} & \varphi(\chi(t_f)) \\ \textit{subject to} & \chi^{'} = f(\chi,u,p), \chi(0) = \chi_0 \\ & g(\chi,u,p,t) \leq 0, t \in S \\ & h(\chi,u,p,t) = 0, t \in V \end{cases} \tag{2.19}$$

Now, both state variables (χ) and control variables (u) may be considered to be time-dependent functions.

- φ. Objective Function for the very last time.
- g, h . Constrained route spanning S and V time domains
- t_f.Final time, which may be variable or fixed.
- p. Variables or Parameters that are referred to as "constants"(np).

Numerous variables and limits are possible [18].

2.2.1 Concepts of Optimization Problem

Definition 2.2.1. Objective Function [72] , [102]: In a mathematical optimization issue, the objective function is a real-valued function whose value must be minimized or maximized across the set of possible options. In the preceding problem 2.18 or 2.19 , the function f or φ denotes the objective function .

To summarize, using mathematical modeling to solve real-world problems needs the cyclic execution of the four steps . Figure 2.2 shown that.

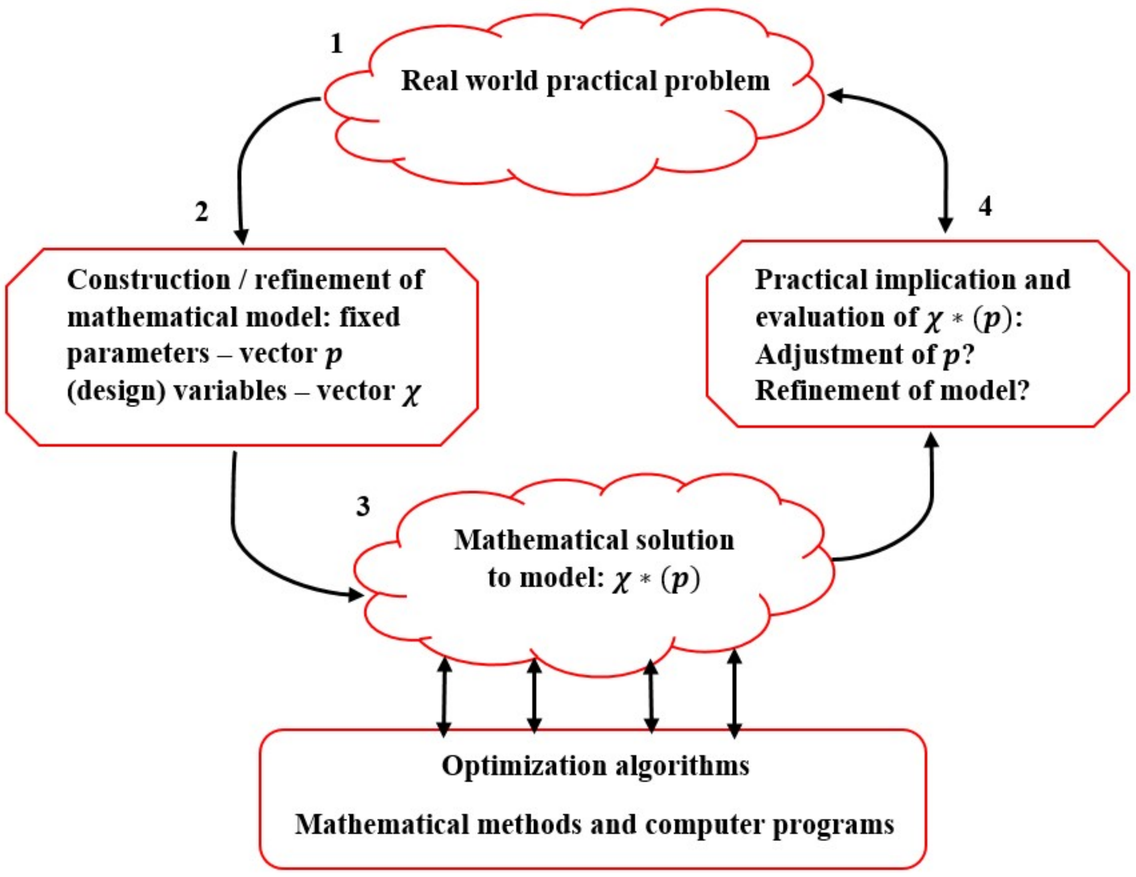

1
Real world practical problem
2
Construction / refinement of mathematical model: fixed parameters – vector p (design) variables – vector χ
4
Practical implication and evaluation of $\chi*(p)$: Adjustment of p? Refinement of model?
3
Mathematical solution to model: $\chi*(p)$
Optimization algorithms
Mathematical methods and computer programs

Figure 2.2: The Mathematical Modeling Process

It's possible that you'll have to go through the modeling process many times before getting a satisfactory result. Typically, the mathematical issue to be addressed in (3) is a numerical optimization problem. The most important part of Realistic Mathematical Optimization is constructing a practical and consistent optimization problem (or model), yet it is also the most ignored [83].

Example 2.2.1. **A real-world practical problem:** How do you boil a potato by heating its boundaries to perfection?

Solution: Let's look at Fraser's recommended model method first.

This basic model is helpful since it gives the formula to correlate data. used Many simplifying assumptions to model the process:

1. A potato is a spherical body with a well-defined form. (Selecting truly spherical potatoes was a challenging effort!).
2. The rate of cooking is determined by the rate at which heat enters the cooking interface.
3. The heat transmitted from the shell to the interface is consumed entirely by the cooking process. (Perhaps this assumption is based on the fact that reaction heat is much higher than ordinary heating effects).
4. The heat transfer driving force can be regarded as constants.

Assume that the temperatures outside the potato and at the cooking interface remain constant at $98°C$ and $65°C$, respectively.
An expression for conduction via a spherical shell is necessary to build the differential equation for this system. So together with all of the previous assumptions, it leads to a differential equation that functions the outer radius and the radius of an uncooked potato at any given moment, according to fraser [26].

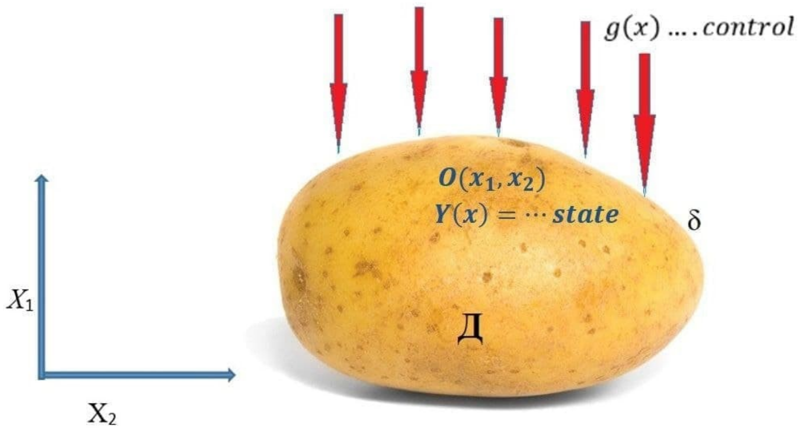


Figure 2.3: Showing: Д is potato, δ its boundary, $y(\chi)$ temp in Д and $g(\chi)$ heat source on δ.

❊ On the boundary δ, choose $g(\chi)$ (control variable), such that $y(\chi)$ (state variable) (heat within the potato), approaches the desired function $y_d(\chi)$.

❊ Control g and state y satisfy the stationary heat equation ($\lambda \geq 0$):

- The Differential Equation is:

$$- \triangle y = 0 \qquad in \quad Д \tag{2.20}$$

$$\frac{\partial y}{\partial n} = \lambda(g(\chi) - y(\chi)) \qquad\qquad on \quad \delta \tag{2.21}$$

- To improve the objective function, so which formulate ($\gamma \geq 0$):

$$\min \phi(y, g) = \frac{1}{2}\int_{Д}(y(\chi) - y_d(\chi))^2 d\chi + \frac{\gamma}{2}\int_{\delta} g(\chi)^2 d\chi \tag{2.22}$$

- Also apply the following (inequality) constraints:

$$g_a(\chi) \leq g(\chi) \leq g_b(\chi) \quad on \quad \delta \tag{2.23}$$

- g_a, g_b are given .

Definition 2.2.2. Decision Variables: [49] In an Optimization Problem, decision variables are variables whose values may change throughout the feasible set of alternatives to improve or decrease the main objective function's value. The vector χ represents the vector of choice variables in the preceding problem 2.18 , 2.19

Definition 2.2.3. Constraint and Unconstrained: [72] The general form 2.18 , 2.19 problems may be characterized by the number of variables, their size, and the smoothness of their relationships with the objective function and constraints (linear, non-linear, convex) (differentiable or no differentiable), among other characteristics. Perhaps the most critical difference is between issues with and without changeable constraints.

Numerous practical applications directly involve unconstrained optimization issues. If variables have inherent limitations, it is sometimes reasonable to overlook them and assume that they do not influence the best solution. Unconstrained issues may alternatively be thought of as Optimizing problems with constraints that have been changed to make them easier to solve. The constraints are replaced with penalty terms in the objective function, which makes the objective function more difficult. Constrained optimization difficulties develop as a result of models with explicit variable restrictions. These constraints may take the form of basic boundaries such as $[0 \leq \chi \leq 100\ ]$, more complicated linear constraints such as $g_j(\chi)$ 2.18 , 2.19 , or non-linear inequalities that express complex interactions between variables [100].

Definition 2.2.4. Feasible Solution: [42, 96] Points that satisfy all restrictions are known as feasible points, and hence they are viable solutions for the issue at hand. Figure 2.4 **(a)** shows that; the feasible area is the collection of all possible points.

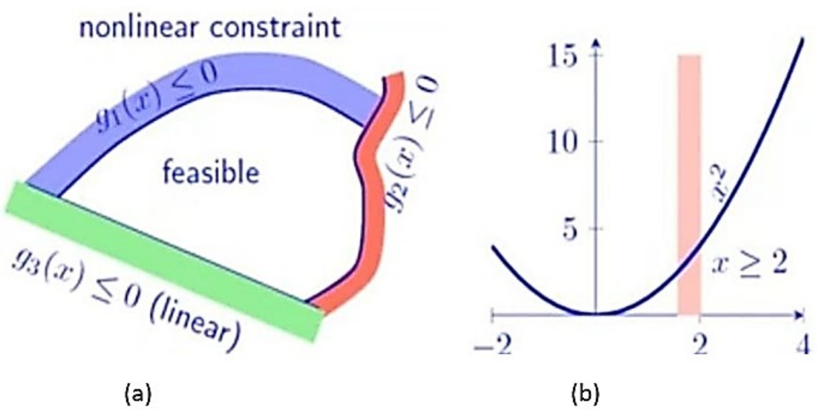


Figure 2.4: **(a)** A feasible domain with non-linear inequality constraints $g_1(\chi)$ and $g_2(\chi)$ and a linear inequality constraint $g_3(\chi)$ (left).
(b) A case in which the objective is $f(\chi) = \chi^2$ subject to: $\chi \geq 2$ (right).

Definition 2.2.5. Infeasible Solutions: [31], [79] If no solution exists that meets all of the restrictions, the issue is said to be infeasible. that fore will look at two forms of infeasibility. The first is called continuous infeasibility, whereas the second is referred to as discrete or integer infeasibility. Continuous infeasibility occurs when a non–MIP issue cannot be solved. The feasible zone formed by the intersecting restrictions is empty in this situation. Discrete or integer infeasibility occurs when a MIP issue has a feasible relaxation (the problem is obtain when remove the discreteness requirement from the variables), but the feasible area of the relaxation has no solution that meets the discreteness criterion. For example:

$$\begin{cases} \min & 200\chi_1 + 300\chi_2 \\ \text{subject to:} & 2\chi_1 + 3\chi_2 \geq 1200 \\ & \chi_1 + \chi_2 \leq 400 \\ & 2\chi_1 + 1.5\chi_2 \geq 900 \\ & \chi_1, \chi_2 \geq 0 \end{cases} \tag{2.24}$$

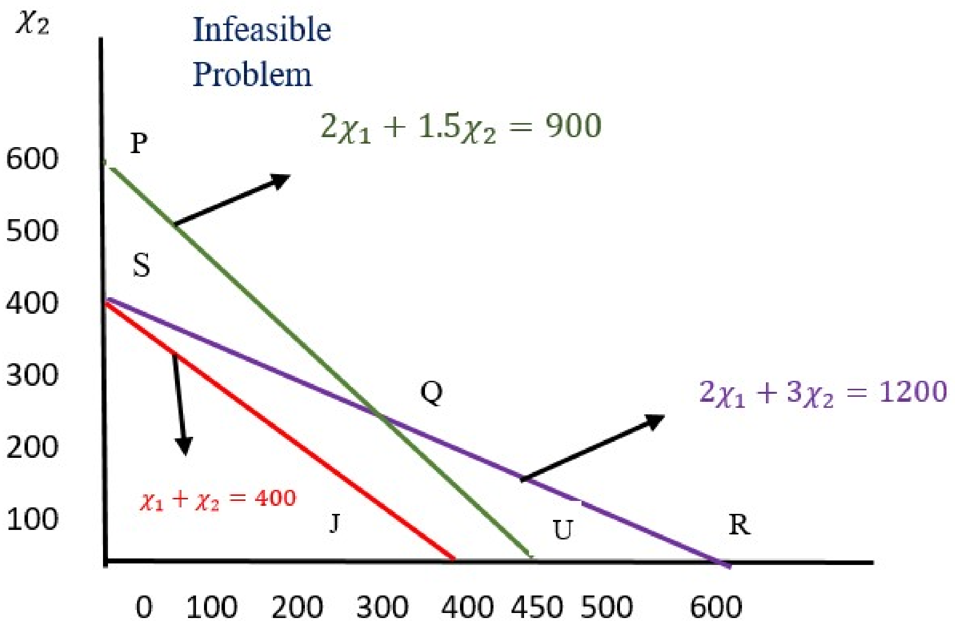


Figure 2.5: First and third constraint-satisfying solutions can be found on the right-hand side of PQR. All solutions that meet a second restriction may be found in the area to the left of SJ. There is no collection of points that fulfil all three requirements; hence the issue is unsolvable.

2.3 Linear Programming Problem

Definition 2.3.1. Linear Programming Problem: [59] A linear program (LP) is a mathematical formulation of the optimization problems. In the unknowable, an objective function is linear, and the constraints are made up of linear equality and inequality constraints. The issue takes on its typical shape:

$$\begin{cases} \min & c^T\chi \\ \text{subject to:} & A\chi = b \quad ; \chi \geq 0 \end{cases} \tag{2.25}$$

In this case, χ denotes an n-dimensional column vector, c^T denotes a row vector with n dimensions, A represents an m-dimensional matrix, and b ; denotes a column vector with m-dimensions.$\chi \geq 0$ is a vector inequality, on the other hand, shows that each of χ components is positive.

For instance [91,96], the Internet service provider (ISP) may provide two unique services, indicated by the letters χ_1 and χ_2, respectively. The first is a set monthly cost with download and capacity constraints, whilst also second is a more expensive option with no download restriction. The first service earns $\alpha\chi_1$, whereas the second product earns $\beta\chi_2$, whereas the profit margin on the second product is bigger than the profit margin on the first $\beta > \alpha > 0$ as a result, the overall profit .

$$P(\chi) = \chi_1 + \chi_2 \quad , \quad \beta/\alpha > 1$$

Since the purpose of the ISP corporation is to maximize profit, furthermore is an objective function. Assuming that the ISP company's entire bandwidth limits the supplied service, no more than $n_1 = 16$ (in 1000000 units) of the first and no more than $n_2 = 10$ (in 1000000 units) of the second may be delivered per unit period, say, each day. As a result, will obtain.

$$\chi_1 \geq n_1$$

and

$$\chi_2 \geq n_2$$

If both service packages demand the same amount of time from the employees, a maximum of $n = 20$ (in 1000000 units) It can be kept up, which equals

$$\chi_1 + \chi_2 \leq n$$

Negative values are unlikely, hence both χ_1 and χ_2 must be non-negative. The following constraints can apply:

$$0 \leq \chi_1 \leq n_1$$

and

$$0 \leq \chi_2 \leq n_2$$

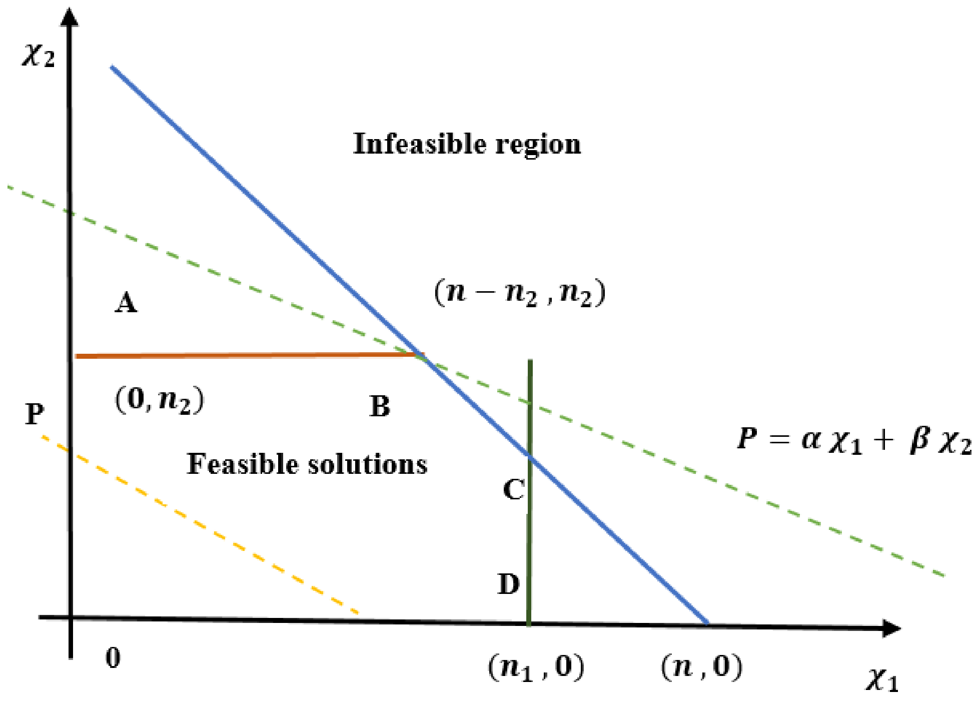


Figure 2.6: Linear Programming is shown schematically. If α is equal to 2 , β is equal to 3 , n_1 is equal to 16 , n_2 is equal to 10 , and n is equal to 20 , then the optimal solution is at $B(10, 10)$.

The objective now is to determine the optimal values for χ_1 and χ_2 so that the profit P is maximized. From a mathematical standpoint, so have

$$\begin{cases} \max\limits_{(\chi_1,\chi_2)\in N_2} & P(\chi_1,\chi_2) = \alpha\chi_1 + \beta\chi_2 \\ \text{subject to :} & \chi_1 + \chi_2 \leq n \\ & 0 \leq \chi_1 \leq n_1 \\ & 0 \leq \chi_2 \leq n_2 \end{cases} \tag{2.26}$$

The possible solutions to issue 2.26 may be represented visually as the inner area of the polygon OABCD, as seen in Figure 2.6. Due to the fact that the objective P is for profit maximization. The optimal answer is found at point B with $(n - n_2, n_2)$, which is the most severe and $P = (n - n_2) + n_2$.

2.4 Non-Linear Programming

Definition 2.4.1. Non-Linear Programming: [22, 33] Problems involving non-linear programming (NLP) have a non-linear objective function or constraints, or both of these. No efficient approaches exist for tackling the non-linear programming issue in its entirety 2.5 , 2.9. When it comes to solving issues with ten variables and hundreds of variables, even the most straightforward situations can be difficult to solve. As a result, numerous techniques to solve the general non-linear programming issue have been developed, each of which entails some degree of compromise.

2.5 Convexity

This is defined as a line passing by way of the points χ_1 and $\chi_2 \in R^n$ [83].

$$L = \{\chi; \chi = \chi_1 + \lambda(\chi_1 - \chi_2), \quad for \quad all \quad \lambda \in R\} \tag{2.27}$$

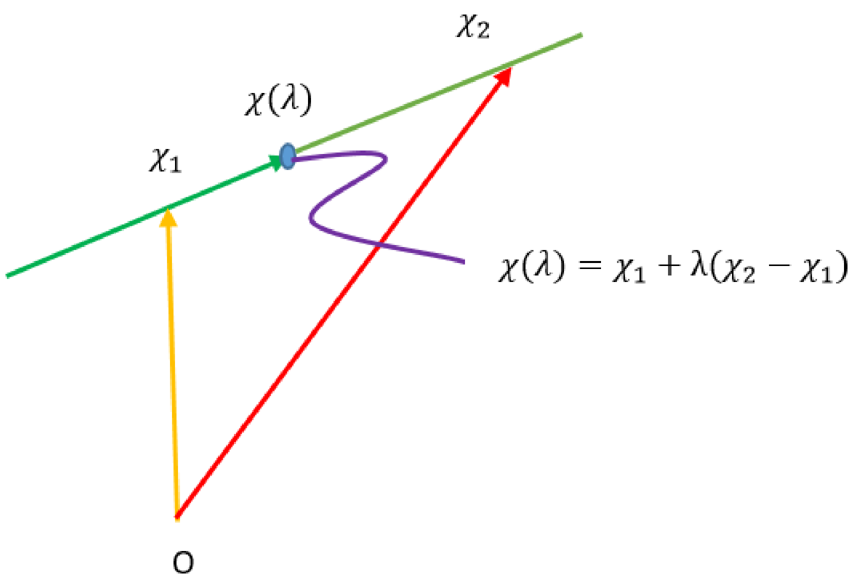


Figure 2.7: A Point on Straight Line that Connects two Points χ_1 and χ_2 is Shown Graphically

Definition 2.5.1. Convex Sets : [8, 83] If a chord links any two points p, q in a set χ , then the set χ is convex. And is also

$$\chi : \lambda p + (1 - \lambda)q \in \chi, \quad for \quad 0 \leq \lambda \leq 1 \tag{2.28}$$

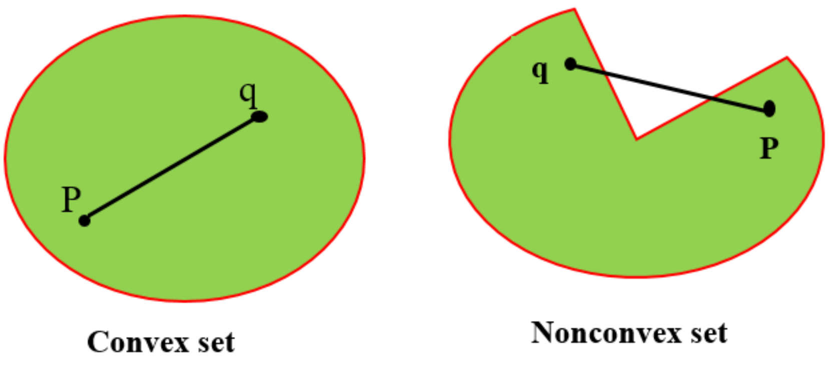


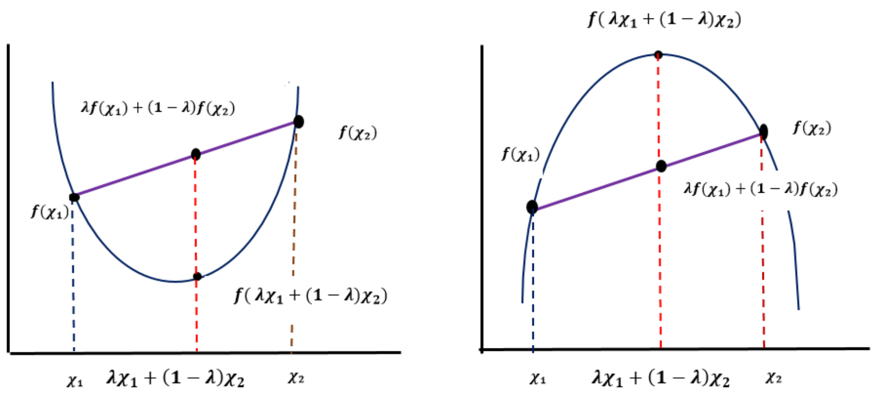
Figure 2.8: Convex and Non-Convex Set

Definition 2.5.2. Convex Functions : [22, 83] A function is said to be convex if the chord that connects two points on its curve is never lower than the curve, Figure 2.9 Mathematically:

$$\mathrm{f}\left(\lambda\chi_1 + (1-\lambda)\chi_2\right) \leq \lambda \mathrm{f}\left(\chi_1\right) + (1-\lambda)\mathrm{f}\left(\chi_2\right), \quad 0 \leq \lambda \leq 1 \tag{2.29}$$

When the function is reversed, it becomes concave:

$$f\left(\lambda\chi_1 + (1-\lambda)\chi_2\right) \geq \lambda f\left(\chi_1\right) + (1-\lambda)f\left(\chi_2\right), \quad 0 \leq \lambda \leq 1 \tag{2.30}$$



$\lambda f(\chi_1) + (1-\lambda)f(\chi_2)$
$f(\chi_2)$
$f(\chi_1)$
$f(\lambda\chi_1 + (1-\lambda)\chi_2)$
χ_1
$\lambda\chi_1 + (1-\lambda)\chi_2$
χ_2
$f(\lambda\chi_1 + (1-\lambda)\chi_2)$
$f(\chi_1)$
$f(\chi_2)$
$\lambda f(\chi_1) + (1-\lambda)f(\chi_2)$
χ_1
$\lambda\chi_1 + (1-\lambda)\chi_2$
χ_2

Figure 2.9: Convex and Concave Functions

Example 2.5.1. With regard to a linear function $f(\chi) = c^T\chi$

$$\begin{aligned} \mathrm{f}\left(\lambda\chi_1 + (1-\lambda)\chi_2\right) &= \mathrm{c}^{\mathrm{T}}\left(\lambda\chi_1 + (1-\lambda)\chi_2\right) \\ &= \lambda \mathrm{c}^{\mathrm{T}}\chi_1 + (1-\lambda)\mathrm{c}^{\mathrm{T}}\chi_2 \\ &= \lambda \mathrm{f}\left(\chi_1\right) + (1-\lambda)\mathrm{f}\left(\chi_2\right). \end{aligned} \tag{2.31}$$

Linear functions are both convex and concave by definition.

Definition 2.5.3. Smooth Function: [40] A distinct slope or gradient for each point on a smooth function. There are no sudden bends or breaks in the line representing a single variable's smooth function. Figure2.10 denoted for smooth functions .

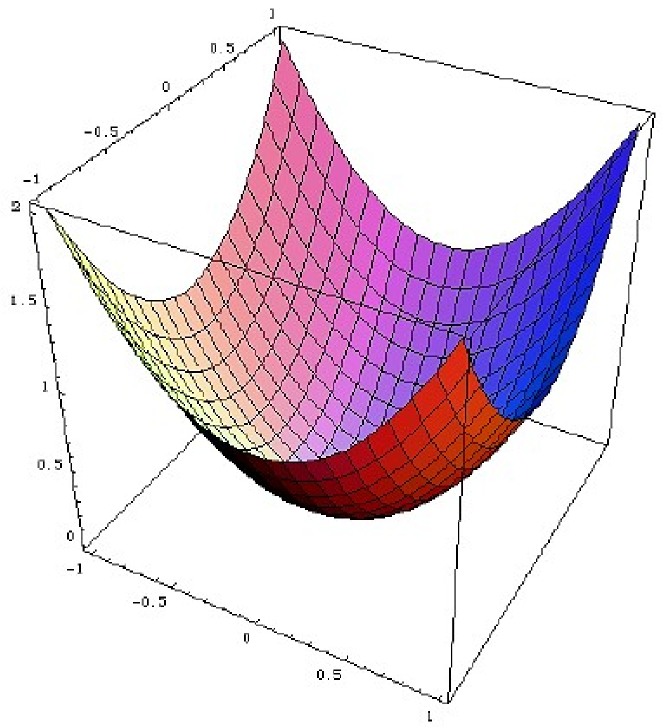


Figure 2.10: Smooth function.

Definition 2.5.4. Non Smooth Function: [12,40] Non-differentiable and discontinuous functions are among the non-smooth functions that are not smooth. The term "non-differentiable" refers to functions whose initial derivatives have undefined areas. Graphs of non-differentiable functions may have abrupt curves. Figure 2.11 show the non-smooth function.

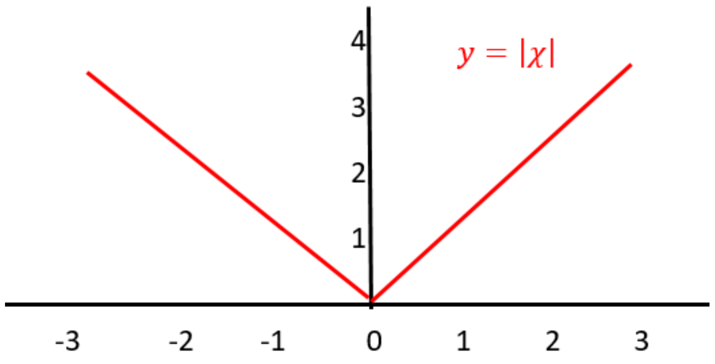


Figure 2.11: Absolute-Value Function.Function $f(\chi) = |\chi|$ is convex and non-smooth, The point $\chi = 0$ be a global minimum of f(χ) .

Definition 2.5.5. Local: [12] , [61] A point $\chi^* \in R^n$ is a problem's local optimal (1.0) if and only if exists $\delta > 0$ such that f $(\chi^*) \leq$ f(χ) for every $\chi \in \mathrm{S} \cap \mathrm{B}(\chi^*; \delta)$. Figure 2.12 shown local minimum points.

Definition 2.5.6. Global: [12] , [61] If f $(\chi^*) \leq$ f(χ) for all $\chi \in$ S, a point $\chi^* \in$ S be a global optimum of the problem 2.18 and 2.19. Figure 2.12 shown global minimum points.

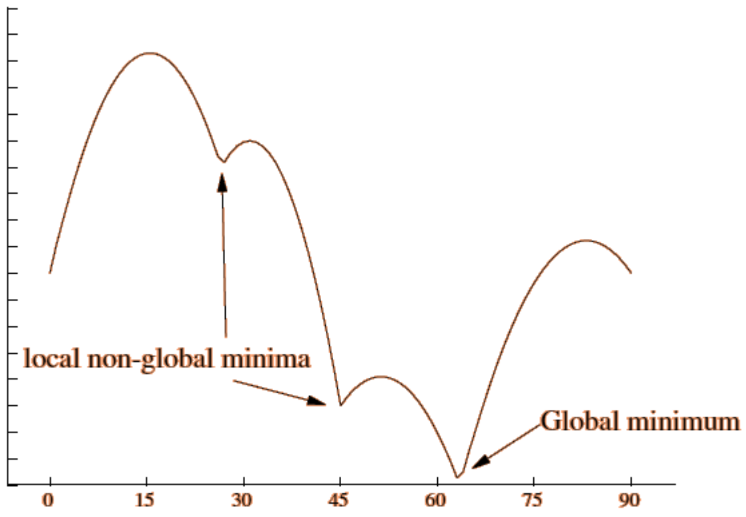
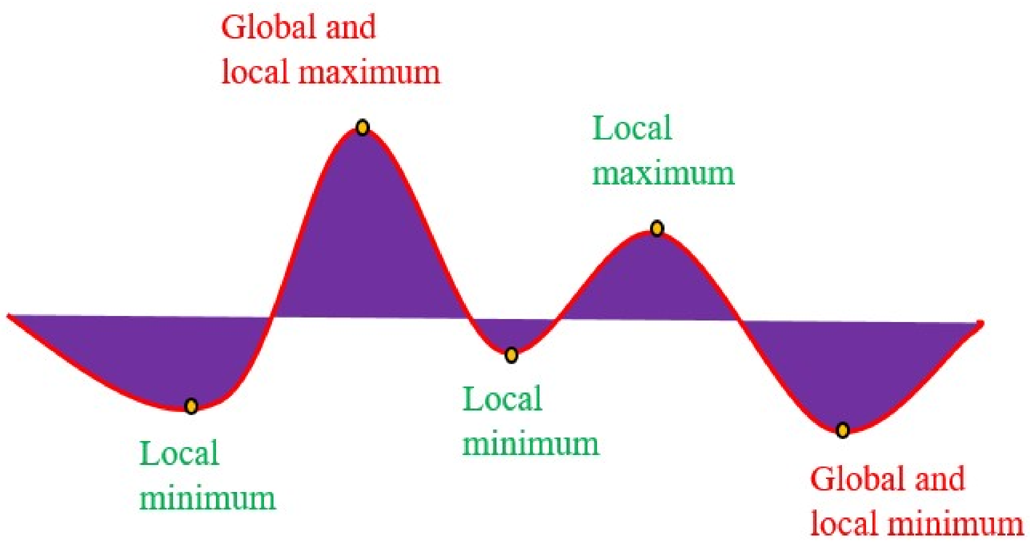


Figure 2.12: Local and Global Minimum.

Theorem 2.5.1. *[12] , [61] Consider the convex set* $(M \in R^p)$ *and the convex function* f*:* $M \to R$*. If* $\chi \in \mathrm{M}$ *is a local minimum of* f *on* M*, then* χ *is a global minimum of* f *on* M*. If* u $\in$ intM *denotes the local maximum of* f*, then* u *denotes the global minimum of* f*.*

Theorem 2.5.2. *To prove the following theorem, let* χ *and* χ^* *be two points in* $\mathbf{R}^{\mathrm{J}}$ *. A point* z *on the line segment* $[\chi^*; \chi]$ *connecting* x *with the* χ^* *is therefore defined as follows:*

- ❇ χ^* *is a global minimizer of* $f(\chi)$ *if* $(\chi - \chi^*) \cdot H(z)\,(\chi - \chi^*) \geq 0$ *for all* χ *and for all* z *in* $[\chi^*; \chi]$*;*
- ❇ χ^* *is a strict global minimizer of* $f(\chi)$ *if* $(\chi - \chi^*)\,.H(z)\,(\chi - \chi^*) > 0$ *for all* $\chi \neq \chi^*$ *and for all* z *in* $[\chi^*; \chi]$*;*
- ❇ χ^* *is a global maximizer of* $f(\chi)$ *if* $(\chi - \chi^*) \cdot H(z)\,(\chi - \chi^*) \leq 0$ *for all* χ *and for all* z *in* $[\chi^*; \chi]$*;*
- ❇ χ^* *is a strict global maximizer of* $f(\chi)$ *if* $(\chi - \chi^*) \cdot H(z)\,(\chi - \chi^*) < 0$ *for all* $\chi \neq \chi^*$ *and for all* z *in* $[\chi^*; \chi]$.



Definition 2.5.7. Vector: [20]It is a number with magnitude and direction, which is what is meant by the term "vector". A vector is a set of ordered scalars or integers in mathematics. **For example**, a three-dimensional vector may express as:

$$a = \begin{pmatrix} a_1 \\ a_2 \\ a_3 \end{pmatrix} = \begin{pmatrix} \chi \\ y \\ z \end{pmatrix} \tag{2.32}$$

Where a_1, a_2, and a_3 are its χ, y, and $z - axis$ components. A vector is normally bolded in minuscule ($\rightarrow$). The component is now a column vector, may also represent as a row vector of the type:

$$\overrightarrow{a} = \begin{pmatrix} a & b & c \end{pmatrix} = \begin{pmatrix} \chi & y & z \end{pmatrix} \tag{2.33}$$

A simple transposition may turn a column vector into a row vector (using the notation T as a superscript):

$$\begin{pmatrix} a_1 & a_2 & a_3 \end{pmatrix}^T = \begin{pmatrix} a_1 \\ a_2 \\ a_3 \end{pmatrix} \text{ or } \begin{pmatrix} \chi & y & z \end{pmatrix}^T = \begin{pmatrix} \chi \\ y \\ z \end{pmatrix} \tag{2.34}$$

The Cartesian norm of a three-dimensional vector is the magnitude or length of the vector.

$$|a| = \sqrt{a_1^2 + a_2^2 + a_3^2} = \sqrt{\chi^2 + y^2 + z^2} \tag{2.35}$$

Definition 2.5.8. Inner Product: [5] A function $\langle .,. \rangle : \mathrm{R^n} \times \mathrm{R^n} \to \mathrm{R}$ is an inner product if :

1. $\langle \chi, \chi \rangle > 0, \langle \chi, \chi \rangle = 0, \forall \chi, \in R^n \chi = 0$ (positivity).

2. $\langle \chi, y \rangle = \langle y, \chi \rangle, \forall \chi, y \in R^n$ (symmetric),

3. $\langle \chi + y, z \rangle = \langle \chi, z \rangle + \langle y, z \rangle, \forall \chi, y, z \in R^n$ (additivity).

4. $\langle \alpha \chi, y \rangle = \alpha \langle \chi, y \rangle$ for all $\alpha \in R, \forall \chi, y \in R^n$ (homogeneity).

Definition 2.5.9. Vector algebra : A vector in a space with n dimensions is called a vector. may express as a column or a row vector in general.

$$\chi = \begin{pmatrix} \chi_1 \\ \chi_2 \\ . \\ . \\ . \\ \chi_n \end{pmatrix} \quad \text{or} \quad \chi = \begin{pmatrix} \chi_1 & \chi_2 & \dots & \chi_n \end{pmatrix} \tag{2.36}$$

Its length may express in the following way:

$$\|\chi\| = \sqrt{\chi_1^2 + \chi_2^2 + \cdots + \chi_n^2} \tag{2.37}$$

Its Euclidean norm is called the norm.

Definition 2.5.10. Norm : [97] Which may define a p-norm or L_p-norm (also L_p-norm) for an n-dimensional vector χ as

$$\|\chi\|_p = (|\chi_1|^p + |\chi_2|^p + \cdots + |\chi_n|^p)^{\frac{1}{p}} = \left(\sum_{i=1}^{n} |\chi_i|^p\right)^{\frac{1}{p}} \ (p > 0) \tag{2.38}$$

If you want to know how long something is, you need to know the Cartesian norm, which is an L_2-norm, such it stands for length in two dimensions.

$$\|\chi\|_2 = \sqrt{\left(|\chi_1|^2 + |\chi_2|^2 + \cdots + |\chi_n|^2\right)} = \sqrt{\chi_1^2 + \chi_2^2 + \cdots + \chi_n^2} \tag{2.39}$$

There are three norms that appear most often are $p = 1, 2$, and when $p = 2$, the Cartesian mentioned above L_2-norm is used. When p is equal to one, the L_1-norm is given by:

$$\|\chi\|_1 = |\chi_1| + |\chi_2| + \cdots + |\chi_n| \tag{2.40}$$

For $p = \infty$, it becomes:

$$\|\chi\|_\infty = \max\{|\chi_1| + |\chi_2| + \cdots + |\chi_n|\} = \chi_{max} \tag{2.41}$$

It is the greatest component of χ in terms of absolute value. This is due to:

$$\begin{aligned}\|\chi\|_\infty &= \lim_{p\to\infty}\left(\sum_{i=1}^{n} |\chi_i|^p\right)^{1/p} \\ &= \lim_{p\to\infty}\left(|\chi_{\max}|^p \sum_{i=1}^{n} \left|\frac{\chi_i}{\chi_{\max}}\right|^p\right)^{1/p} \\ &=\chi_{\max} \lim_{p\to\infty}\left(\sum_{i=1}^{n} \left|\frac{\chi_i}{\chi_{\max}}\right|\right)^{1/p} \\ &=\chi_{\max}\end{aligned} \tag{2.42}$$

There is where have used the fact that $\left|\dfrac{\chi_i}{\chi_{\max}}\right| < 1$, (apart from one component, say,

$|\chi_k| = \chi_{\max}$). Thus, $\lim_{p\to\infty}\left|\frac{\chi_i}{\chi_{\max}}\right|^p \to 0$ for all $i \neq k$. As a result, the total of all ratio terms equals one. That is:

$$\left(\lim_{p\to\infty}\left|\frac{\chi_i}{\chi_{\max}}\right|^p\right)^{1/p} = 1 \tag{2.43}$$

Definition 2.5.11. Frobenius Norm : [56] A matrix's norms may be defined in various ways, much like vector norms. The Frobenius Norm calculates using a matrix of dimensions' $m \times n$.

$$\|A\|_F = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n}|\chi_{ij}|^2} \tag{2.44}$$

This rule establishes the absolute column sum norm at its maximum value.

$$\|A\|_1 = \max_{1\leq j\leq n}\sum_{i=1}^{m}|a_{ij}| \tag{2.45}$$

It's also defined by how many rows of data you have.

$$\|A\|_\infty = \max_{1\leq i\leq m}\sum_{j=1}^{n}|a_{ij}| \tag{2.46}$$

Note that the norm $\|\chi\|$ from R^{n} to R^1 satisfy the following:

1. $\|\chi\| \geq 0, \forall\chi \in \mathrm{R}^n; \|\chi\,| = 0$ only if $\chi = 0$
2. $\|\alpha\chi\| = |\alpha|\|\chi\|; \forall\chi \in R^n$ and $\alpha \in R^1$
3. $\|\chi + y\| \leq \|\gamma\| + \|y\|, \forall\chi, y \in R^n$

Example 2.5.2. For $A = \begin{pmatrix} 10 & -12 & 13 \\ -15 & 0 & 17 \end{pmatrix}$ therefore have:

$$\|A\|_1 = \max\{|10| + |-15|, |-12| + |0|, |13| + |17|\} = \max\{25, 12, 30\} = 30$$

$$\|A\|_\infty = \max\{|10| + |-12| + |13|, |-15| + |0| + |17|\} = \max\{35, 32\} = 35$$

$$\|A\|_F = \sqrt{|10|^2 + |-12|^2 + |13|^2 + |-15|^2 + |0|^2 + |17|^2} = \sqrt{927}$$

Additional rules may be defined for specific purposes.

2.6 Differentiability

Let $F : C \subseteq R^n \to R^m$ have nonempty interiors for n real variables specified on a domain D .

1. If $f' : R^J \to R$ is differentiable, then the gradient of f at χ_0 is $f'(\chi_0) = \bigtriangledown f(\chi_0)$. If each function's initial partial derivatives are continuous, the function is differentiable. $\bigtriangledown f$ is continuous on the set C; for each finite, convex, and differentiable function f, an open convex set C exists.

2. If $f' : R^J \to R^J$ is differentiable, then $f'' : R^J \to R^{J\times J}$, and $f''(\chi) = H(\chi) = \bigtriangledown^2 f(\chi)$, the Hessian matrix whose entries are the second partial derivatives of f, is also differentiable. In this case, the function $f(\chi)$ will be twice differentiable if each of the second partial derivatives is continuous. The chain rule applies because it doesn't matter which variables are in which order, and the Hessian matrix is symmetric when it comes to mixed second partial derivatives. Consider the function $f : R^J \to R$ to be a differentiable function [5].

Definition 2.6.1. [5] Let $f : R^n \to R$ and $\chi \in R^n$. The expression denotes the partial derivative of $f(\chi)$ at χ concerning χ_i;

$$\frac{\partial f(\chi)}{\chi_i} = \lim_{t\to 0} \frac{f(\chi + te_i) - f(\chi)}{t} \tag{2.47}$$

Where (e_i) is an ith unit vector.

The gradient of $f(\chi)$ at χ is defined as the column vector

$$\nabla f(\chi) = \begin{bmatrix} \dfrac{\partial f(\chi)}{\partial \chi_1} \\ \dfrac{\partial f(\chi)}{\partial \chi_2} \\ . \\ \dfrac{\partial f(\chi)}{\partial \chi_n} \end{bmatrix} \tag{2.48}$$

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

$$\nabla^2 f(\chi) = \begin{bmatrix} \dfrac{\partial^2 f(\chi)}{\partial \chi_1^2} & \dfrac{\partial^2 f(\chi)}{\partial \chi_1 \partial \chi_2} & \cdots & \dfrac{\partial^2 f(\chi)}{\partial \chi_1 \partial \chi_n} \\ \vdots & \ddots & \vdots & \\ \dfrac{\partial^2 f(\chi)}{\partial \chi_n \partial \chi_1} & \dfrac{\partial^2 f(\chi)}{\partial \chi_n \partial \chi_2} & \cdots & \dfrac{\partial^2 f(\chi)}{\partial \chi_n^2} \end{bmatrix} \tag{2.49}$$

The directional derivative of the function f at χ in the direction d given by:

$$f'(\chi, d) = \lim_{\chi \to 0^+} \frac{f(\chi + td) - f(\chi)}{t} \tag{2.50}$$

If and only if the gradient $\bigtriangledown f(\chi)$ exists, so that the function f can be differentiable at a point called χ and satisfies $\langle \bigtriangledown f(\chi), d \rangle = f'(\chi, d)$,$\forall \chi \in R^{^n}$, f is also differentiable over a subset of R^n called S if it can be differentiable at every $\chi \in S$, then f is continuously differentiable over S if,

$$\lim_{d \to 0} \frac{f(x + td) - f(\chi) - \langle \nabla f(\chi), d \rangle}{\|d\|} = 0 \tag{2.51}$$

where $\|\|$ is an arbitrary vector norm [27].

2.6.1 The Importance of Iterations

From elementary calculus, so know that if want for wish to optimize a differentiable function $g(\chi)$ for a single real variable χ, must first define the function, and locate the points at which the derivative is zero, $g'(\chi) = 0$. by a similar vein, if want for wish to optimize a differentiable function $g(\chi)$ of a real vector variable χ, may use the same approach; so it begin by identifying the locations where the gradient $\bigtriangledown g(\chi)$ equals zero [24]. However, this is not the end of the tale in general since which is still need to solve an equation for the optimal χ. If we are very lucky, may not be able to solve this problem algebraically at all, or it may even be impossible, necessitating the use of iterative approaches. It shows that might immediately minimize $g(\chi)$ using iterative approaches rather than solving an equation. Imagine the over-determined system of linear equations $A\chi = b$ that want to solve but are uncertain about the existence of solutions. In this instance, may prefer to reduce the function's size [70].

$$g(\chi) = \frac{1}{2}\|A\chi - b\|^2 \tag{2.52}$$

To get the least-squares answer. If the matrix $A^T A$ is invertible, the unique minimizer of $g(\chi)$ is well-known and given by the matrix $A^T A$.

$$\chi^* = (A^T A)^{-1} A^T b \tag{2.53}$$

There might be an immense number of equations and unknowns in certain cases, making it too costly to compute even the matrix $A^T A$'s entries. As a result, might determine χ by using an iterative approach, such as Algorithms.

2.6.2 Maximum and Minimum may be found by following the directions below

Assume $g : R^J \to R$ is differentiable and converges to its smallest value. Would want to make the function g smaller in overall size (χ). It may be difficult to solve $\bigtriangledown g(\chi) = 0$ to get the optimal $\chi = (\chi^*)$, but it is possible. In which case, may use an iterative technique. For locating $\bigtriangledown g(\chi)$ roots or one that directly minimizes $g(\chi)$. In the latter scenario, the steepest descent method may be considered once again.

$$\chi^{k+1} = \chi^k - \propto \bigtriangledown g(\chi^k) \tag{2.54}$$

For some $\propto > 0$, T stands for the operator, which indicate.

$$T\chi = \chi - \propto \bigtriangledown g(\chi) \tag{2.55}$$

Then, with $\bigtriangledown g(\chi) = 0$, see that:

$$\|\chi^* - \chi^k\|_2 = \|T\chi^* - T\chi^k\|_2 \tag{2.56}$$

so will interested in learning if any options imply convergence of the iterative sequence. As with single-variable functions, the answer is true for convex functions $g(\chi)$.

Example 2.6.1. To minimize the function, use the steepest descent approach:

$$f(\chi) = 2\chi_1^2 + 2\chi_2^2 + 6\chi_1 + 13 \tag{2.57}$$

With initial point $\chi_0 = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$, putting χ_0 in Eq. 2.57, therefore getting :

$$f(\chi_0) = 2(1^2) + 2(2^2) + 6(1) + 13 = 29$$

The gradient and the hessian of the function

$$\nabla f = \begin{bmatrix} \dfrac{\partial f}{\partial \chi_1} \\ \dfrac{\partial f}{\partial \chi_2} \end{bmatrix} = \begin{bmatrix} 4\chi_1 + 6 \\ \\ 4\chi_2 \end{bmatrix} \tag{2.58}$$

Putting χ_0 in Eq.2.58 getting:

$$\nabla f = \begin{bmatrix} 10 \\ \\ 8 \end{bmatrix}$$

$$H = \begin{bmatrix} \dfrac{\partial^2 f}{\partial^2 \chi_1} & \dfrac{\partial^2 f}{\partial \chi_1 \partial \chi_2} \\ \dfrac{\partial^2 f}{\partial \chi_2 \partial \chi_1} & \dfrac{\partial^2 f}{\partial^2 \chi_2} \end{bmatrix} \tag{2.59}$$

At χ_0 in Eq. 2.59 getting:

$$H = \begin{bmatrix} 4 & 0 \\ \\ 0 & 4 \end{bmatrix} \tag{2.60}$$

The step length of the quadratic function is computed from as:

$$\alpha_0 = \frac{\nabla^T f(\chi_0) \nabla f(\chi_0)}{\nabla^T f(\chi_0) H \nabla f(\chi)} | \tag{2.61}$$

$$\alpha_0 = \frac{\begin{bmatrix} 10 \\ 8 \end{bmatrix}^T \begin{bmatrix} 10 \\ 8 \end{bmatrix}}{\begin{bmatrix} 40 \\ 32 \end{bmatrix}^T \begin{bmatrix} 10 \\ 8 \end{bmatrix}} = \frac{100+64}{400+256} = \frac{164}{656} = 0.25 \tag{2.62}$$

$$\chi_1 = \chi_0 - \alpha_0 \nabla f(\chi_0) \tag{2.63}$$

When put χ_0 in Eq. 2.63 getting:

$$\chi_1 = \begin{bmatrix} 1 \\ 2 \end{bmatrix} - (0.25) \begin{bmatrix} 10 \\ 8 \end{bmatrix}$$

$$\chi_1 = \begin{bmatrix} 1 \\ 2 \end{bmatrix} - \begin{bmatrix} 2.5 \\ 2 \end{bmatrix} = \begin{bmatrix} -1.5 \\ 0 \end{bmatrix} \tag{2.64}$$

When compensating a value of χ_1 in Eq. 2.58 getting:

$$\nabla f(\chi_1) = \begin{bmatrix} 4(-1.5)+6 \\ 4(0) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{2.65}$$

Which can conclude:

1. The steepest descent method comes to a halt when the gradient of at χ_1 equals zero.

2. The minimum point is $\chi^* = \chi_1 = \begin{bmatrix} -1.5 \\ 0 \end{bmatrix}$, which is the optimal value.

3. The minimum value of the objective function is $f(\chi^*)$.

4. The Optimal solution: $f(\chi^*) = 2 * (-1.5) + 2 * (0) + 6 * (-1.5) + 13 = 8.5$.

2.7 Integration Function

The definite integral of a real-valued continuous function $f(\chi)$, defined on a real interval $[a, b]$, and for which the primitive (antiderivative) $F(\chi)$ can be found, The Newton–Leibniz formula is capable of being used to assess:

$$\int_a^b f(\chi)d\chi = F(b) - F(a) \tag{2.66}$$

However, in real applications, the basic $F(\chi)$ is unavailable or requires an enormous amount of analytical work to determine elementary methods. Occasionally, the integrand $f(\chi)$ is defined just as a table of numerical values, rendering the primitive idea unusable. Additionally, the intrinsic complexities of variable coupling in many multidimensional scenarios preclude analytical integration entirely. These and several more practical considerations demonstrate the critical necessity of developing numerical methods for computing definite integrals. Quadrature is another term for numerical integration. While the phrase is often used to refer to one-dimensional integrals, in a broader sense, it is also used in conjunction with the term cubature to refer to higher-dimensional integration [17]. In general, every quadrature formula substitutes the integral with a weighted sum of the values of the integrands at specified locations in the integration domain, $\chi_i \in [a, b]$, and contains a unique truncation error (error term) R_n:

$$\int_a^b f(\chi)d\chi = \sum_{i=1}^{n} w_i f(\chi_i) + R_n \tag{2.67}$$

And an open quadrature formula uses just information from the interior locations of the integration domain, whereas a closed quadrature formula includes information from the integrand values on the borders [24].

2.8 Optimality Conditions

Initially let's get to know Tyler's sequence before explaining the optimal conditions with the proof:

Theorem 2.8.1. ***(Taylor Expansion):*** *[5]*
Allow for continuous differentiation of $\mathrm{f} : \mathrm{R^n} \to \mathrm{R}$. *Then, for all* $\chi_1; \chi_2 \in \mathrm{R^n}$, *as a result of the* $\alpha \in [0, 1]$

$$f(\chi_2) = f(\chi_1) + \nabla f(\alpha\chi_1 + (1-\alpha)\chi_2)^T (\chi_2 - \chi_1).$$

Additionally, there exists $\alpha \in [0; 1]$ *if* f *is twice continuously differentiable; for any* χ_1; $\chi_2 \in \mathrm{R^n}$.

$$\begin{aligned} f(\chi_2) = f(\chi_1) &+ \nabla f(\chi_1)^T (\chi_2 - \chi_1) \\ &+ \frac{1}{2}(\chi_2 - \chi_1)^T \nabla^2 f(\alpha\chi_1 + (1-\alpha)\chi_2)^T (\chi_2 - \chi_1). \end{aligned}$$

Additionally, if have $\chi; \mathrm{u} \in \mathrm{IR^n}$ *and* $\eta \in \mathrm{IR}$, *therefore:*

$$f(\chi + \eta u) = f(\chi) + \eta u^T \nabla f(\chi) + \frac{\eta^2}{2} u^T \nabla^2 f(\chi) u + O\left(\eta^2\right), \text{ as } \eta \to 0.$$

2.8.1 Optimality Conditions for Unconstrained Optimization

Theorem 2.8.2. ***First-Order Necessary Condition (FONC)*** *[84]*

An open set S in which the local minimum is located may be defined as the set $f : R^n \to R$. $\bigtriangledown f(\chi)$ then equals zero.

Theorem 2.8.3. ***Second-Order Necessary Condition(SONC)*** *[86]*

Let $f : D \subset R^n \to R$ be twice continuously differentiable on an open set D. If χ^* is a local minimizer of 2.8.2, then $\nabla f(\chi^*) = 0$ and $\nabla^2 f(\chi^*)$ is positive semidefinite.

Theorem 2.8.4. ***Second-Order Sufficient Condition (SOSC)*** *[86]*

Let $f : \mathbb{R}^n \to \mathbb{R}$ be twice continuously differentiable. If $\nabla f(\chi^*) = 0$ and $\nabla^2 f(\chi^*)$ is positive definite, then χ^* is a strict local minimizer.

2.8.2 Optimality Conditions for Constraints

The constraint system can be converted to unrestricted using **(Lagrange Multiplier Rule)**

Theorem 2.8.5. ***(Lagrange Multiplier Rule):*** *[29] If χ^* is a local minimizer for a constraint problem, then there exists a sequence of numbers $(\lambda_1, \ldots, \lambda_0)$ such that;*

$$\nabla f(\chi^*) - \sum_{1}^{\ell} \lambda_i^* \nabla c_i(\chi^*) = 0 \quad , i = 1 \ldots \text{n}$$

And

$$c_i(\chi^*) = 0, \quad i = 1, \ldots, \ell$$

The numbers λ_i are referred to as Lagrange Multipliers.

Theorem 2.8.6. ***First-Order Necessary Condition:*** *[29, 90] If χ^* is a local minimizer of $f(\chi)$ and $f(\chi)$ is disputably differentiable in χ^* s open neighborhood $\left(f \in C^1\right)$, then $\nabla f(\chi) = 0$. In other words, $f(\chi)$ is stationary at χ^*.*

Theorem 2.8.7. ***Second-Order Necessary Condition:*** *[29, 84] If the function f is twice continuously differentiable on a set X in R^n, and $\chi^* \in X$ is a local minimizer of f, then for each viable direction p at χ^*, and $p \in R^n$;*

1. $\nabla f(\chi^*)\, p \geq 0$
2. *if* $\nabla f(\chi^*)\, p = 0$*, then* $p^T \nabla^2 f(\chi^*) p \geq 0.$

condition 2 means that $\nabla^2 f(\chi^*)$ must be positive semidefinite.

Theorem 2.8.8. ***Second-order sufficient conditions:*** *[29,86] Consider f to be a twice continuously differentiable function on the set $X \in R^n$. Let $\bar{\chi}$ represent an interior point of X. If*

1. $\nabla f(\bar{\chi}) = 0.$
2. $\nabla^2 f(\bar{\chi})$ *is positive definite.*

2.9 Application of Optimization in General

Numerical optimization techniques can be applied to a wide variety of areas that can be solved mathematically. Additionally, there are some areas that have their mathematical model in the form of differential equations, whether ordinary or partial, and thus use optimization to solve differential equations:

1. **Mechanics**
   Rigid body dynamics problems (particularly articulated rigid body dynamics problems) frequently require mathematical programming techniques, as rigid body dynamics can be viewed as an attempt to solve an ordinary differential equation on a constraint manifold, where the constraints are various nonlinear geometric

constraints such as "these two points must always coincide", "this surface must not penetrate any other", or "this point must always lie somewhere on this curve". Contact force computation may be achieved by solving a linear complementarity problem, which is a QP (Quadratic Programming) problem [1]. Numerous design difficulties may also be classified as optimization programs. This is referred to as optimization of the design. One subset is engineering optimization, while a more recent and fast-growing subset is interdisciplinary design optimization. Although relevant to a broad range of problems, it has been applied to aeronautical engineering challenges in particular. This method holds for cosmology and astrophysics as well [63].

2. **Economics and Finance**

   Economics is sufficiently intertwined with agent optimization that one important definition characterizes economics as a science as the "study of human behavior as a connection between aims and limited means" with alternate applications. Modern optimization theory encompasses both classical optimization theory and the study of economic equilibrium. Optimization methods, and related subjects. The utility maximization problem and its twin, the expenditure minimization issue, are economic optimization problems in microeconomics. Consumers are expected to maximize their utility since their behavior is consistent, but corporations are often believed to maximize profit . Additionally, agents are often depicted as risk-averse, seeking to avoid danger. Optimization theory is also used to simulate asset values, albeit the underlying mathematics is based on optimizing stochastic processes rather than static optimization. International trade theory also makes use of optimization to understand international trade trends. Portfolio optimization is a kind of multi-objective optimization used in economics. Since the 1970s, economists have used control theory to describe dynamic choices across time. Dynamic search models, for example, are used to examine labor market behavior. The contrast between deterministic and stochastic models is critical.

Macroeconomists develop dynamic stochastic general equilibrium (DSGE) models to capture the dynamics of the whole economy as a consequence of the interconnected optimal choices made by workers, consumers, investors, and governments [92] .

3. **Electrical Engineering**

   Optimization techniques are frequently used in electrical engineering for a variety of purposes, including active filter design, stray field reduction in superconducting magnetic energy storage systems, space mapping design of microwave structures and handset antennas, and electromagnetics-based design. Since the discovery of space mapping in 1993, electromagnetically verified design optimization of microwave components and antennas have made substantial use of a suitable physics-based or empirical surrogate model and space mapping methodology [37].

4. **Civil Engineering**

   Civil engineering has made extensive use of optimization. Construction management and transportation engineering are two of the most important fields of civil engineering that place a high premium on optimization. Optimization is often used to tackle civil engineering issues such as cut and fill, life-cycle analysis of structures and infrastructures, resource levelling, water resource allocation, traffic management, and timetable optimization [98].

5. **Operations Research**

   Operations Research is another subject that makes considerable use of optimization methods. Additionally, operations research makes use of stochastic modelling and simulation to aid in decision-making. Operations research is increasingly using stochastic programming to describe dynamic choices that adapt to changing conditions; such issues may be handled using large-scale optimization and stochastic optimization techniques [64, 87].

6. **Control Engineering**

   Mathematical Optimization is employed extensively in the design of contemporary controllers. Mathematical optimization is used in high-level controllers such as Model Predictive Control (MPC) and Real-Time Optimization (RTO). These algorithms operate in the background and regularly generate values for decision variables such as choke. Iteratively solve a mathematical optimization problem that involves restrictions and a description of the system to be controlled to find openings in a process plant [69, 73].

7. **Geophysics**

   Optimization techniques are often used to address geophysical parameter estimation challenges. The physical properties and geometrical shapes of the underlying rocks and fluids are often determined using a series of geophysical measurements, such as seismic recordings. The overwhelming majority of geophysical problems are nonlinear in nature, and both deterministic and stochastic techniques are often used [62].

8. **Computational Systems Biology**

   Numerous elements of computational systems biology make use of optimization approaches, including model building, optimal experimental design, metabolic engineering, and synthetic biology. Linear programming has been used to predict the highest practical yields of fermentation products and to infer gene regulatory networks from a variety of microarray datasets and transcriptional regulatory networks from high-throughput data. Nonlinear programming has been used to investigate energy metabolism and to create and evaluate the parameters of metabolic circuits [77].

9. **Machine Learning**

   Additionally, Machine Learning is intrinsically tied to optimization: many learning problems are characterized in terms of the minimization of a loss function over a training set of samples. Loss functions are used to assess the discrepancy between model predictions and actual problem scenarios (for example, in classification, one wants to assign a label to instances, and models are trained to correctly predict the pre-assigned labels of a set of examples) [84].

CHAPTER 3

MATHEMATICAL OPTIMIZATION FOR SOLVING DIFFERENTIAL EQUATION MODEL OF COVID-19 NUMERICALLY

Introduction

The global COVID-19 pandemic in 2020 wreaked havoc, costing billions of dollars and claiming the lives of thousands of people. Much of the economic harm is attributable to government attempts to slow the development of the illness; although these measures are unquestionably successful, they come at a colossal cost. Consequently, much study has been concentrated on mathematical modeling of epidemics, and epidemics in general, hoping that these models would someday prove beneficial to decision-makers and guide more targeted and less disruptive actions [65]. Many modeling methods have been presented to analyze the efficiency of various social distancing tactics, combining differential equations and empirical methodologies. Many of these models discretize distinct areas along geopolitical lines (or the like) to incorporate spatial differences across multiple regions, employing a network structure to reflect mobility between people in different regions. In contrast to these methodologies, the scientists used partial differential equation (PDE) models to predict the disease's geographical spread [82]. The models were built using the partitioning framework, but they also included nonlinear heterogeneous diffusion elements, resulting in the equations' reaction-diffusion system. While such an approach has a substantially greater computing cost than the ode model, numerical testing has demonstrated that it has several benefits. The timing of various dynamics over different areas may be addressed continuously, allowing for a more detailed description of spatiotemporal development. Will want to use the ordinary differential equations model to simulate various dynamics, including the incubation time. Ordinary differential equation models and other biological forms models, have been frequently utilized to research epidemics [44].

3.1 Mathematical Modeling of Understand The Epidemiology

The COVID-19 pandemic is a major concern, with a high death rate and a worsening economic impact. There is already a lot of interest in the pandemic's length, expected patient numbers, quarantine estimations, and recurrence rates. Mathematicians with expertise in infectious disease modeling are needed for the COVID-19 pandemic. To estimate the values of unknown parameters in a mathematical model, just a modest number of disease observations throughout time are necessary. Even long-term mention of the epidemic is unreliable because of the constant changes in quarantine and testing settings, algorithms for segregating unwell persons, pathogen activity, and so forth. Any mathematical model of epidemic dynamics would be beneficial if could reliably anticipate the epidemic's duration and sickness load using data obtained immediately after the commencement. This has led to efforts to predict the dynamics of the COVID-19 pandemic in different countries and regions. For two reasons, it is challenging to reproduce the COVID-19 pandemic statistically [95]. To begin with, there is a shortage of accurate statistics on the number of instances. People who don't show any symptoms and doctors who can't find out what's going on are responsible. A substantial discrepancy between reported and actual patient counts persisted following the COVID-19 epidemic. It's challenging to estimate accurately since which don't know when the reported incidents will reach their correct total number. Secondly, the situation around the epidemic is ever-changing (quarantine measures, social behavior, pathogen virulence, etc.). As a consequence, previous data cannot be used to predict the future. Depending on unique coronary pneumonia transmission features, these dynamic infectious disease models are evaluated in three time periods, based on the availability of information and published scientific studies on unknown coronary pneumonia epidemics. Mathematical models may be used to make objective and effective decisions in the fight against the disease. They have supported and will continue to support policies and actions aimed at preventing problems with COVID-19. The hunt for mathematical models, or models that characterize illness behavior has lately re-emerged for infectious disease researchers. Bernoulli developed mathematical models for illnesses like smallpox, which devastated England in 1760. Other than that, Ronald Ross was awarded the Nobel Prize in 1902 for his work on the human malaria cycle, the mosquito vector, and the parasite itself [43].

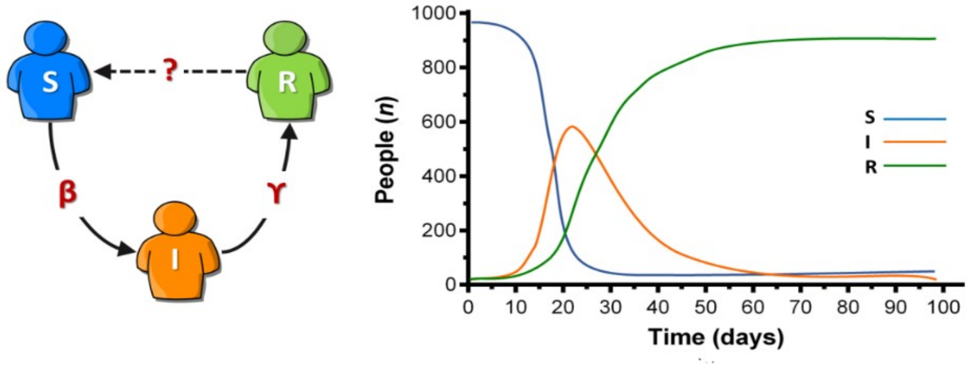


Figure 3.1: Model showing the COVID-19 epidemic

1. **Malthus - Euler Model:** The first works are known as Euler (1707-1783) (general research on the mortality and multiplication of the human race (1760)) in particular Euler calculated the population of a city or county for a given year, he found that the population P_n For a year n she stuck regressive $P_{n+1} = \lambda P_n$. It leads towards a geometric sequence with a basis $\lambda > 1$, and this gives rapid growth. Which also can provide the Model time constant for this phenomenon by assuming that the population is increasing, competing with it $P(t)$. And so that got a differential equation $P'(t) = \lambda P(t)$, and after solving it, find $P(t) = ce^{\lambda t}$. The idea of growing the ASEAN community was developed in 1798 by Robert Malthus (1766-1834). His analysis led to the modelling of a population. It showed that the increase is done in a geometric sequence, while the growth of Natural Resources is based on a mathematical sequence [14].

2. **Verhulst (Logistic) Population Model:** The logistic function, which was created and subsequently proven by Belgian mathematician Pierre-François Verhulst (1804-1849), was employed in fields such as statistics, economics, and epidemiology. Additionally, it has simulated area overcrowding and bacteria development in broths and bilateral economic and financial decision-making processes [82]. The differential equation for the logistic function used to characterize the spread of COVID - 19 coronavirus in various nations and cities is as follows:

$$\frac{dN}{dt} = \alpha N \left(1 - \frac{N}{k}\right) \tag{3.1}$$

If $N(0) = N_0$, and $t \geq 0$ then the unique solution of a logistic equation is:

$$N(t) = \frac{k}{\left(\frac{k}{N_0} - 1\right) e^{-\alpha t} + 1} \tag{3.2}$$

Or if $\alpha = k$ and $N(t) = P(t)$ then the equation will be [9]

$$P(t) = \frac{k}{\left(\frac{k}{N_0} - 1\right) e^{-kt} + 1} \tag{3.3}$$

3. **The Model SIR:** The SIR epidemic model was the first epidemic model presented by Kermack-McKendrick in 1927. The SIR model is a basic one that divides the population into separate groups of people. The following are the locations of the three groups:

❇ **S= Susceptible Class:** Individuals who are not infectious but may develop the illness and become infectious.

❇ **I = Infectious class:** People who have the potential to spread sickness to others.

❇ **R = Removed Class:** Individuals who have had the illness and one has died (or) who have recovered and are always immune, (or) who have been isolated till they recover.

With these three groups, the transmission of the virus is expected to be governed by the following assumptions. First, individuals are eliminated from the infectious class at a rate proportionate to the magnitude of I. The population size remains constant. As a result, the model is represented by the system of ODEs below [48, 89, 93].

$$\begin{aligned} \frac{dS}{dt} &= -\beta SI \\ \frac{dI}{dt} &= \beta SI - \gamma I \\ \frac{dR}{dt} &= \gamma I \end{aligned} \tag{3.4}$$

Where initial conditions $S(0) = S_0, I(0) = I_0, R(0) = 0$. β Represents the rate of transmission and γ calls the recovery rate. Other models of the necessity of conditions were derived from the model SIR:

- **The SEIR model:** Host populations may be viewed as being made up of people who can be categorized according to their infection state, giving birth to the SEIR model (Susceptible-Exposed-Infectious-Recovered) [46]

$$\begin{aligned} \frac{dS}{dt} &= \mu(N - S) - \frac{\beta IS}{N} + \eta R \\ \frac{dE}{dt} &= \frac{\beta IS}{N} - (\mu + \sigma)E \\ \frac{dI}{dt} &= \sigma E - (\mu + \gamma)I \\ \frac{dR}{dt} &= \gamma I - (\mu + \eta)R \end{aligned} \tag{3.5}$$

- **The SEIQR Model:** The new SEIQR model is based on the (Susceptible - Infected - Recovered = (SIR)) compartmental models used to simulate disease transmission. This advanced model was created by adding a quarantine compartment and dividing the exposed population into two groups (namely essential services and average population). This difference is significant for the overall cases since the critical population gamma γ has a contact rate almost three times that of the typical population beta β. In the next part, will go through this in further depth. Because existing models lack a designated compartment to separate the quarantine (Q) population from the actual infected (I) people, the SEIQR model is essential. Furthermore, it is critical because COVID-19 is a disease unlike any other, with symptomatic and asymptomatic populations, each with dynamics and pandemic protection implications.Thus the model is given by the system of ODES [71, 76].

$$\begin{aligned}
\frac{dS}{dt} &= -\beta S(\omega E + I + Q)\\
\frac{dE}{dt} &= \beta S(\omega E + I + Q) - \varepsilon t E\\
\frac{dI}{dt} &= \varepsilon \tau E - \alpha(1-v)I - v\varphi I\\
\frac{dQ}{dt} &= v\varphi I - \alpha Q\\
\frac{dR}{dt} &= \alpha Q + \alpha(1-v)I
\end{aligned} \tag{3.6}$$

- **The SEIQLR Model:** It is possible to categorize a province in one of a total of six different ways. First, those exposed to the new coronavirus are classified as having; as a result, the daily numbers of suspicions, diagnoses and quarantined individuals are categorized as latent. Finally, as a group, patients who are healed and died are referred to as survivors [101].

$$\begin{aligned}
\frac{dS}{dt} &= -\beta S(\omega E + I + Q)\\
\frac{dE}{dt} &= \beta S(\omega E + I + Q) - \varepsilon \tau E\\
\frac{dI}{dt} &= \varepsilon \tau E - \alpha I(1-v) - v\varphi I\\
\frac{dQ}{dt} &= v\varphi I - \alpha Q\\
\frac{dL}{dt} &= (1-\varepsilon)\tau E - \eta L\\
\frac{dR}{dt} &= \alpha Q + \alpha I(1-v) + \eta L
\end{aligned} \tag{3.7}$$

The Parameters	The Distinct Meaning
μ	**The birth / death rate**
β	**The transmission rate**
$1/\nu$	**The average length of immunity**
$1/\sigma$	**The average latent period**
$1/\gamma$	**The average infectious time**
ω	**Infection- reducing factor in exposed infection**
α	**Removal rate for quarantine**
v	**Transfer rate of diagnosed cases, where in the SIR model, v is 1**
$1/\varphi$	**Average delayed reporting period**
η	**Removal rate for the late**

Table 3.1: Defining Parameters for SIR , SEIR , SEIQR , SEIQLR Models

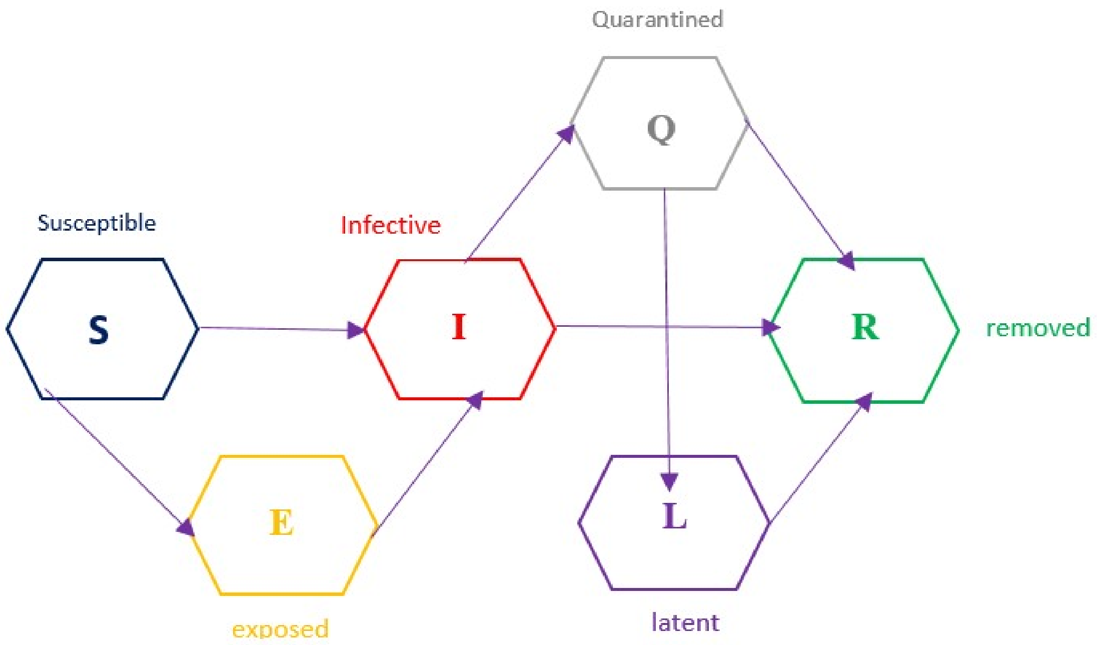


Figure 3.2: The SIR Model and The Models Developed from It Are Shown in This Diagram.

Categories	Explanations for each category
$S(t)$	**People who are possible to be infected by COVID-19**
$E(t)$	**People who are exposed to the COVID-19, but not diagnosed yet**
$I(t)$	**People who are diagnosed currently, but not quarantined**
$Q(t)$	**People who are diagnosed and quarantined**
$L(t)$	**People who are infected, but have no symptoms**
$R(t)$	**People who cured after infection and not be re-infected by COVID-19 again and people who died because COVID-19**

Table 3.2: Classification and Definition of The COVID-19-Infected Population

3.2 Coronavirus in Iraq

On February 24, 2020, in Najaf, Iraq, the 2020 coronavirus pandemic started when an Iranian religious student tested positive with SARS-2. More instances of covid-19 have been discovered, bringing the total number of verified cases in Iraq to 1,684,955, with 19,000 fatalities as of August 5, 2021. On March 4, 2020, Iraqi officials confirmed the country's first fatality from the novel coronavirus and the first death in the Arab world. Since the first virus case was discovered on February 24, an Iranian student visiting Najaf, the Ministry of Health, has announced that new topics are registered almost daily. The provinces where confirmed casualties were recorded are Baghdad, Najaf, Karbala, Babylon. Wasit, Sulaymaniyah, Kirkuk and Maysan. Since it emerged last December in China, the disease has spread to dozens of countries, infecting some 90,000 people and three thousand dead, mainly in China [3, 4, 6, 81]. The following Table 3.3 shows both the infected and deceased cases of COVID - 19 in Karbala province from January to November 2021.

The Health Organization's monthly data for Iraq, mainly the province of Karbala, are shown in Table No.3.3 ; see that the number of infected, removed, and dead fluctuates from month to month and does not follow a consistent pace. It can be seen more directly by Figure 3.3, where is observe that if the curve of the infected color red, removed green, black for dead people. It also offers us the month of November increased in cases of recovery and decrease in the cases of infected and dead.

Months	**Injectives**	**Removed**	**Death**
January	1486	792	13
February	7696	5349	46
March	6573	7861	65
April	6573	4975	33
May	3083	3664	19
June	5064	3936	14
July	14038	11541	48
August	12242	14114	104
September	2479	5017	48
October	525	606	8
November	102	129	2

Table 3.3: Monthly Information on The Number of Persons Infected, Removed, and Death by The coronavirus in The Karbala Governorate for 2021

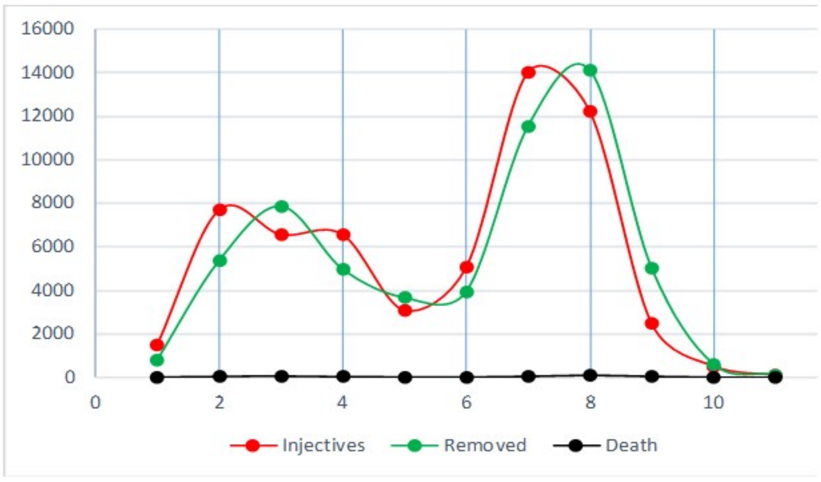


Figure 3.3: Grants for The victims of The Coronavirus in Karbala Governorate from 2021 to November of that year

Example 3.2.1. The following Python code integrates the equations 3.4 for a disease characterized by parameters $\beta = 0.2, 1/\ \gamma = 10$ days in a population of $N = 1000000$. The form begins with 102 infected individuals per day 0 : $I(0) = 102$ and $R(t) = 129$. The drawn curves $S(t), I(t)$ and $R(t)$ are designed to look a little prettier than the default settings of Matplotlib.

Code 1: Solving SIR

```
import numpy as np

from scipy.integrate import odeint

import matplotlib.pyplot as plt

N= 1 0 0 0 0 0 0

I0, R0=102,129

S0=N -I0-R0

\beta, \gamma =0.2, 1./10}

t= np.linspace (0,365,365)

def deriv(y, t, N, \beta, \gamma):

S, I, R=y
```

```
dSdt =- \beta * S * I / N

dIdt = \beta * S * I / N - \gamma* I

dRdt = \gamma * I

return dSdt, dIdt, dRdt

y0}= S0, I0, R0

ret = odeint(deriv, y0, t, args =(N, \beta, \gamma ))

S, I, R= ret.T

fig =plt.figure(facecolor='w'

ax = fig.add_subplot(111, facecolor='#dddddd', axisbelow=True)

ax.plot(t, S/1000000, 'b', \alpha =0.5, lw=2, label ='Susceptible')

ax.plot (t, I/1000000, 'r', \alpha =0.5, lw=2, label = 'Infected' )

ax.plot(t, R/1000000, 'g', \alpha =0.5,1 w=2, label = 'Recovered with immunity')

ax.set_xlabel('Time/days')

ax.set_ylabel('Number (1000000s)')

ax.set ylim(0,1.2)

ax.yaxis.set_tick_params(length =0)

ax.xaxis.set_tick params(length =0)

ax.grid b= True, which = 'major','c'= 'w', lw=2, ls='-')

legend = ax.legend()

legend.get_frame ().set_\alpha(0.5)

for spine in ('top', 'right', 'bottom', 'left'):

ax.spines[spine].set_visible(False)

plt.show()
```

The number of patients infected and recovered from the coronavirus for November was collected from Table 3.3. Python code was used to anticipate the number of infected and recovered within 365 days; Figure 3.4 illustrates the forecast precisely.

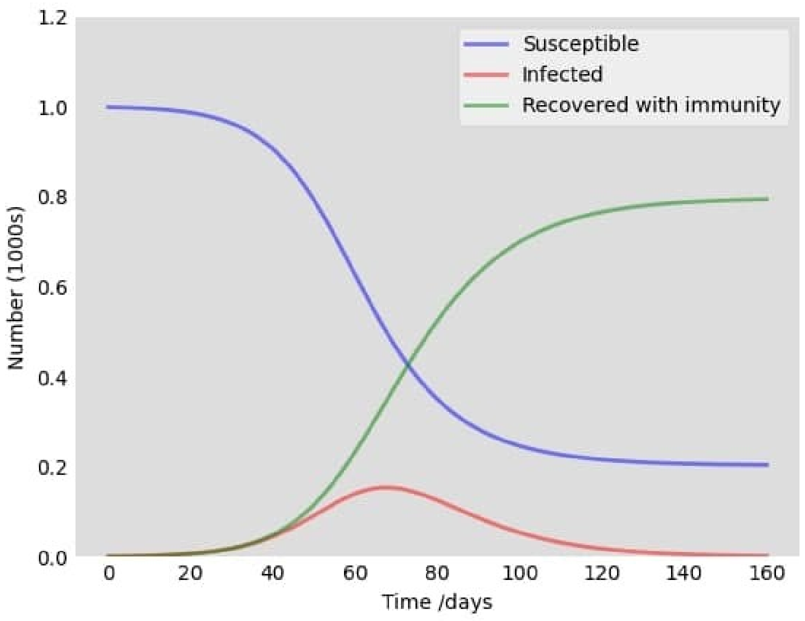


Figure 3.4: Describes An Increase and Decrease in The Numbers of Those Susceptible, Infected and Recovered within 365 days.

3.3 Numerical Solution for SIR Model by Using Euler Method

The SIR model can be solved numerically using the Euler method and remind Euler's method. If it have the slope formula means calculating the derivative $\frac{dy}{dt}$ at a certain point, (t, y) it will enable us to generate a series of y values: $y_0, y_1, y_2, \ldots$ Provided that there is a certain initial starting point y_0 and calculate each raise as slope χ run, that is :

$$y_n = y_n + \frac{dy_{n-1}}{dt}h \tag{3.8}$$

Such that, h in the temporal domain, is a suitable tiny step size. It makes no difference in the computation if the slope formula is dependent on variables other than t and y, such as χ and z, as long as which know-how χ and z are connected to t and y. May create values for χ and z in the same manner if they are other dependent variables in a system of differential equations. Of course, the dependent variable names for the SIR model should be S, I and R. As a result, will obtain three Euler formulae that are of the type:

$$\begin{aligned} S_n &= S_{n-1} + \frac{dS_{n-1}}{dt}h \\ I_n &= I_{n-1} + \frac{dI_{n-1}}{dt}h \\ R_n &= R_{n-1} + \frac{dR_{n-1}}{dt}h \end{aligned} \tag{3.9}$$

Given the equations of SIR 3.4, more precisely. Euler formulas become:

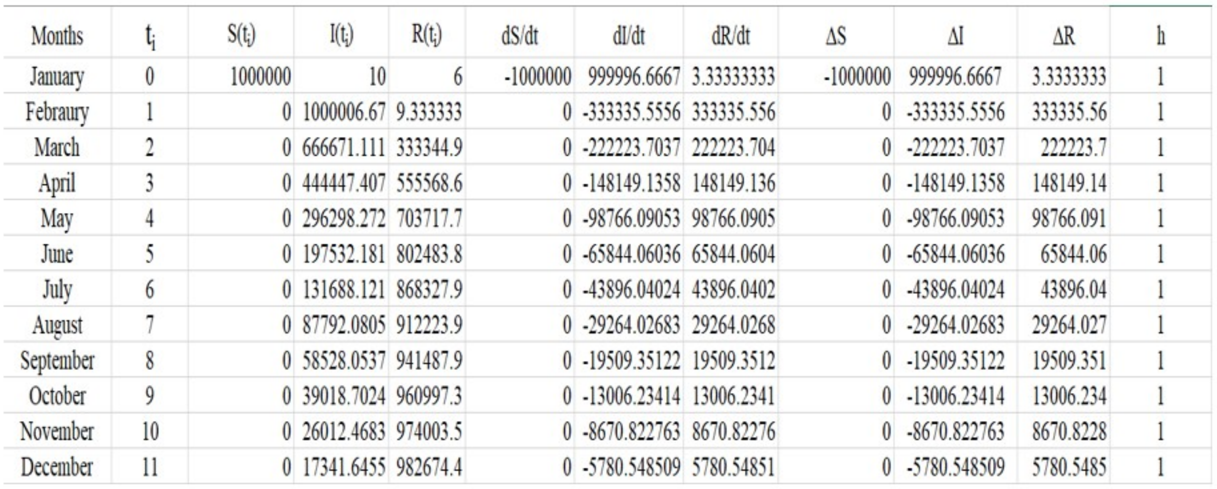
$$\begin{aligned} S_n &= S_{n-1} - \beta S_{n-1} I_{n-1} h \\ I_n &= I_{n-1} + \left(\beta S_{n-1} I_{n-1} - \gamma I_{n-1}\right) h \\ R_n &= R_{n-1} + \gamma I_{n-1} h \end{aligned} \tag{3.10}$$

Naturally, so must know the explicit values of $\beta, \gamma, S(o), I(0), R(0)$ and h calculate anything from these formulas. This section investigates whether these formulas are sufficient to obtain SIR model solutions. Will use Euler's method instead of creating a recipe from scratch, as shown in example below:

Example 3.3.1. Consider the initial value problem for the SIR model that is shown in Eq.3.4 is: $S(0) = 1000000, I(0) = 10, R(0) = 6$ and $h = 1$, using Euler formula 3.9 to find the expected Monthly data For people who are likely to be infected with the corona virus, who have been confirmed infected with the virus and who are completely recovered from it: Both the spread of COVID-19, using the Euler formula described by equations 3.9 or 3.10, where needed to determine the time difference and symbolized by the symbol h, as well as the difference for both those who are susceptible to infection and those who are infected and recovering from corona virus were found, look Table 3.4.

$$\begin{aligned} h &= h_n - h_{n-1} \\ \Delta S &= \frac{dS}{dt}h \\ \Delta I &= \frac{dI}{dt}h \\ \Delta R &= \frac{dR}{dt}h \end{aligned} \tag{3.11}$$

It's also worth noting from Table 3.4, which shows that in February, therefore see a total drop-off in exposure to the injury, and the number of those who have been infected and healed will continue to drop as well.



Months	t_i	$S(t_i)$	$I(t_i)$	$R(t_i)$	dS/dt	dI/dt	dR/dt	ΔS	ΔI	ΔR	h
January	0	1000000	10	6	-1000000	999996.6667	3.33333333	-1000000	999996.6667	3.3333333	1
Febraury	1	0	1000006.67	9.333333	0	-333335.5556	333335.556	0	-333335.5556	333335.56	1
March	2	0	666671.111	333344.9	0	-222223.7037	222223.704	0	-222223.7037	222223.7	1
April	3	0	444447.407	555568.6	0	-148149.1358	148149.136	0	-148149.1358	148149.14	1
May	4	0	296298.272	703717.7	0	-98766.09053	98766.0905	0	-98766.09053	98766.091	1
June	5	0	197532.181	802483.8	0	-65844.06036	65844.0604	0	-65844.06036	65844.06	1
July	6	0	131688.121	868327.9	0	-43896.04024	43896.0402	0	-43896.04024	43896.04	1
August	7	0	87792.0805	912223.9	0	-29264.02683	29264.0268	0	-29264.02683	29264.027	1
September	8	0	58528.0537	941487.9	0	-19509.35122	19509.3512	0	-19509.35122	19509.351	1
October	9	0	39018.7024	960997.3	0	-13006.23414	13006.2341	0	-13006.23414	13006.234	1
November	10	0	26012.4683	974003.5	0	-8670.822763	8670.82276	0	-8670.822763	8670.8228	1
December	11	0	17341.6455	982674.4	0	-5780.548509	5780.54851	0	-5780.548509	5780.5485	1

Table 3.4: The Data of Solving SIR by using Euler Method

Note 3.5(a): There are curves for cases of infection and recovery, where the red curve represents infected people, and the green represents those who have recovered from the epidemic, and can see from this that the curve of the recovering continues to increase, while the curve of infected continues to decrease. Additionally,3.5 (b) and (c) illustrated the SIR model's differential equation solution curves, as well as the variations in the number of coronavirus susceptible, infected, and removed persons, respectively

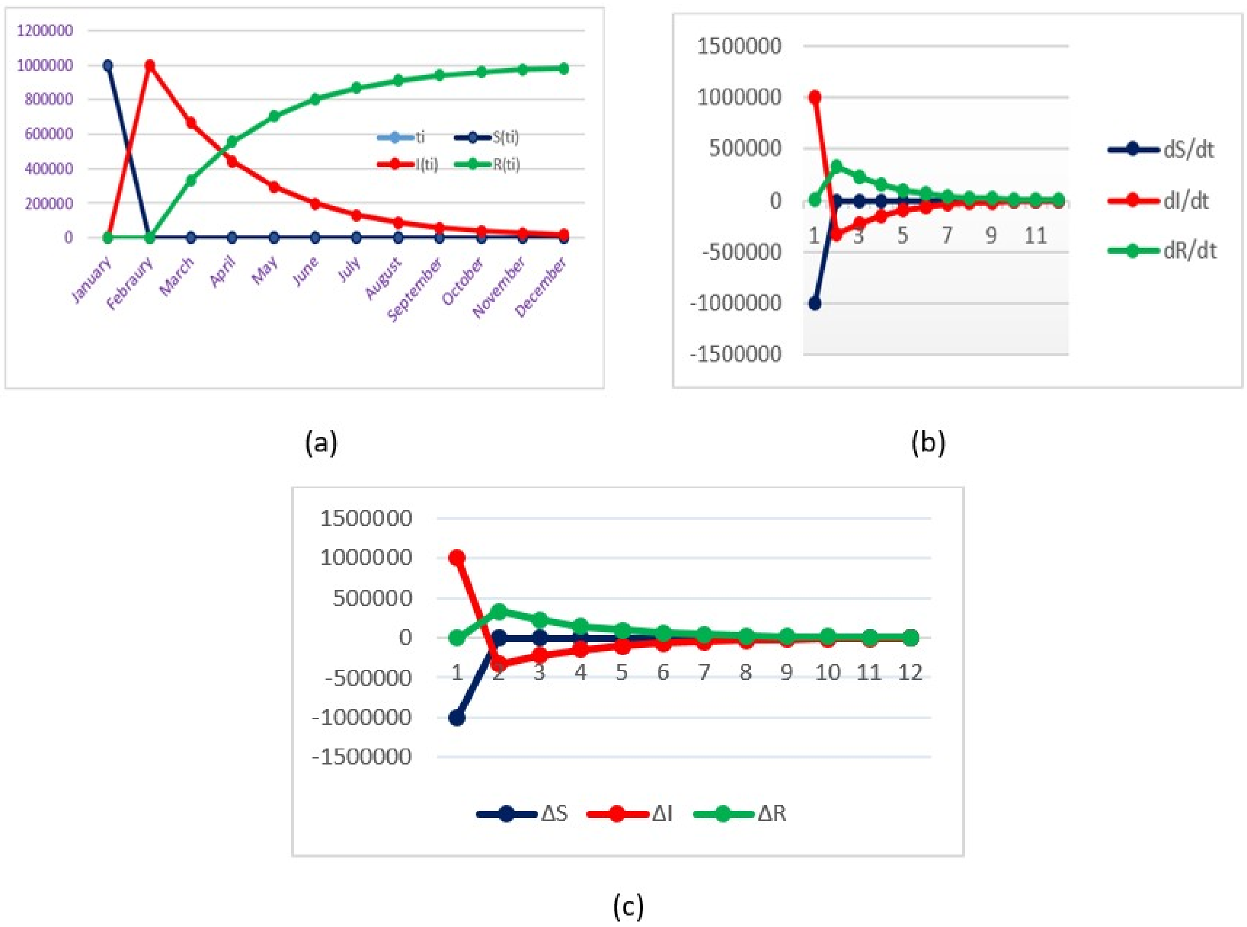


Figure 3.5: (a) Observed Serial Interval (Months). (b) The Solving of SIR. (c) Describes The Changes of S, I, R

3.4 The Bridge between Optimization and Coronavirus

COVID-19 should not be allowed to spread, and this section highlights the need of developing a mathematical control mechanism. COVID-19 infection from person to person and worldwide spread are both regulated. The proposed mathematical technique was created to fight COVID-19's worldwide spread. The suggested mathematically valid control mechanism optimizes the number of safe nations. The COVID-19 spread is: During coronavirus proliferation, N regions (countries) have four states:

1. It is considered that nations are safe.
2. While nations are regarded safe, COVID-19 infection is possible.
3. COVID-19 is a virus that may be spread internationally.
4. COVID-19 may infect a nation but not spread to others.

Because the states are under lockdown to prevent disease spread, adjacent regions do not need to assist. 2 and 3 may still get help since they are not under lockdown. To keep things simple, we'll ignore this assumption. State 2 countries may also get aid in finding and quarantining state three guests. The COVID-19 test for all departing travelers may help nations in condition 3. Quarantined and transported to state four. During the fast-spreading phase, coordinating COVID-19 propagation increases the number of countries in state one. It is a mixed binary-integer programming problem.

$$\max \sum_{k=1}^{N} \chi_1 t_k \tag{3.12}$$

subject to :

$$\chi_1 t_k = \chi_1 t_{k-1} + \sum_{k=1}^{N} C_i 2 t_{k-1} h_i t_{k-1} - \sum_{k=1}^{N} C_i 3 t_{k-1} S_i t_{k-1} \tag{3.13}$$

$$\chi_2 t_k = \sum_{k=1}^{N} C_i 3 t_{k-1} S_i t_{k-1} \tag{3.14}$$

$$\chi_3 t_k = \chi_3 t_{k-1} + \sum_{k=1}^{N} C_i 2 t_{k-1} \left(1 - h_i t_{k-1}\right) - \sum_{k=1}^{N} C_i 3 t_{k-1} h_i t_{k-1} \tag{3.15}$$

$$\chi_4 t_k = \chi_4 t_{k-1} + \sum_{k=1}^{N} C_i 3 t_{k-1} h_i t_{k-1} \tag{3.16}$$

$$\chi_1 t_1 = N - 1, \chi_2 t_1 = 0, \chi_3 t_1 = 1, \chi_4 t_1 = 0 \tag{3.17}$$

$$\sum_{j=1}^{4} \chi_j t_k = N, \text{ for each } k = 1, 2, \ldots, m \tag{3.18}$$

Where m is the period, and N how many nations are there :

$$0 \leq \chi_j t_k \leq N, C_{ij} t_k \in \{0, 1\}, e_{ij} t_k \geq 0, i = 1, 2, 3, \ldots, N, j = 1, 2, 3, 4, k = 1, 2, \ldots, m$$

The Variables	**Define Them or What They Represent**
$\chi_i t_i$	**Is the number of nationalities in a state j at t_k a given period.**
$C_{ij} t_k$	**Is 1 if i^{th} country be in state j at the stage t_k and its 0 , otherwise.**
$h_i t_k$	**Is 1 if i^{th} country at the stage t_k gets help otherwise its 0.**
$S_i t_k$	**Is the spreading rate country i at the stage t_k.**

Table 3.5: Constraints and Objective Function Variables are Explained

State 3 is the only state that can broadcast the COVID-19 to other countries for any k, implying that $S_1 t_k = S_2 t_k = S_4 t_k = 0$. Therefore, constraint 3.14 was also created as follows:

$$x_2 t_k = x_2 t_k + \sum_{k=1}^{N} C_i t_{k-1} 3 S_i + \sum_{k=1}^{N} C_i t_{k-1} t 2 h_i - \sum_{k=1}^{N} C_i t_{k-1} 2 \left(1 - h_i\right) \tag{3.19}$$

The calculation shows that the number of nations in state two decreases if they get help but increases if they do not 3.19. As a result, all nationalities in this state will be relocated. Also, nations affected by state three countries will move to state 2. The number of nations in state three is shown as a constraint here 3.15. The state three-country count is a hindrance 3.15. Except for states 3 and 4, which get help at this level, states 2 and 4 do not. Also, constraint 3.16 is defined by the number of nations in state 4 (closed case) at stage k, from states 4 and 3. Countries' positions during the commencement of a coronavirus pandemic are represented in constraint 3.17. According to constraint 3.17, the total number of nations must match the number of countries in all states. Figure3.6 shows the COVID-19 spreading control state transition diagram

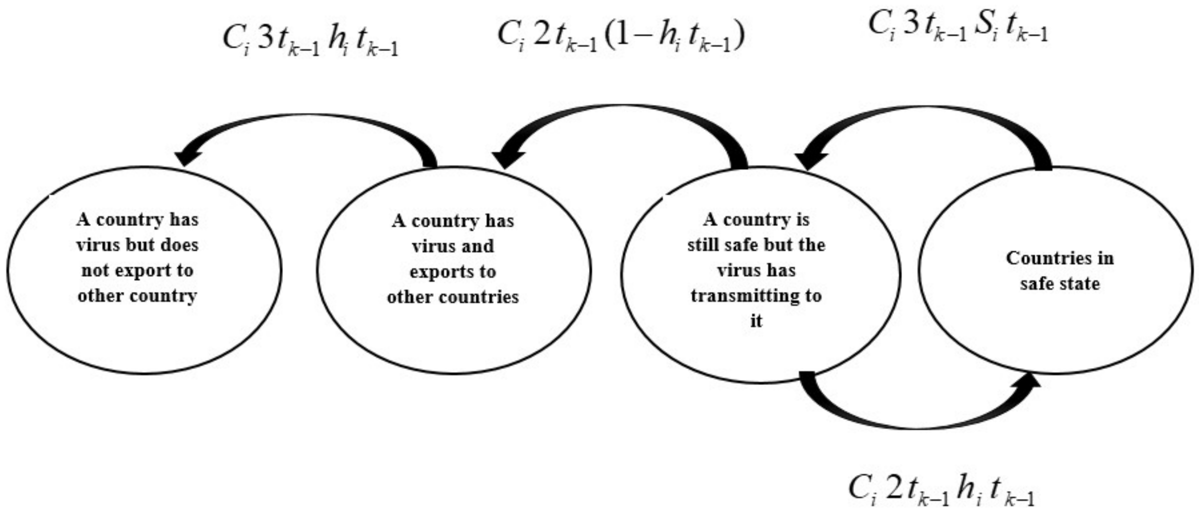

$C_i 3t_{k-1} h_i t_{k-1}$
$C_i 2t_{k-1}(1-h_i t_{k-1})$
$C_i 3t_{k-1} S_i t_{k-1}$
A country has virus but does not export to other country
A country has virus and exports to other countries
A country is still safe but the virus has transmitting to it
Countries in safe state
$C_i 2t_{k-1} h_i t_{k-1}$

Figure 3.6: COVID-19 Spreading Control State Transition Diagram.

CHAPTER 4

DYNAMIC OPTIMIZATION AND OPTIMAL CONTROL OF SYSTEMS

Introduction

This chapter discusses dynamic continuous unconstrained, and constrained optimization problems, including single objective functions. The most extensively widely used and renowned benchmark generators in this subject are based on the notion of a landscape made up of multiple components. The width, height, and location of these components change over time in most published DOP benchmarks.

4.1 Dynamics Optimization Modeling

The past two decades have seen a marked growth in interest in dynamic modeling and optimization of chemical processes. Control and scheduling of batch operations are often encountered difficulties, as are startup, upset, shutdown, and transient analysis, as well as safety studies and control scheme assessment. Dynamical equations are used to model chemical reactions (DAEs). The DAE formulation comprises of differential equations that describe the system's dynamic behavior, such as mass and energy balances, and algebraic equations that establish physical and thermodynamic relationships [11, 25, 55]. The following is a representation of the dynamic optimization model:

$$\begin{aligned} \text{minimize} \quad & J(\chi, y, u) \\ \text{subject to} \quad & f\left(\frac{d\chi}{dt}, \chi, y, u\right) = 0 \\ & 0 \leq g\left(\frac{d\chi}{dt}, \chi, y, u\right) \end{aligned} \tag{4.1}$$

Note that $J(\chi, y, u)$ is the objective function of a dynamic optimization system; where u input values, y output values, and χ are the system.

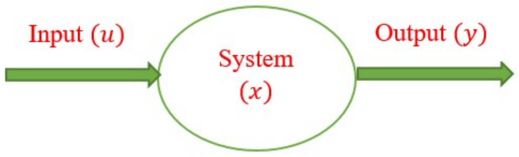


Figure 4.1: Describe the Input, Output, and System

f, g are constraints of a dynamic optimization system where these constraints are either equality f or inequality g, and the constraints contain differential equations $\frac{d\chi}{dt}$ and u its control variables with respect to time. [66]. .

4.2 Python for Dynamic Optimization

Python is a programming language used to aid with dynamic simulation, estimation, and control solutions. Numerous tools for modeling and optimizing dynamic systems are available. Additionally, there are other things to consider while picking the best instrument for a specific activity. For example, many factors should be considered when choosing an optimization tool for dynamic scalings, such as size (scaling with number equations and degrees of freedom), speed (use in real-time estimation or control), adaptability, extensibility, and support (commercial vs. open-source community) (ability to embed on a system or communicate with a particular platform) [47].

4.2.1 GEKKO

Combines machine learning and optimization strategies to solve complex mixed-integer and differential algebraic problems. It is used with solvers for large-scale linear, quadratic, nonlinear, and mixed-integer programming (LP, QP, NLP, MILP, MINLP). All types of operation are available, including parametric regression, data reconciliation, real-time optimization, dynamic simulation, and nonlinear predictive control. GEKKO is a Python object-oriented library for executing AP Monitor on a local machine [13].

4.3 Non-Linear of Dynamic Optimization Problem

The discretization of a continuous time representation enables large-scale nonlinear programming (NLP) solvers to locate solutions at defined time intervals in a time horizon. Numerous terms and associated strategies exist for establishing mathematical links between derivatives and non-derivative values. Several terms relevant to this discussion include orthogonal collocation, direct transcription, Gaussian pseudospectral method, Gaussian quadrature, Lobatto quadrature, Radau collocation, Legendre polynomials, Chebyshev polynomials, Jacobi polynomials, Laguerre polynomials, and kkt conditions.

4.3.1 The Karush Kuhn Tucker (KKT) Conditions

The Karush Kuhn Tucker (KKT) Criteria are the conditions that must be met in order for a constrained local optimum to exist. These conditions are very significant in constrained optimization theory and algorithm development.
For optimum primal χ and dual λ variables, there exist four KKT criteria. Values marked with an asterisk $(*)$ are the best possible ones.

1. Feasible Constraints
2. No Feasible Descent
3. Complementary Slackness
4. Positive Lagrange Multipliers

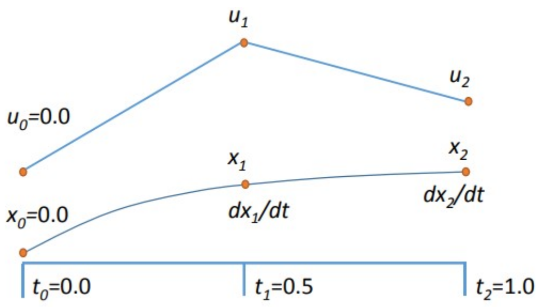
The feasibility condition (1) states that the limits on equality and inequality cannot be broken under ideal circumstances. The gradient condition (2) assures that there is no possible path that might possibly enhance the objective function. These two requirements (3 and 4) only apply to inequality constraints and impose a positive Lagrange multiplier while the constraint is active ($= 0$) and a zero multiplier when the constraint is inactive (> 0).

Example 4.3.1. Solving Dynamic Optimization Problem with Orthogonal Collocation on Finite Elements by Using KKT Conditions:

Optimize a differential equation-dependent objective function. To minimize the objective function, use orthogonal collocation to discretize the differential equation and the criteria of optimality.

$$\begin{aligned} \min_{u} \quad & (\chi - 5)^2 \\ \text{subject to} \quad & 2\frac{d\chi}{dt} = -\chi + 5u \end{aligned} \tag{4.2}$$

The above coupled differential equation must be solved from the beginning of time to the end of time. Use orthogonal collocation with three nodes (time points) at $t = [0, 0.5, 1.0]$ to solve the system of equations.



The starting condition, χ_0, is set to zero, as is the initial value of the controlled variable, u_0 (although, this initial condition has no effect on the solution). The following approximations are possible for the derivative values at χ_i, where i is the collocation point t_0, t_1, or t_2.

$$t_2 \begin{bmatrix} 0.75 & -0.25 \\ 1.00 & 0.00 \end{bmatrix} \begin{bmatrix} \frac{d\chi_1}{dt} \\ \frac{d\chi_2}{dt} \end{bmatrix} = \begin{bmatrix} \chi_1 \\ \chi_2 \end{bmatrix} - \begin{bmatrix} \chi_0 \\ \chi_0 \end{bmatrix} \tag{4.3}$$

By differentiating the objective function with respect to χ_1 and χ_2 and then setting those expressions to zero, the last two equations required to meet the optimality criteria (Karush Kuhn-Tucker conditions) are obtained.

Report the solution of u_1,u_2, χ_1, and χ_2 as well as the derivative values (χ_1 and χ_2) at $t_1 = 0.5$ and $t_2 = 1.0$ (6 total values). Demonstrate work using a six-variable system $(u_1, u_2, \chi_1, \chi_2, \frac{d\chi_1}{dt}, \frac{d\chi_2}{dt})$ and six equations. By using python code, so that getting:

Code 2: Karush Kuhn-Tucker Conditions

```
import numpy as np
from gekko import GEKKO
# Check solution with GEKKO
m = GEKKO()
m.time=[0,1]
u = m.Var(value=1)
x = m.Var(value=0)
m.Obj((x-3)**2)
m.Equation(5*x.dt()==-x+2*u)
m.options.IMODE = 6
m.options.NODES = 3
m.solve()
# min (x-3)^2
# s.t. 5 * dx/dt = -x + 2 * u
# 5 * xdot1 = -x1 + 2*u1

# 5 * xdot2 = -x2 + 2*u2

# t2 * 0.75 * xdot1 - t2 * 0.25 * xdot2 = x1 - x0

# t2 * 1.00 * xdot1 = x2 - x0

# 2 * (x1-3) = 0

# 2 * (x2-3) = 0

# Rearrange to put all variables on left-hand side

# 5 * xdot1 + x1 - 2*u1 = 0

# 5 * xdot2 + x2 - 2*u2 = 0

# t2 * 0.75 * xdot1 - t2 * 0.25 * xdot2 - x1 = -x0

# t2 * 1.00 * xdot1 - x2 = -x0

# 2 * x1 = 6

# 2 * x2 = 6

# Set-up and solve A y = b

# y = [xdot1 xdot2 x1 x2 u1 u2]

# Matrix A

A = np.array([[5,0,1,0,-2,0],\

[0,5,0,1,0,-2],\

[0.75,-0.25,-1,0,0,0],\

[1,0,0,-1,0,0],\

[0,0,2,0,0,0],\

[0,0,0,2,0,0]])

# Column vector b

b = np.array([0,0,0,0,6,6])

# Solve A y = b as y = A^-1 * b

ymat = np.linalg.solve(A,b)

print('Variables ')

print(['u1 = ' + str(ymat[4])])
print(['u2 = ' + str(ymat[5]) + \
' (Matrix) vs ' + str(u.value[-1]) + ' (GEKKO)'])
print(['x1 = ' + str(ymat[2])])
print(['x2 = ' + str(ymat[3]) + \
' (Matrix) vs ' + str(x.value[-1]) + ' (GEKKO)'])
print(' ')
print('Derivatives ')
print(['d(x11)/dt = ' + str(ymat[0])])
print(['d(x21)/dt = ' + str(ymat[1])])
```

When the code is performed, the following output will show the result of KKT dynamic optimization for the problem in example 4.3.1 after running the code of python 4.3.1, therefore get:

Note the number of iterations is 2 and also:

Solution time : $4.899999999906868E - 003$ sec

Objective : $7.888609052210118E - 031$

Variables:

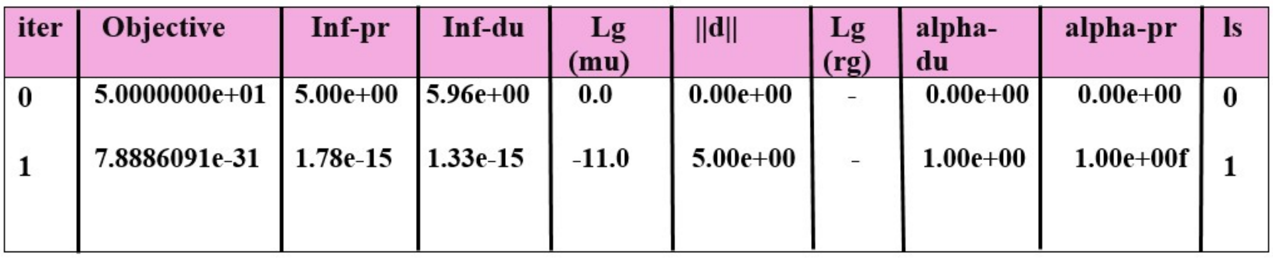
$u_1 = 3.0$, $\quad u_2 = -1.0$

$\chi_1 = 5.0$, $\quad \chi_2 = 5.0$

Derivatives :

$$\frac{d(\chi 11)}{dt} = 5.0 \ , \quad \frac{d(\chi 21)}{dt} = -5.0$$

The final value of the objective function is: $7.888609052210118E - 031$



iter	Objective	Inf-pr	Inf-du	Lg (mu)	||d||	Lg (rg)	alpha-du	alpha-pr	ls
0	5.0000000e+01	5.00e+00	5.96e+00	0.0	0.00e+00	-	0.00e+00	0.00e+00	0
1	7.8886091e-31	1.78e-15	1.33e-15	-11.0	5.00e+00	-	1.00e+00	1.00e+00f	1

Table 4.1: Numerical Results of 4.3.1

4.4 Algorithms of Dynamic Optimization and Optimal Control Systems

Any computer procedure that receives an input value or set of values and outputs an output value or set of values is a formula. Thus, an algorithm is a sequence of computations that alter the data state. Additionally, an algorithm may be seen as a tool to solve a well-defined computing issue. The issue statement provides a comprehensive overview of the intended input/output relationship. The algorithm defines the computational technique for creating the suitable input (output) connection. For instance, may need to ascend a series of integers. This is a common occurrence in reality and gives an excellent opportunity to introduce a variety of conventional design methodologies and analytical tools [28].

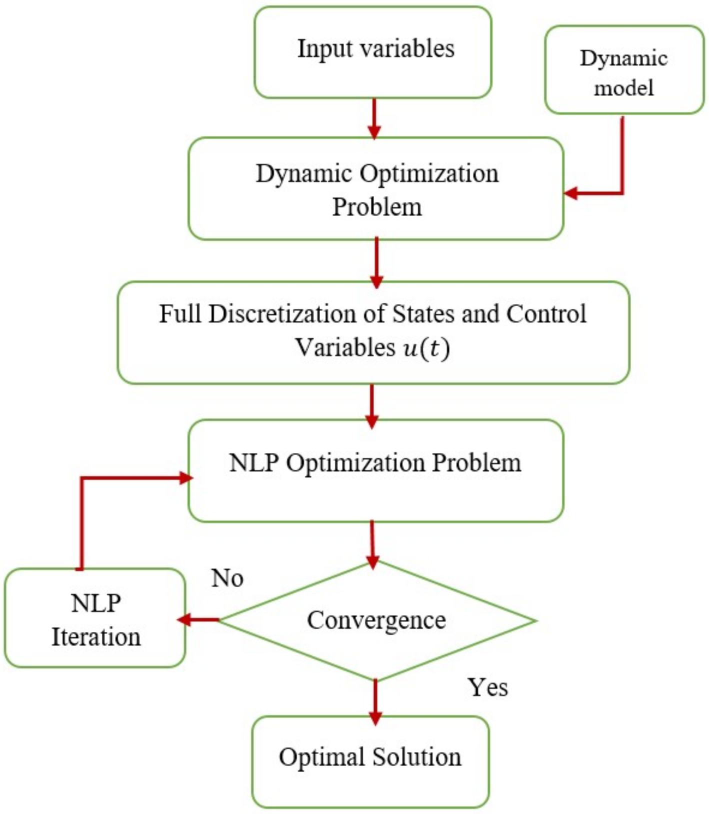


Figure 4.2: Dynamic Optimization Algorithm

4.5 Interior-Point Methods

With increasing interest in efficient optimization approaches, interior-point or barrier methods for large-scale nonlinear programming have been developed. These approaches, in particular, provide an appealing alternative to active set strategies for situations involving a high number of inequality constraints. Additionally, there has been increased knowledge of the convergence qualities of interior-point techniques during the last 15 years , and efficient algorithms with desired global and local convergence properties have been devised. To enable convergence from suboptimal beginning locations, interior-point approaches have been developed in both the trust region and line-search frameworks, which utilize precise penalty merit functions to compel movement toward the solution [94]. Fletcher and Leyffer, on the other hand, have presented filter approaches as an alternative to merit functions for ensuring global convergence in nonlinear programming algorithms. The essential principle is that trial points are allowed provided they enhance either the objective function or constraint violation, rather than a combination of the two metrics specified by a merit function. This filter methodology has been extended in a variety of ways to barrier approaches in recent years. M. Ulbrich, S. Ulbrich, and Vicente discuss a trust area filtering approach that accepts trial steps based on the optimality conditions' norm. Additionally, Benson, Shanno, and Vanderbei suggested many heuristics based on the concept of filter techniques, claiming that they are more efficient than their earlier merit function approach, however no convergence study is included. Finally, analyzes the worldwide convergence of an interior-point algorithm combined with filter line search. The assumptions provided are less stringent than those established in previously suggested line-search interior-point approaches for nonlinear programming. Numerous interior-point methods have been implemented in robust software codes and numerical tests have demonstrated their efficacy and robustness in practice [94].

Consider the use of a primal-dual barrier approach to handle nonlinear optimization problems of the following kind.

$$\begin{aligned} \min_{\chi \in \mathbb{R}^n} \quad & f(\chi) \\ \text{s.t.} \quad & g(\chi) \leq 0 \\ & \chi_L \leq \chi \leq \chi_U \end{aligned} \tag{4.4}$$

where $\chi_L \in [-\infty, \infty)^n$ and $\chi_U \in (-\infty, \infty]^n$, with $\chi_L^{(i)} \leq \chi_U^{(i)}$, are the lower and upper bounds on the variables χ.

Definition 4.5.1. A function $B : \mathbb{R}^n \to \mathbb{R}$ is called an interior penalty (Barrier) function if $B(\chi)$ satisfies:

1. $B(\chi)$ is continuous on $\mathbb{R}^n$
2. $B(\chi) \geq 0$ each $\chi \in \text{int } S$, where S denotes the feasible region for Problem 4.4.
3. $B(\chi) \to +\infty$ as χ approaches the border of S

Then can write the barrier problem as:

1. Inverse function: $B(\chi) = -\sum_{i=1}^{M} \frac{1}{g_i(\chi)}$, for $g_i(\chi) < 0$
2. Logarithm function: $B(\chi) = -\sum_{i=1}^{M} \log\left[-g_i(\chi)\right]$, for $\quad g_i(\chi) < 0$

The most commonly used types of interior penalty functions are:

$$\begin{cases} \phi(\chi,\mu) = \text{ minimize } \quad f(\chi) + \mu B(\chi) \\ \\ \text{subject to} \quad g_i(\chi) = 0, \quad i = 1, 2, \ldots, M \\ \\ \chi \in \mathbb{R}^n \end{cases}$$

4.6 The Basics of Interior Point Algorithm

The suggested method is motivated by the parts that follow:

4.6.1 The Primal-Dual Barrier Approach

To simplify notation, we discuss the problem formulation approach first.

$$\begin{cases} \min\limits_{\chi \in \mathbb{R}^n} & f(\chi) \\ \text{s.t.} & g(\chi) = 0 \\ & \chi \geq 0. \end{cases} \tag{4.5}$$

The suggested technique computes (approximate) answers to a series of barrier problems as a barrier approach.

$$\begin{aligned} &\min_{\chi \in \mathbb{R}^n} \phi(\chi,\mu) := f(\chi) - \mu \sum_{i=1}^{n} \ln\left(\chi_{(i)}\right) \\ &\text{s.t.} \quad g(\chi) = 0 \end{aligned} \tag{4.6}$$

For a series of barrier parameters μ that decrease and eventually reach zero. This is equivalent to using the homotopy approach to solve the primal-dual equations,

$$\begin{aligned} \nabla f(\chi) + \nabla g(\chi)\lambda - s = 0 \\ g(\chi) = 0 \\ \chi S e - \mu e = 0 \end{aligned} \tag{4.7}$$

using the homotopy parameter μ , which has been driven to have a value of zero. see that , $\lambda \in \mathbb{R}^m$ and $s \in \mathbb{R}^n$ represent the Lagrangian multipliers for the equality and bound constraints, respectively.

The optimality error for the barrier issue is defined as using the components of the primal-dual equations 4.7.

$$E_\mu(\chi, \lambda, s) := \max\left\{ \frac{\|\nabla f(\chi) + \nabla g(\chi)\lambda - s\|_\infty}{z_d}, \|g(x)\|_\infty, \frac{\|\chi S e - \mu e\|_\infty}{z_c} \right\} \tag{4.8}$$

where, scaling parameters $z_d, z_c \geq 1$, If an approximate solution is found (χ^*, λ^*, s^*), the entire procedure is terminated(incorporating multiplier estimates) , so satisfying:

$$E_0(\chi^*, \lambda^*, s^*) \leq \epsilon_{tol}$$

where $\epsilon_{tol} > 0$ is the user-specified error tolerance , then select the scaling factors to modify the termination criterion to such instances.

$$s_d = \max\left\{ z_{\max}, \frac{\|\lambda\|_1 + \|s\|_1}{(m+n)} \right\} / s_{\max} \quad , \; z_c = \max\left\{ z_{\max}, \frac{\|s\|_1}{n} \right\} / z_{\max} \tag{4.9}$$

In this method, anytime the average value of the multipliers exceeds a predefined number $z_{\max} \geq 1$, a component of the optimality error is scaled. It is also important to note that in the event that the multipliers diverge, the value of $E_0(\chi, \lambda, s)$ can only become tiny if a Fritz-John point is approached or if the primary variables also diverge. then use the technique described by Byrd, Liu, and Nocedal to obtain rapid local convergence, which has been shown to produce superlinear convergence under typical second order adequate conditions. Denoting the iteration counter for the "outer loop" with(j), and demand that the approximate solution $(\chi^*_{j+1}, \lambda^*_{j+1}, s^*_{j+1})$ of the barrier issue meets the tolerance for a given value of μ_j.

$$E_{\mu_j}(\chi^*_{j+1}, \lambda^*_{j+1}, s^*_{j+1}) \leq k_\epsilon \mu_j$$

before moving on to solve the next barrier problem, the method waits for a constant k_ϵ that is greater than 0. The fresh barrier metric is derived from

$$\mu_{j+1} = \max\left\{\frac{\epsilon_{\text{tol}}}{10}, \min\left\{\kappa_\mu \mu_j, \mu_j^{\theta_\mu}\right\}\right\} \tag{4.10}$$

Such that , $k_\mu \in (0,1)$ and $\theta_\mu \in (1,2)$. The barrier parameter progressively decreases in this manner at a superlinear pace. As opposed to that, the update rule does not allow μ to shrink beyond what is required given the intended tolerance ϵ_{tol}, preventing numerical issues at the conclusion of the optimization process. Additionally, so select a "fraction-to-the-boundary" option for subsequent use.

$$\tau_j = \max\{\tau_{\min}, 1 - \mu_j\}$$

where $\tau_{\min} \in (0,1)$ is its minimum value [45, 94].

4.6.2 A Line-Search Filter Method

Filter methods were originally proposed by Fletcher and Leyffer. In the context of solving the barrier problem for μ_j, the basic idea behind this approach is to interpret as a ϕ-objective optimization problem with the two goals of minimizing the objective function $\phi(\chi, \mu)$ and the constraint violation $\theta(\chi) := \|g(\chi)\|$ (with a certain emphasis on the latter quantity). Following this paradigm, we might consider a trial point $\chi_k(\alpha_{k,l}) := \chi_k + \alpha_{k,l} d^\chi_k$

during the backtracking line search to be acceptable, if it leads to sufficient progress toward either goal compared to the current iterate, i.e., if

$$\theta\left(\chi_k\left(\alpha_{k,l}\right)\right) \leq \left(1-\gamma_\theta\right)\theta\left(\chi_k\right) \tag{4.11}$$

or

$$\phi_{\mu_j}\left(\chi_k\left(\alpha_{k,l}\right)\right) \leq \varphi_{\mu_j}\left(\chi_k\right) - \gamma_\phi\theta\left(\chi_k\right) \tag{4.12}$$

Applies to fixed constants $\gamma_\theta, \gamma_{phi} \in (0,1)$, However, when for the present iteration we have sufficient progress in the barrier objective function, the aforementioned criteria is substituted $\theta\left(\chi_k\right) \leq \theta^{\min}$, for $\theta^{\min} \in (0,\infty]$ as well as the subsequent "switching condition"

$$\nabla\phi_{\mu_j}\left(\chi_k\right)^T d_k^\chi < 0 \quad \text{and} \quad \alpha_{k,l}\left[-\nabla\phi_{\mu_j}\left(\chi_k\right)^T d_k^\chi\right]^{s_\phi} > \delta\left[\theta\left(\chi_k\right)\right]^{s_\theta} \tag{4.13}$$

with $\delta > 0, s_\theta > 1, s_\phi \geq 1$ holds. If $\theta\left(\chi_k\right) \leq \theta^{\min}$ and it is true for the current step size $\alpha_{k,l}$, the trial point has to satisfy the Armijo condition

$$\phi_{\mu_j}\left(\chi_k\left(\alpha_{k,l}\right)\right) \leq \phi_{\mu_j}\left(\chi_k\right) + \zeta_\phi\alpha_{k,l}\nabla\phi_{\mu_j}\left(\chi_k\right)^T d_k^\chi \tag{4.14}$$

therefore, $\zeta \in (0, \frac{1}{2})$. The algorithm also keeps a "filter," which is a collection of rules $\mathcal{F}_k \subseteq \left\{(\theta,\phi) \in \mathbb{R}^2 : \theta \geq 0\right\}$ for each iteration k. The filter $\mathcal{F}_k$ includes those set combinations of constraint violation values θ and the objective function values ϕ, that must not be used for a successful trial point during iteration k : A trial point was used in the line search. $\chi_k\left(\alpha_{k,l}\right)$ is rejected, if $\left(\theta\left(\chi_k\left(\alpha_{k,l}\right)\right), \phi_{\mu_j}\left(\chi_k\left(\alpha_{k,l}\right)\right)\right) \in \mathcal{F}_k$. The trial point is not acceptable to the existing filter, we say next. The filter is first initialized to during the optimization to

$$\mathcal{F}_0 := \left\{(\theta,\phi) \in \mathbb{R}^2 : \theta \geq \theta^{\max}\right\} \tag{4.15}$$

for some $\theta^{\max}$, In order to ensure that the algorithm would never accept trial points with a constraint violation greater than $\theta^{\max}$. The update formula is used later to modify the filter

$$\mathcal{F}_{k+1} := \mathcal{F}_k \cup \left\{(\theta, \phi) \in \mathbb{R}^2 : \theta \geq (1-\gamma_\theta)\,\theta\left(\chi_k\right), \quad \phi \geq \phi_{\mu_j}\left(\chi_k\right) - \gamma_\phi \theta\left(\chi_k\right)\right\} \tag{4.16}$$

Following each iteration, in case the Armijo requirement is not met or the approved trial step size does not fulfill the switching condition. This makes sure the iterates can't go back to the area around χ_k. However, the filter does not change if both of them hold for the agreed-upon step size. Overall, this method makes sure that the algorithm cannot cycle, for instance, between two places that alternately reduce the constraint violation and the barrier objective function. Finally, in some circumstances, it may not be feasible to locate a trial step size $\alpha_{k,l}$ that meets the aforementioned requirements. With the help of linear models of the relevant functions, we approximate the minimal desired step size. As a result, we define

$$\alpha_k^{\min} := \gamma_\alpha \begin{cases} \min\left\{\gamma_\theta, \dfrac{\gamma_\phi \theta\left(\chi_k\right)}{-\nabla\phi_{\mu_j}\left(\chi_k\right)^T d_k^\chi}, \dfrac{\delta\left[\theta\left(\chi_k\right)\right]^s \theta}{\left[-\nabla\phi_{\mu_j}\left(\chi_k\right)^T d_k^\chi\right]^{s\phi}}\right\} \\ \text{ if } \nabla\phi_{\mu_j}\left(\chi_k\right)^T d_k^\chi < 0 \text{ and } \theta\left(\chi_k\right) \leq \theta^{\min} \\ \min\left\{\gamma_\theta, \dfrac{\gamma_\phi \theta\left(\chi_k\right)}{-\nabla\phi_{\mu_j}\left(\chi_k\right)^T d_k^\chi}\right\} \\ \text{if } \nabla\phi_{\mu_j}\left(\chi_k\right)^T d_k^\chi < 0 \text{ and } \theta\left(\chi_k\right) > \theta^{\min} \\ \gamma_\theta\ , \ \text{ otherwise} \end{cases} \tag{4.17}$$

possessing a "safety factor" $\gamma_\alpha \in (0, 1]$. If the backtracking line search encounters a trial step size with a $\alpha_{k,l} \leq \alpha_k^{\min}$, specified value, the search is terminated , reverting to a feasibility restoration phase, the algorithm. The algorithm is attempting to locate a new iteration here $\chi_{k+1} > 0$ which the present filter can accept and then holds, by using an iterative strategy to decrease the constraint violation. It is important to emphasize that the restoration phase algorithm might not be able to provide a new iteration for the filter line-search technique in some circumstances, such as when the problem cannot be solved. In this specific situation, an appropriate restoration phase method should converge to a local minimizer (or at least a stationary point) for the constraint violation. This should signal to the user that the issue seems to be infeasible (at least locally).

It is sufficient to guarantee global convergence for each barrier parameter with a fixed value μl in order to guarantee global convergence for the entire procedure. As a result, anytime μl is lowered, the filter $\mathcal{F}_k$ is reset to its original definition. However, in our experience, the reinitialization consistently yields good results. It may be feasible to reset the filter in ways that take into account information from the prior barrier problem [45,94]

.

4.6.3 Gray Wolf Optimization (GWO) Algorithm

In 2014, Mirjalili introduced the Gray Wolf Optimization (GWO) algorithm, one of the contemporary meta-optimization techniques. Mirjalili took inspiration from social structure and gray wolf hunting strategies. According to animal biologists, gray wolves live in packs of 5 to 12 individuals with a rigid social structure. Figure 4.3 and Table 4.2 illustrate the hierarchy of gray wolves and algorithm-inspired solutions. Leader wolves α (First best solutions) are accountable for herd choices, and the rest of the pack must carry them out. The second level in the social hierarchy of gray wolves is β ; (Second best solutions) its function is to aid in decision-making and they are responsible for disciplining the pack; the ideal replacement when one of them becomes crippled or dies. The third level isδ ; (Third best solutions) and the fourth level is ω ; (Worst solutions) must follow all wolves, and their tasks include protecting the herd and alerting it to danger, as well as caring for the old and hunters. In groups, members confirm the alpha's choice by lowering their tails [23, 35, 80].

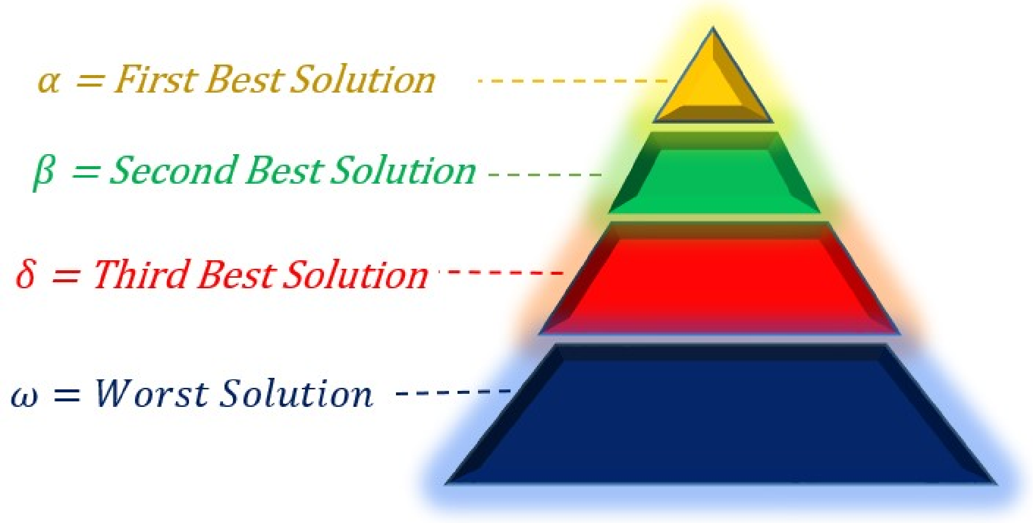

α = First Best Solution
β = Second Best Solution
δ = Third Best Solution
ω = Worst Solution

Figure 4.3: The Social Hierarchy of Gray Wolves

Level	Name	Roles in Herd	Type of Solutions
1	α	Leaders: making herd decisions	First best solutions
2	β	Help α, responsible for disciplining the herd	Second best solutions
3	δ	Protect the herd and alert it in case of danger	Third best solutions
4	ω	Plays the role of a scapegoat and must always submit to all other dominant wolves	Worst solutions

Table 4.2: The Social Hierarchy of Gray Wolves

Collective hunting is one of the behaviors of the gray wolves herd in addition to the social hierarchy. Gray wolves' hunting includes the following three main parts:

1. Tracking, chasing, and approaching the prey.
2. Pursuing, encircling, and harassing the prey till it stops moving.
3. Attacking the prey

4.6.4 Mathematical Model of the GWO Algorithm:

The GWO algorithm is a collection of principles inspired by social hierarchy and the intelligent hunting strategy of gray wolves. The behavior of gray wolves in pursuit of prey, as described by Mirjalili, includes low rest levels and being the last to feed. In groups, members confirm the alpha's choice by lowering their tails.

$$C = |B \cdot \chi_p(n) - \chi(n)| \tag{4.18}$$

$$\chi(n+1) = \chi_p(n) - A \cdot C \tag{4.19}$$

Here n is an iteration number, χ_p is a vector of the prey's positions, χ is a vector of gray wolf's positions, C is a calculated vector used to specify a new position of the gray wolf, while A and B are coefficient vectors can be can be calculated by

$$A = 2a \cdot r_i - a \tag{4.20}$$

$$B = 2r_i \tag{4.21}$$

Here a is a vector set to decrease linearly can found by

$$a = (1 - \frac{\text{iteration}}{\text{number of iterations}}) \tag{4.22}$$

Also r_i are random vectors in [0 1]. Any gray wolf can be placed in every random position around the prey (the best solution) as it is calculated from the equations (4.18) and (4.19), gray wolves have the ability to distinguish the location of prey from others. To emulation this hunting behavior, suppose that the alpha (the best candidate for the solution) while the beta and delta have more knowledge about the location of prey. Therefore, the algorithm saves three best achieved solutions away and forcing omega to update their locations to achieve the best place in the decision space. In the optimization algorithm, such a hunting behavior can be modeled by:

$$C_\tau = |B_i \cdot \chi_\tau - \chi| \tag{4.23}$$

$$\chi_i = \chi_\tau - A_i \cdot C_\tau \tag{4.24}$$

Here $\tau = \alpha, \beta, \delta$ in conjunction with $i = 1, 2, 3$. To understand how the GWO algorithm solves optimization problems theatrically, some notes can be summarized as follows:

- ❆ The social hierarchy helps the algorithm to rank solutions and save the best ones until the last iteration.
- ❆ The random vectors (A and B) help gray wolves (candidate solutions) to define different hyper-spheres with random radii.
- ❆ The hunting approach implemented in the GWO algorithm allows gray wolves (candidate solutions) to locate the probable position of the prey (optimal solution).
- ❆ When reducing the values of A, half of the iterations are allocated to exploration ($|A| > 1$) and the other half of the iterations are allocated to exploitation ($|A| < 1$).
- ❆ a and B are two main vectors of GWO algorithm [2, 23].

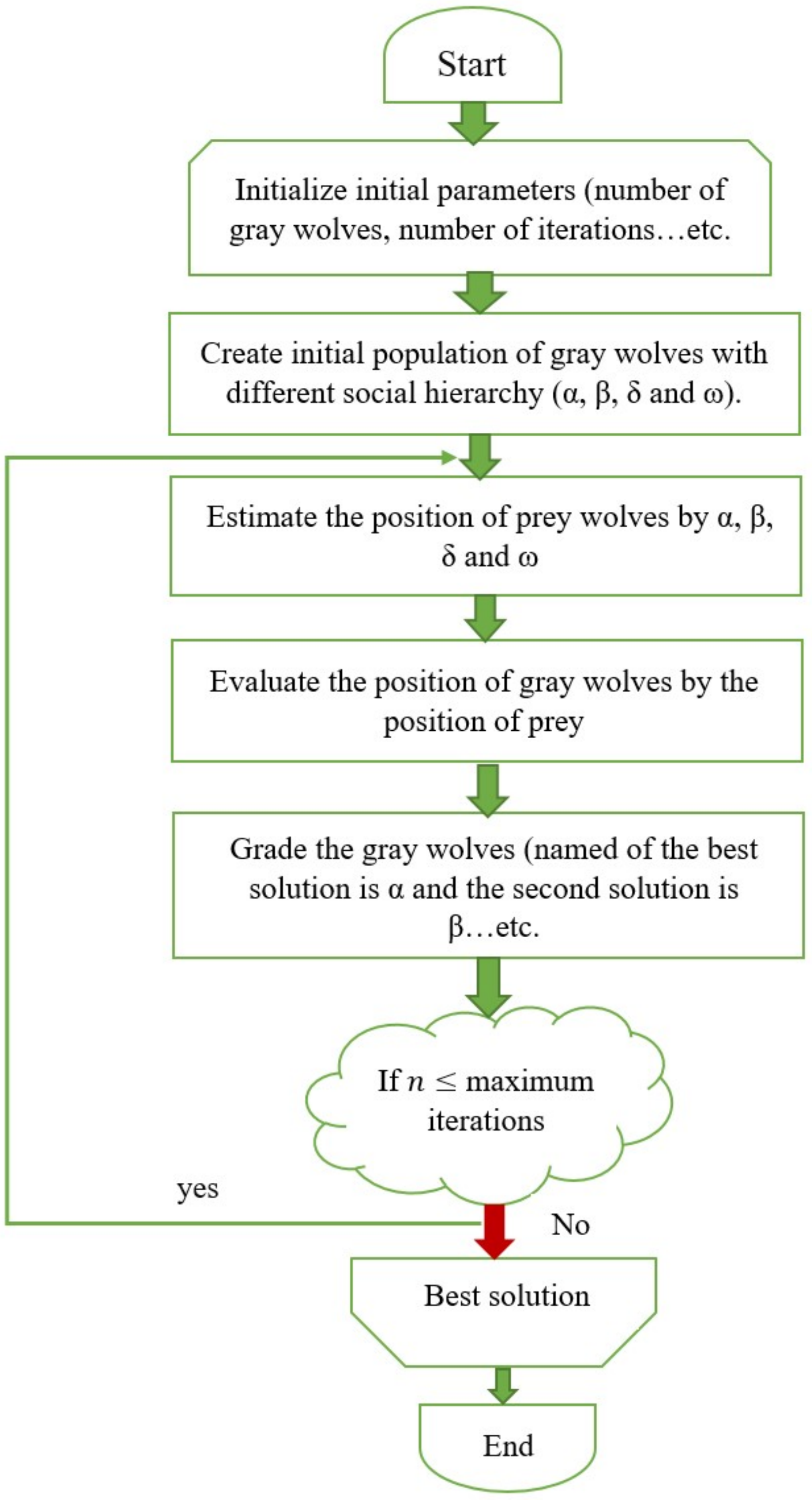

Start
Initialize initial parameters (number of gray wolves, number of iterations…etc.
Create initial population of gray wolves with different social hierarchy (α, β, δ and ω).
Estimate the position of prey wolves by α, β, δ and ω
Evaluate the position of gray wolves by the position of prey
Grade the gray wolves (named of the best solution is α and the second solution is β…etc.
If n ≤ maximum iterations
yes
No
Best solution
End

Figure 4.4: The Scheme of Action of The Gray Wolf Algorithm

CHAPTER 5

THE RESULTS OF DYNAMIC OPTIMIZATION AND OPTIMAL CONTROL

Introduction

In this chapter, will show the numerical solutions to models of mathematical questions related to the criterion of dynamic optimization, as well as the numerical solutions to these problems by using Python code for a special algorithms, which are the Interior Point Line Search Filter method and Grey Wolf Optimization (GWO) in solving dynamic optimization systems.

5.1 Applications for Dynamic Optimization Problems in Chemical Processes

Example 5.1.1. This is a basic chemical dynamic optimization problem with an analytical solution, which is a mathematical system that is not limited. Two state variables are involved in this example, which has an analytical solution and a global optimum. Numerous scholars used this example to demonstrate their methodologies [34, 39]. As an example, consider the following mathematical model:

$$\begin{cases} \min\limits_{u(t)} \quad \chi_2(t_f) \\ \text{subject to} \quad \dfrac{d\chi_1}{dt} = u \\ \qquad\qquad\quad \dfrac{d\chi_2}{dt} = \chi_1^2 + u^2 \\ \qquad\qquad\quad \chi(0) = [2 \quad 0]^T \\ \qquad\qquad\quad -1 \le u \le 0 \quad , \quad t_f = 1 \end{cases} \tag{5.1}$$

$$\begin{cases} \min\limits_{u(t)} \quad \chi_2(t_f) \\ \text{subject to} \quad \dfrac{d\chi_1}{dt} = u \\ \qquad\qquad\quad \dfrac{d\chi_2}{dt} = \chi_1^2 + u^2 \\ \qquad\qquad\quad \chi(0) = [2 \quad 0]^T \\ \qquad\qquad\quad \chi_1(t_f) = 1 \\ \qquad\qquad\quad -1 \leq u \leq 0 \quad , \quad t_f = 1 \end{cases} \tag{5.2}$$

Example 5.1.2. It's a typical chemical dynamic optimization issue with an analytical solution, a dynamic mathematical system. Including four state variables results in an analytical and global optimal solution for the state. For example, have a look at this mathematical formula:

$$\begin{cases} \min\limits_{u(t)} \quad \chi_4(t_f) \\ \text{subject to} \quad \dfrac{d\chi_1}{dt} = \chi_2 \\ \qquad\qquad\quad \dfrac{d\chi_2}{dt} = -\chi_3 u + 16t - 8 \\ \qquad\qquad\quad \dfrac{d\chi_3}{dt} = u \\ \qquad\qquad\quad \dfrac{d\chi_4}{dt} = \chi_1^2 + \chi_2^2 + 0.00516\left(\chi_2 + 16t - 8 - 0.1\chi_3 u^2\right)^2 \\ \qquad\qquad\quad \chi(0) = \begin{bmatrix}0 & -1 & -\sqrt{3} & 1\end{bmatrix}^T \\ \qquad\qquad\quad -3 \leq u \leq 11 \quad , \quad t_f = 1 \end{cases} \tag{5.3}$$

$\chi_1(t)$ to $\chi_4(t)$ are state vectors, while $u(t)$ is the control vector.

Example 5.1.3. **Parallel Reactions in Tubular Reactor:**

This instance depicts two concurrent chemical processes occurring in a tubular reactor $A \rightarrow B$ and $B \rightarrow C$. The ideal control variable's response trajectory must be designed in such a way that the concentration of target product B is maximized. The issue is as follows: χ_2 at the end, where reaction $A \rightarrow B \rightarrow C$ takes place. An example of how the math model can be explained is as follows:

$$\begin{cases} \max\limits_{u(t)} \quad J(t_f) = \chi_2(t_f) \\ \text{subject to} \quad \dfrac{d\chi_1}{dt} = -\left(u + 0.5u^2\right)\chi_1 \\ \qquad\qquad\quad \dfrac{d\chi_2}{dt} = u\chi_1 \\ \qquad\qquad\quad \chi(0) = [1 \quad 0]^T \\ \qquad\qquad\quad 0 \leq u \leq 5 \quad , \quad t_f = 1 \end{cases} \tag{5.4}$$

Example 5.1.4. **(Catalyst Mixing Problem)**

This challenge is solved using a plug flow reactor that is filled with two catalysts. The objective is to find the optimal catalyst concentration profile that optimizes the yield of the intermediate product C for a certain reactor length in which the reaction $(A \leftrightarrow B \rightarrow C)$ occurs. This issue's dynamic optimization problem may be characterized as follows:

$$\begin{cases} \max\limits_{u(t)} \quad J(t_f) = (1 - \chi1(t_f) - \chi_2(t_f)) \\ \text{subject to} \quad \dfrac{d\chi_1}{dt} = u\,(10\chi_2 - \chi_1) \\ \qquad\qquad\quad \dfrac{d\chi_2}{dt} = -u\,(10\chi_2 - \chi_1) - (1-u)\,\chi_2 \\ \qquad\qquad\quad \chi(0) = [1 \quad 0]^T \\ \qquad\qquad\quad 0 \leq u \leq 1 \quad , \quad t_f = 12 \end{cases} \tag{5.5}$$

5.2 Numerical Results of the Interior Point Line Search Filter method

1. Note example 5.1.1 the system in system 5.1 nonlinear, unconstrained, and with the goal of minimizing the final state. To solve this dynamic optimization system 5.1, will use **GEKKO** python code to find the optimal solution by minimizing the final state of the nonlinear system with an unconstraint dynamic and value of $u = -0.5$.

Code 3: Chemical Dynamic Optimization Problem, Nonlinear, Unconstrained

```
import numpy as np

import matplotlib.pyplot as plt

from gekko import GEKKO

m = GEKKO()

nt = 101

m.time = np.linspace(0,1,nt)

x1 = m.Var(value=2)

x2 = m.Var(value=0)

u = m.Var(value=-0.5)

p = np.zeros(nt)

p[-1] = 1.0

final = m.Param(value=p)

m.Equation(x1.dt()==u)

m.Equation(x2.dt()==x1**2 + u**2)

m.Obj(x2*final)

m.options.IMODE = 6

m.solve()

plt.figure(1)

plt.plot(m.time,x.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()
```

With a two-variable solution to the dynamic issue, will also minimize by altering the value of u, which will do by changing χ_1 and χ_2. χ_1 is equal to 2, and χ_2 is equal to 0; therefore, will going to compute the value until χ_1 is equal to 2, and which will do the same thing for χ_2 until it is equal to 0, as you can see from our input or variables used to manipulate the χ conditions, note Figure 5.1. Then getting:

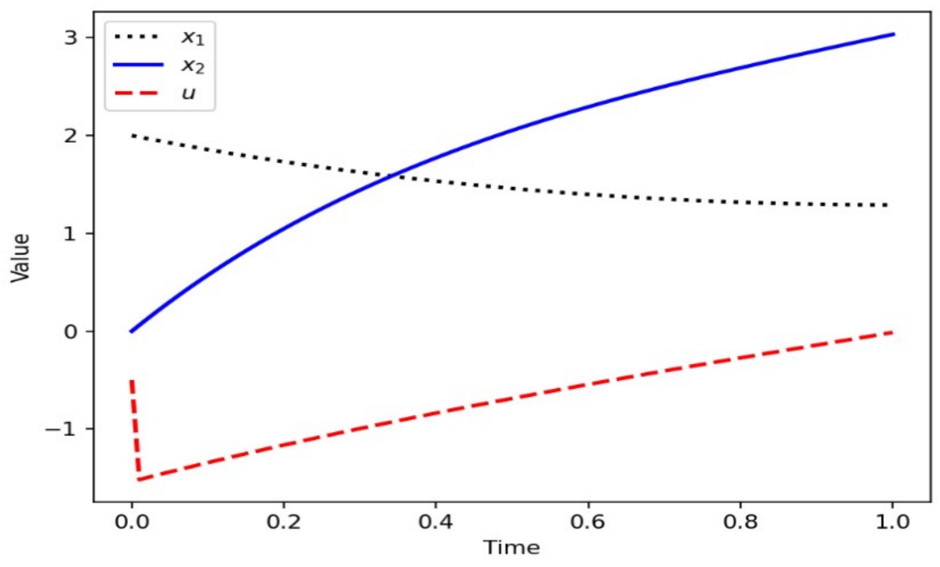
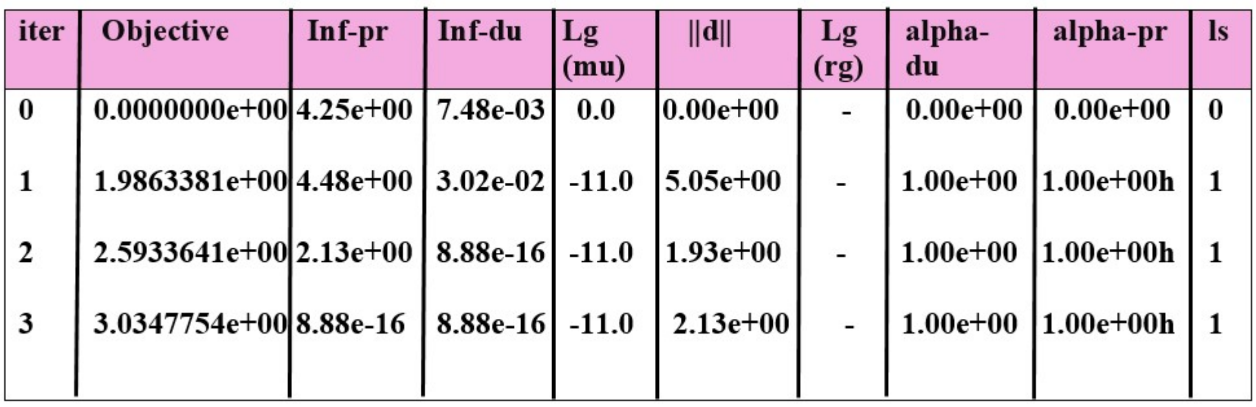


Figure 5.1: Dynamic optimization created the control and state profiles

Then the result of iterations shown in the table 5.1



iter	Objective	Inf-pr	Inf-du	Lg (mu)	||d||	Lg (rg)	alpha-du	alpha-pr	ls
0	0.0000000e+00	4.25e+00	7.48e-03	0.0	0.00e+00	-	0.00e+00	0.00e+00	0
1	1.9863381e+00	4.48e+00	3.02e-02	-11.0	5.05e+00	-	1.00e+00	1.00e+00h	1
2	2.5933641e+00	2.13e+00	8.88e-16	-11.0	1.93e+00	-	1.00e+00	1.00e+00h	1
3	3.0347754e+00	8.88e-16	8.88e-16	-11.0	2.13e+00	-	1.00e+00	1.00e+00h	1

Table 5.1: Numerical Results of Example 5.1.1

Number of Iterations: 3 ; Therefore, have the final result of

Solution time: $2.749999999650754E-002sec$

Objective: 3.03477538043263

Successful solution

2. Notice example 5.1.1. Final state minimization with terminal constraint is nonlinear and unconstrained for 5.2. To solve this dynamic optimization system, by use GEKKO python code to find the optimal solution by minimizing the final state of the nonlinear system with an unconstraint dynamic such that the value of $u = -0.5$.

Code 4: Chemical Dynamic Optimization Problem, Nonlinear and Constrained

```
import numpy as np
import matplotlib.pyplot as plt
from gekko import GEKKO
m = GEKKO()
nt = 101
m.time = np.linspace(0,1,nt)
x1 = m.Var(value=2)
x2 = m.Var(value=0)
u = m.Var(value=-0.5)
p = np.zeros(nt)
p[-1] = 1.0
final = m.Param(value=p)
m.Equation(\chi1.dt()==u)
m.Equation(x2.dt()==x1**2 + u**2)
m.Equation(final*(x1-1)==0)
m.Obj(x2*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()
```

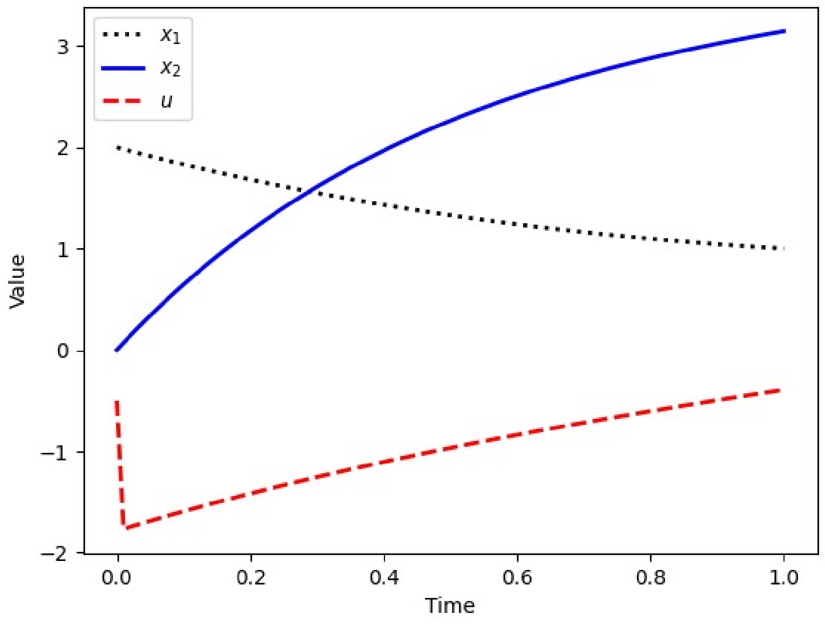
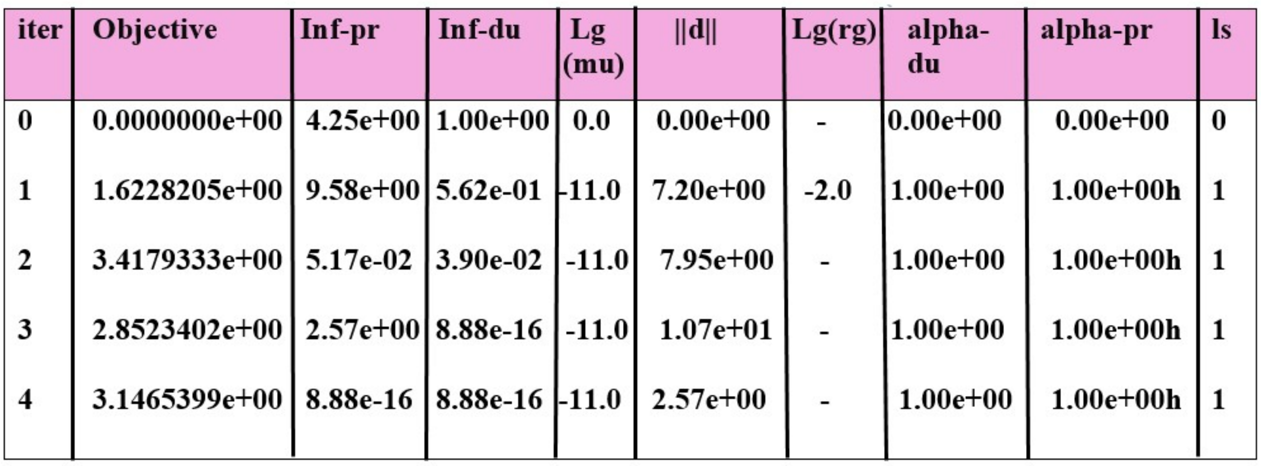


Figure 5.2: Dynamic Optimization Created the Control and State Profiles

With a two-variable solution to the dynamic problem, may also minimize by modifying the value of u, which can accomplish by modifying χ_1 and χ_2. Because χ_1 equals 2 and χ_2 equals 0 , we'll calculate the value till χ_1 equals 2 and χ_2 equals 0 , as indicated by our input or variables utilized to alter the χ conditions, see Figure 5.2, then getting:

Number of Iterations....: 4



iter	Objective	Inf-pr	Inf-du	Lg (mu)	||d||	Lg(rg)	alpha-du	alpha-pr	ls
0	0.0000000e+00	4.25e+00	1.00e+00	0.0	0.00e+00	-	0.00e+00	0.00e+00	0
1	1.6228205e+00	9.58e+00	5.62e-01	-11.0	7.20e+00	-2.0	1.00e+00	1.00e+00h	1
2	3.4179333e+00	5.17e-02	3.90e-02	-11.0	7.95e+00	-	1.00e+00	1.00e+00h	1
3	2.8523402e+00	2.57e+00	8.88e-16	-11.0	1.07e+01	-	1.00e+00	1.00e+00h	1
4	3.1465399e+00	8.88e-16	8.88e-16	-11.0	2.57e+00	-	1.00e+00	1.00e+00h	1

Table 5.2: Numerical Results of Example 5.1.1

Also getting: Solution time: 0.119099999996251 sec

Objective: 3.14653989628957

3. For solving nonlinear systems in example 5.1.2 for eq. 5.3 that have the constraint to minimize the final state, used code GEKKO of Python and get:

Code 5: Solving Nonlinear Systems That have Constraint

```
import numpy as np
import matplotlib.pyplot as plt
from gekko import GEKKO
m = GEKKO()
nt = 101
m.time = np.linspace(0,1,nt)
u = m.MV(value=9,lb=-3,ub=11)
u.STATUS = 1
u.DCOST = 0
t = m.Var(value=0)
```

```
x1 = m.Var(value=0)

x2 = m.Var(value=-1)

x3 = m.Var(value=-np.sqrt(3))

x4 = m.Var(value=1)

p = np.zeros(nt)

p[-1] = 1.0

final = m.Param(value=p)

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 \

+0.005*(x2+16*t-8-0.1*x3*(u**2))**2)

m.Obj(x4*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(tf): ' + str(x4[-1]))

plt.figure(2)

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()
```

With the four-variable solution to the dynamic issue, will also minimize by altering the value of u, which do it by changing χ_1, χ_2, χ_3 and $\chi_4.\chi_1$ is equal to $2, \chi_2 = -1$, $\chi_3 = \sqrt{3}$ and χ_4 is equal to 1 ; therefore, will going to compute the value of χ' s, as you can see from our input or variables being used to manipulate the X conditions, see Figure 5.3

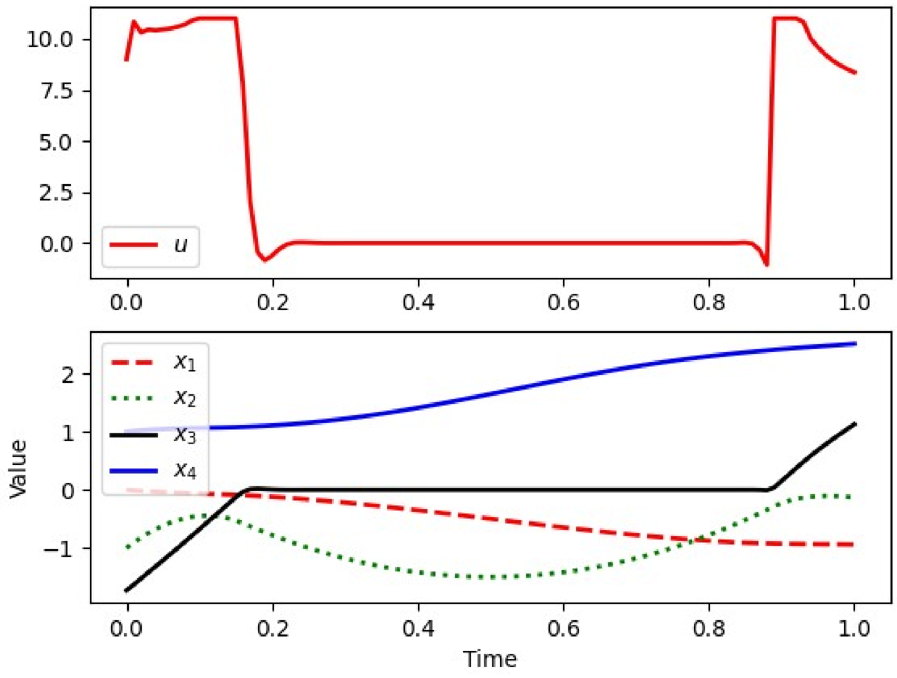


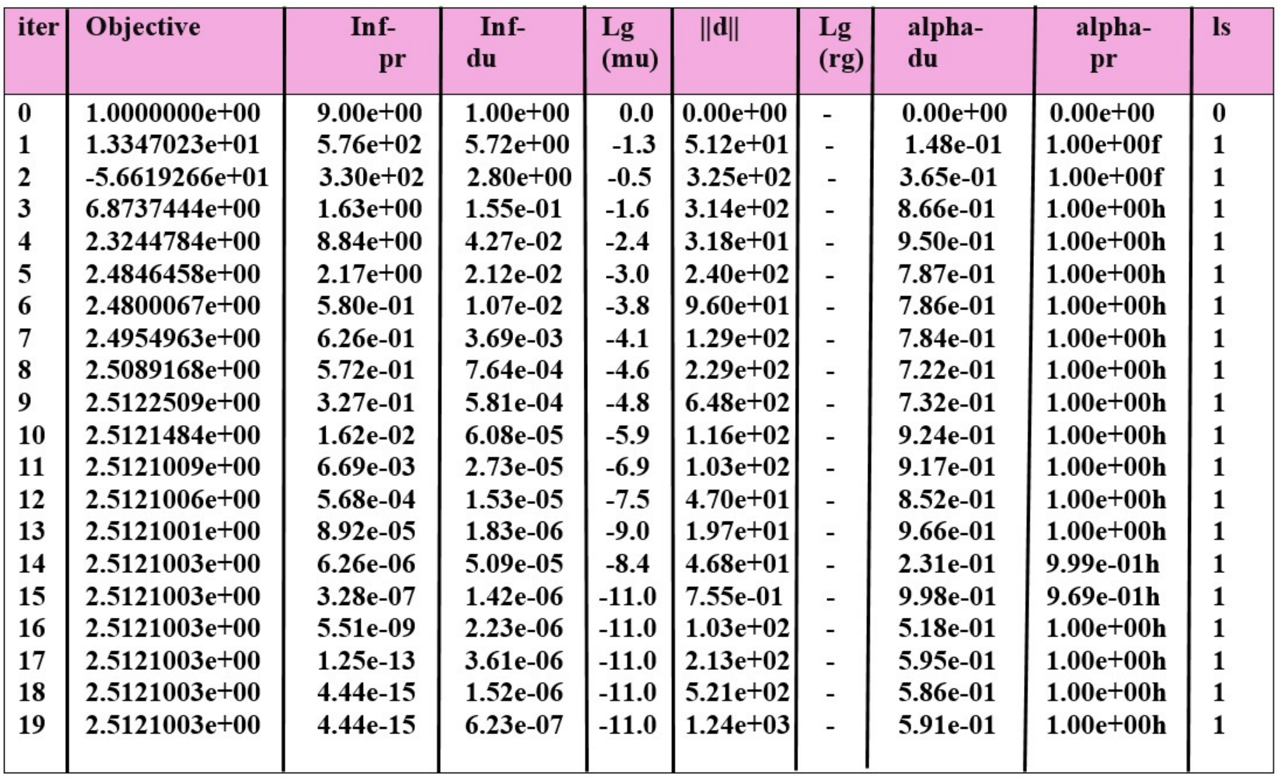
Figure 5.3: The Control and State Profiles were Developed Using Dynamic Optimization

Therefore have the following results:

Number of Iterations....: 19

Solution time: 0.756499999988591sec

Objective: 2.51210029228964



iter	Objective	Inf-pr	Inf-du	Lg (mu)	||d||	Lg (rg)	alpha-du	alpha-pr	ls
0	1.0000000e+00	9.00e+00	1.00e+00	0.0	0.00e+00	-	0.00e+00	0.00e+00	0
1	1.3347023e+01	5.76e+02	5.72e+00	-1.3	5.12e+01	-	1.48e-01	1.00e+00f	1
2	-5.6619266e+01	3.30e+02	2.80e+00	-0.5	3.25e+02	-	3.65e-01	1.00e+00f	1
3	6.8737444e+00	1.63e+00	1.55e-01	-1.6	3.14e+02	-	8.66e-01	1.00e+00h	1
4	2.3244784e+00	8.84e+00	4.27e-02	-2.4	3.18e+01	-	9.50e-01	1.00e+00h	1
5	2.4846458e+00	2.17e+00	2.12e-02	-3.0	2.40e+02	-	7.87e-01	1.00e+00h	1
6	2.4800067e+00	5.80e-01	1.07e-02	-3.8	9.60e+01	-	7.86e-01	1.00e+00h	1
7	2.4954963e+00	6.26e-01	3.69e-03	-4.1	1.29e+02	-	7.84e-01	1.00e+00h	1
8	2.5089168e+00	5.72e-01	7.64e-04	-4.6	2.29e+02	-	7.22e-01	1.00e+00h	1
9	2.5122509e+00	3.27e-01	5.81e-04	-4.8	6.48e+02	-	7.32e-01	1.00e+00h	1
10	2.5121484e+00	1.62e-02	6.08e-05	-5.9	1.16e+02	-	9.24e-01	1.00e+00h	1
11	2.5121009e+00	6.69e-03	2.73e-05	-6.9	1.03e+02	-	9.17e-01	1.00e+00h	1
12	2.5121006e+00	5.68e-04	1.53e-05	-7.5	4.70e+01	-	8.52e-01	1.00e+00h	1
13	2.5121001e+00	8.92e-05	1.83e-06	-9.0	1.97e+01	-	9.66e-01	1.00e+00h	1
14	2.5121003e+00	6.26e-06	5.09e-05	-8.4	4.68e+01	-	2.31e-01	9.99e-01h	1
15	2.5121003e+00	3.28e-07	1.42e-06	-11.0	7.55e-01	-	9.98e-01	9.69e-01h	1
16	2.5121003e+00	5.51e-09	2.23e-06	-11.0	1.03e+02	-	5.18e-01	1.00e+00h	1
17	2.5121003e+00	1.25e-13	3.61e-06	-11.0	2.13e+02	-	5.95e-01	1.00e+00h	1
18	2.5121003e+00	4.44e-15	1.52e-06	-11.0	5.21e+02	-	5.86e-01	1.00e+00h	1
19	2.5121003e+00	4.44e-15	6.23e-07	-11.0	1.24e+03	-	5.91e-01	1.00e+00h	1

Table 5.3: Numerical Results of Example 5.1.2

4. To solve this dynamic optimization system in example 5.1.3, used **GEKKO** python code to find the optimal solution by maximizing the final state of the nonlinear system for eq. 5.4:

Code 6: Parallel Reactions in Tubular Reactor

```
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=5,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)

m.Equation(x1.dt()==-(u+0.5*u**2)*x1)

m.Equation(x2.dt()==u*x1)

# Objective Function

m.Obj(-x2*final)

m.options.IMODE = 6

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')

plt.show()
```

The concentrations of A and B are represented by χ_1 and χ_2. The performance index J is a number that represents how well a person performs. The control variable $u(t)$ is used in this equation. Terminal time is denoted by the letters t_f.

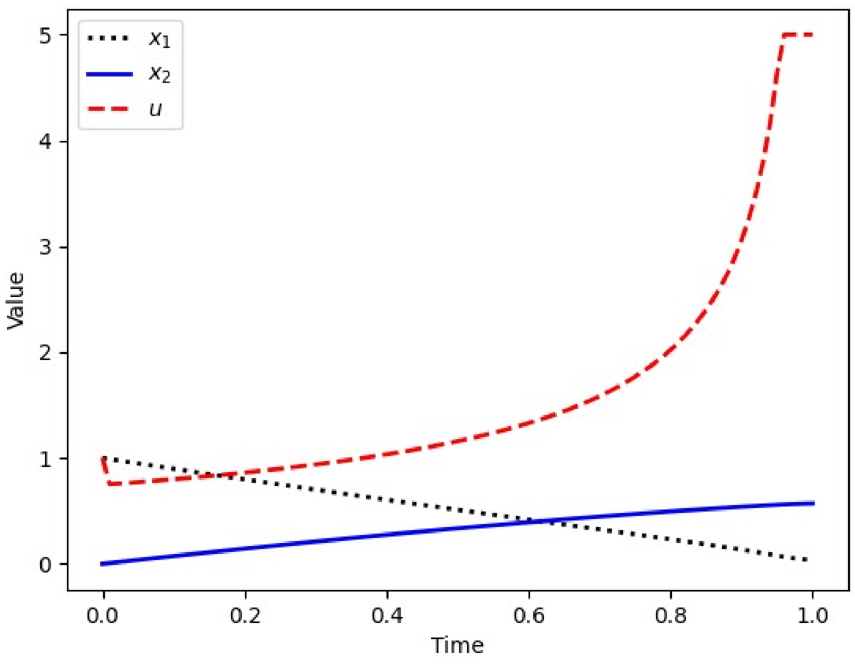
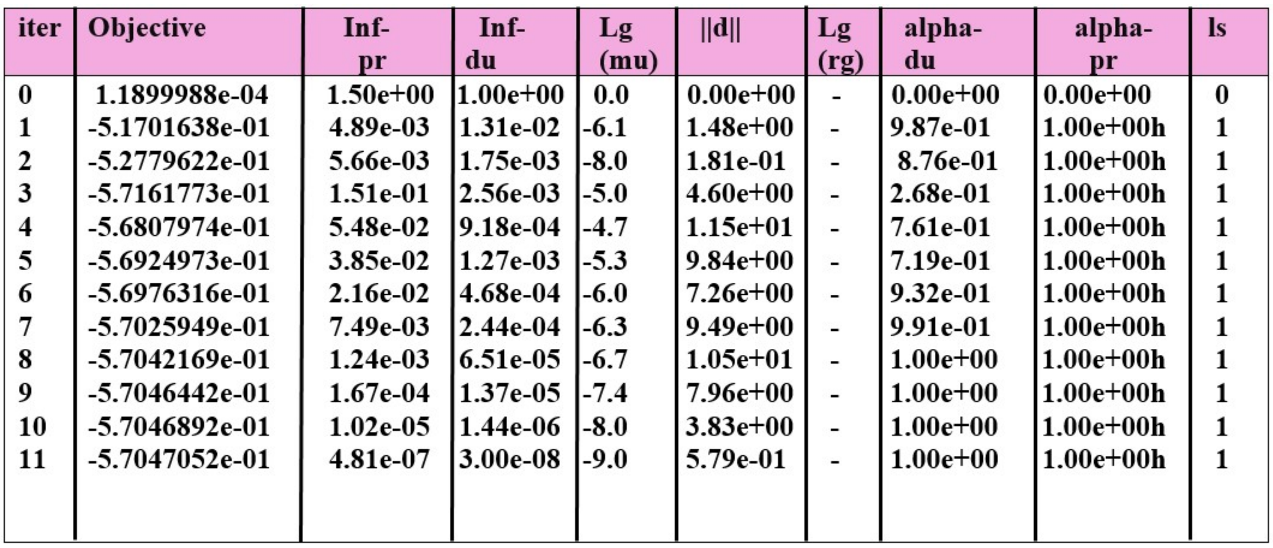


Figure 5.4: The Control and State Profiles were Developed Using Dynamic Optimization

The Table 5.4 illustrates the results of the concentration of the product following the implementation of a special Python code in the solution of such a situation, where the code indicates the maximum value of the objective function after iteration beginning with the initial values of the variables χ_1 and $\chi_2 = [1, 0]$ and the parameter u beginning with the value 1.

Number of Iterations....: 11



iter	Objective	Inf-pr	Inf-du	Lg (mu)	||d||	Lg (rg)	alpha-du	alpha-pr	ls
0	1.1899988e-04	1.50e+00	1.00e+00	0.0	0.00e+00	-	0.00e+00	0.00e+00	0
1	-5.1701638e-01	4.89e-03	1.31e-02	-6.1	1.48e+00	-	9.87e-01	1.00e+00h	1
2	-5.2779622e-01	5.66e-03	1.75e-03	-8.0	1.81e-01	-	8.76e-01	1.00e+00h	1
3	-5.7161773e-01	1.51e-01	2.56e-03	-5.0	4.60e+00	-	2.68e-01	1.00e+00h	1
4	-5.6807974e-01	5.48e-02	9.18e-04	-4.7	1.15e+01	-	7.61e-01	1.00e+00h	1
5	-5.6924973e-01	3.85e-02	1.27e-03	-5.3	9.84e+00	-	7.19e-01	1.00e+00h	1
6	-5.6976316e-01	2.16e-02	4.68e-04	-6.0	7.26e+00	-	9.32e-01	1.00e+00h	1
7	-5.7025949e-01	7.49e-03	2.44e-04	-6.3	9.49e+00	-	9.91e-01	1.00e+00h	1
8	-5.7042169e-01	1.24e-03	6.51e-05	-6.7	1.05e+01	-	1.00e+00	1.00e+00h	1
9	-5.7046442e-01	1.67e-04	1.37e-05	-7.4	7.96e+00	-	1.00e+00	1.00e+00h	1
10	-5.7046892e-01	1.02e-05	1.44e-06	-8.0	3.83e+00	-	1.00e+00	1.00e+00h	1
11	-5.7047052e-01	4.81e-07	3.00e-08	-9.0	5.79e-01	-	1.00e+00	1.00e+00h	1

Table 5.4: Numerical Results of Example 5.1.3

Therefore; the final results

Solution time: 9.8799999999247957E-002 sec

Objective: -0.570470520143042

Successful solution

Objective: 0.57085514067

5. To solve this dynamic optimization system in example 5.1.4, used **GEKKO** python code to find the optimal solution by maximizing the final state of the nonlinear system for eq. 5.5:

Code 7: Catalyst Mixing Problem

```
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
x1 = m.Var(value=1)
x2 = m.Var(value=0)
p = np.zeros(nt)
p[-1] = 1.0
final = m.Param(value=p)
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')
plt.show()
```

χ_1, χ_2, and J are the mole fractions of substances A, B, and C, respectively. The length of the tubular reactor's pipe is denoted by t_f . $u(t_f)$ is the catalyst A mixing fraction.

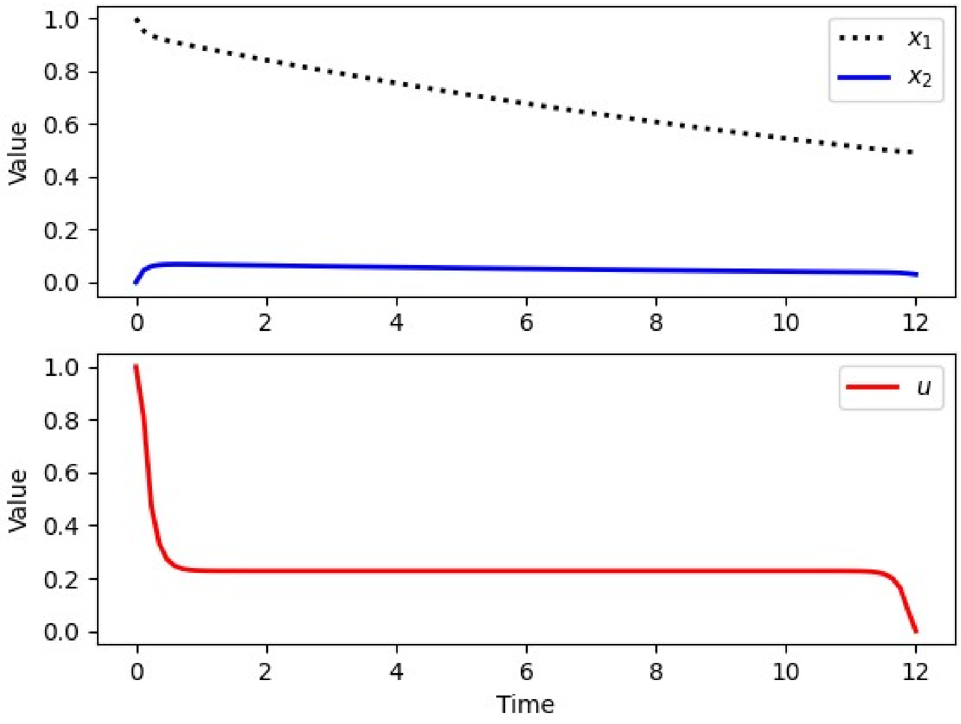
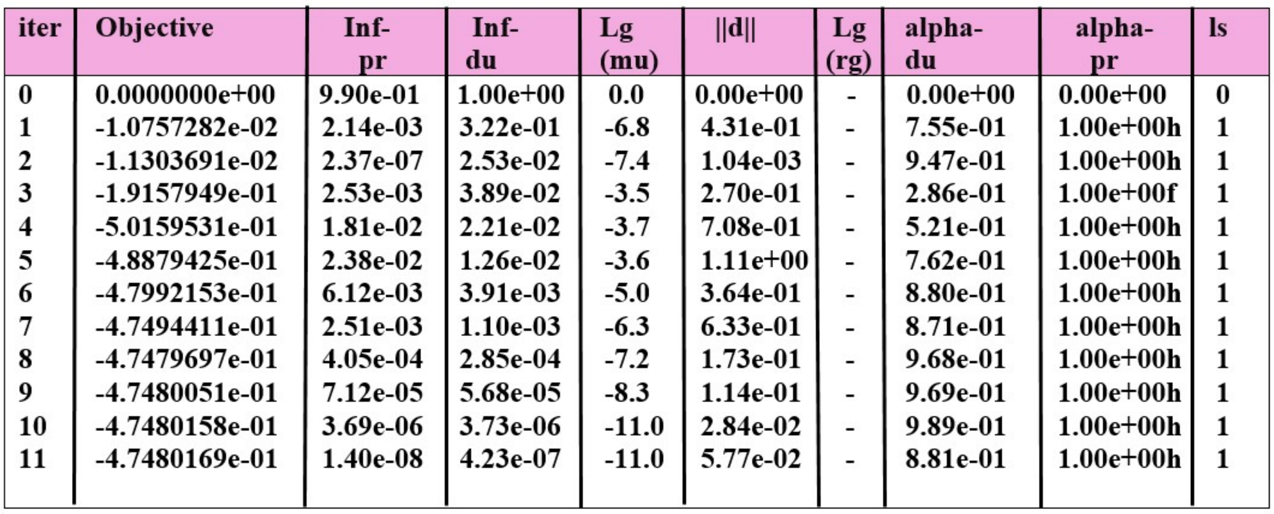


Figure 5.5: The Control and State Profiles were Developed Using Dynamic Optimization

The Table 5.5 summarizes the ideal results produced by using Python code to find the optimal solution for the objective function after several iterations using the initial values of the variables χ_1 and χ_2 and the parameter u.

Number of Iterations....: 11



iter	Objective	Inf-pr	Inf-du	Lg (mu)	||d||	Lg (rg)	alpha-du	alpha-pr	ls
0	0.0000000e+00	9.90e-01	1.00e+00	0.0	0.00e+00	-	0.00e+00	0.00e+00	0
1	-1.0757282e-02	2.14e-03	3.22e-01	-6.8	4.31e-01	-	7.55e-01	1.00e+00h	1
2	-1.1303691e-02	2.37e-07	2.53e-02	-7.4	1.04e-03	-	9.47e-01	1.00e+00h	1
3	-1.9157949e-01	2.53e-03	3.89e-02	-3.5	2.70e-01	-	2.86e-01	1.00e+00f	1
4	-5.0159531e-01	1.81e-02	2.21e-02	-3.7	7.08e-01	-	5.21e-01	1.00e+00h	1
5	-4.8879425e-01	2.38e-02	1.26e-02	-3.6	1.11e+00	-	7.62e-01	1.00e+00h	1
6	-4.7992153e-01	6.12e-03	3.91e-03	-5.0	3.64e-01	-	8.80e-01	1.00e+00h	1
7	-4.7494411e-01	2.51e-03	1.10e-03	-6.3	6.33e-01	-	8.71e-01	1.00e+00h	1
8	-4.7479697e-01	4.05e-04	2.85e-04	-7.2	1.73e-01	-	9.68e-01	1.00e+00h	1
9	-4.7480051e-01	7.12e-05	5.68e-05	-8.3	1.14e-01	-	9.69e-01	1.00e+00h	1
10	-4.7480158e-01	3.69e-06	3.73e-06	-11.0	2.84e-02	-	9.89e-01	1.00e+00h	1
11	-4.7480169e-01	1.40e-08	4.23e-07	-11.0	5.77e-02	-	8.81e-01	1.00e+00h	1

Table 5.5: Numerical Results of Example 5.1.4

Then the final results is

Solution time: 0.103399999992689 sec

Objective: -0.474801689458803

Successful solution

Objective: 0.47480168945600001

5.3 Numerical Results of Gray Wolf Optimization Algorithm

In this section, the researcher will explain the solution of several systems using the gray wolf algorithm described in the section 4.4. The results were obtained after designing the Python code using the data described in the subsection4.6.3, as well as information about the systems described in the section 5.1.

To show the final results of the target function after the implementation of the gray wolf algorithm code designed in Python, the following table can be observed :

Objective func.	Name of Sol.	The value
$\min_{u(t)} \quad \chi_2(t_f)$	First	2.58765435696733
	Second	2.5876560817295533
	Third	2.5876676890101793
$\min_{u(t)} \quad \chi_2(t_f)$	First	2.5876560817295533
	Second	2.5876676890101793
	Third	2.5876702779324914
$\min_{u(t)} \quad \chi_4(t_f)$	First	2.592242295235336
	Second	2.5926471955786026
	Third	2.592696780150721
$\max_{u(t)} \quad J(t_f) = \chi_2(t_f)$	First	2.5876772021343095
	Second	2.587675059027466
	Third	2.587666566836703
$\max_{u(t)} \quad J(t_f) = \chi_2(t_f)$	First	0.04944511236197066
	Second	0.0408791758432258
	Third	0.038651958957129584

Table 5.6: Test Result Benchmark Functions

Notice the Table above 5.6 shows the values of the objective functions, where according to the gray wolf algorithm, the objective function takes three values divided by alpha, beta and delta, where alpha is the first best value, beta is the second best value and delta is the third best value for each target function:

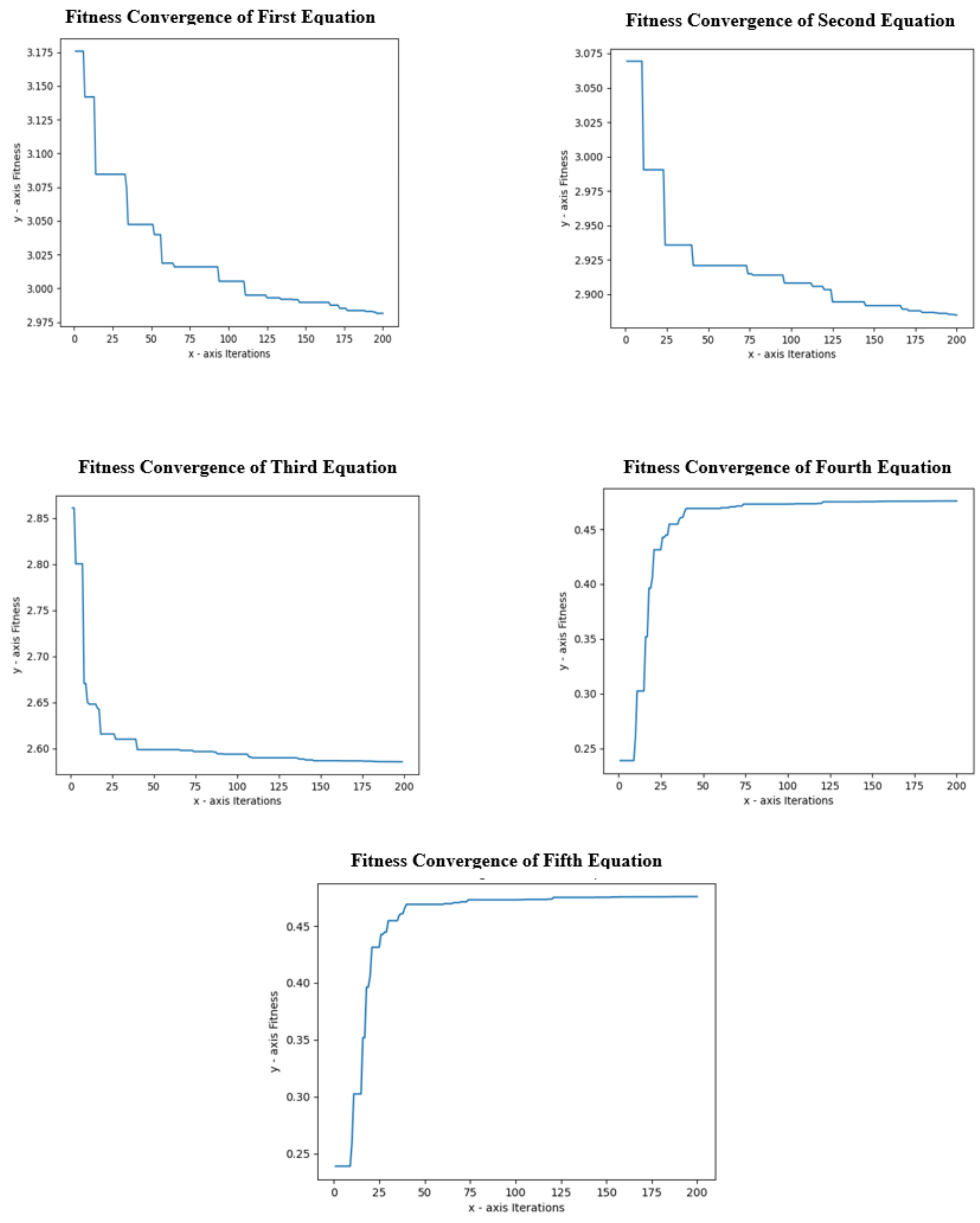


Figure 5.6: The Best Solution Through 200 Iterations Eq.(5.1) , (5.2) ,(5.3) , (5.4) , (5.5)

The Figure (5.6) that show the final solution of the indicated dynamic optimization systems respectively ((5.1) , (5.2) , (5.3) , (5.4) , (5.5)) after using the gray wolf algorithm to obtain the best solutions, where the optimal solution for each system of equations can be found from its own drawing , and the number of repetitions was chosen 200 repetitions and also with the use of the mentioned conditions for each system, the Table 5.6 explain the final solution according to GWO, also the code of solving (5.3) can shown down:

Code8: GWO Code for Solving The System 5.3 Example 5.1.2

```
import numpy as np
import matplotlib.pyplot as plt
from gekko import GEKKO
def print_hi(name):
# Use a breakpoint in the code line below to debug your script.
print(f'Hi, {name}')  # Press Ctrl+F8 to toggle the breakpoint.
# Press the green button in the gutter to run the script.
if __name__ == '__main__':
print_hi('PyCharm')
m=GEKKO()
nt=11
m.time=np.linspace(0,1,nt)
#Time Discret
T=np.linspace(0,1,nt)
dt=0.1
# Input Control
U=np.full(nt,0.5)
#State Variables
X1=[]
```

```
X2=[]

X3=[]

X4=[]

# Initialization of State Variables

X1.append(0)

X2.append(-1)

X3.append(np.sqrt(3))

X4.append(1)

#Fill Control Array U

#for i in range(nt):

#   U.append(random.random()*pow(-1,i))

print('Time Vector T:')

print(T)

print('Time Vector T Length:')

print(len(T))

print('Initial Control Vector U:')

print(U)

print('Initial Control Vector U Length:')

print(len(U))

# Calculation of X3

for i in range(1,nt):

X3.append(U[i]*dt+X3[i-1])

print('Initial State Vector X3:')

print(X3)

# Calculation of X2

for i in range(1,nt):

a=(-X3[i]*U[i]+16*T[i]-8)*dt +X2[i-1]

X2.append(a)
```

```
print('Initial State Vector X2:')

print(X2)

# Calculation of X1

for i in range(1,nt):

a=X2[i]*dt +X1[i-1]

X1.append(a)

print('Initial State Vector X1:')

print(X1)

# Calculation of X4

for i in range(1,nt):

a=(X1[i]**2 +X2[i]**2 +0.00516 *(X2[i]+16*T[i]-8 -0.1*X3[i]*U[i]**2)**2)*dt
 +X4[i-1]X4.append(a)

print('Initial State Vector X4:')

print(X4)

# Grey Wolf Algorithm

X_Alpha = []

X_Beta = []

X_Delta = []

def Gery_Wolf_Fit_Fun (U):

#print('Initial State Vector U:')

#print(U)

X01=np.zeros(nt)

X02=np.zeros(nt)

X03=np.zeros(nt)

X04=np.zeros(nt)

#Initializaing Variables

X01[0]=0

X02[0]=-1
```

```
X03[0]=np.sqrt(3)

X04[0]=1

# Calculation of X3

for i in range(1, nt):

X03[i]= U[i] * dt + X03[i - 1]

#print('Initial State Vector X3:')

#print(X03)

# Calculation of X2

for i in range(1, nt):

X02[i] = (-X03[i] * U[i] + 16 * T[i] - 8) * dt + X02[i - 1]

#print('Initial State Vector X2:')

#print(X02)

# Calculation of X1

for i in range(1, nt):

X01[i] = X02[i] * dt + X01[i - 1]

#print('Initial State Vector X1:')

#print(X01)

# Calculation of X4

for i in range(1, nt):

X04[i] = (X01[i] ** 2 + X02[i] ** 2 + 0.00516 * (X02[i] +

16 * T[i] - 8 - 0.1 * X03[i] * U[i] ** 2) ** 2) * dt + X04[i- 1]

#print('Initial State Vector X4:')

#print(X04)

return X04[T_Final]

print('Fitness Value for U:')

print(Gery_Wolf_Fit_Fun(U))

#Initializing of populations array

for i in range(pop_no):
```

```
rand_arr=np.random.random_sample(nt)
Diff_arr=U_Max-U_Min
xi=U_Min+np.multiply(rand_arr,Diff_arr)
Pop_Arr.append(xi)
print('Initialized Pop_arr:')
for i in range(pop_no):
print(Pop_Arr[i])
print()
'''a=0.01
X=Pop_Arr[0]
A1=2*a*rand_array-a
C1=2*rand_array
print('A1:')
print(A1)
print('C1:')
print(C1)
Dalpha=np.multiply(C1,X)
print('C1*X:')
print(Dalpha)'''
Fitness_Array=[]
for i in range(pop_no):
X = Pop_Arr[i]
FX = Gery_Wolf_Fit_Fun(X)
Fitness_Array.append(FX)
print('Fitness Array:',Fitness_Array)
# Calculation of X_alpha , X_Beta , X_delta
temp_Fit_Arr = np.sort(Fitness_Array)
temp_Fit_Index_Arr=np.argsort(Fitness_Array)
```

```
print('Sorted Fitness Array:',temp_Fit_Arr)

print('Sorted temp_Fit_Index_Arr:',temp_Fit_Index_Arr)

alpha = temp_Fit_Arr[0]

alpha_Index = temp_Fit_Index_Arr[0]

X_Alpha = Pop_Arr[alpha_Index]

print('X_Alpha:',X_Alpha)

print('X_Alpha_Index:',alpha_Index)

beta = temp_Fit_Arr[1]

beta_Index = temp_Fit_Index_Arr[1]

X_Beta = Pop_Arr[beta_Index]

delta = temp_Fit_Arr[2]

delta_Index = temp_Fit_Index_Arr[2]

X_Delta = Pop_Arr[delta_Index]

print('Xalpha:',X_Alpha)

print('Xbeta:',X_Beta)

print('X_Delta',X_Delta)

for k in range(1,Max_Iterator):

a=2*(1-k/Max_Iterator)

for i in range(pop_no):

X=Pop_Arr[i]

# First best

A1=2*a*np.random.random_sample(nt)-a

C1=2*np.random.random_sample(nt)

D_alpha=abs(np.multiply(C1,X_Alpha) -X)

X1=X_Alpha-np.multiply(A1,D_alpha)

# Second best

A2= 2 * a * np.random.random_sample(nt) - a

C2 = 2 * np.random.random_sample(nt)
```

```
D_beta = abs(np.multiply(C2,X_Beta) - X)

X2 = X_Beta - np.multiply(A2,D_beta)

# Third best

A3 = 2 * a * np.random.random_sample(nt) - a

C3 = 2 * np.random.random_sample(nt)

D_delta = abs(np.multiply(C3,X_Delta) - X)

X3 = X_Delta - np.multiply(A3,D_delta)

# Calculate X_new

X_new=1/3*(X1+X2+X3)

# Perform Greedy Selection

FX_new=Gery_Wolf_Fit_Fun(X_new)

FX_old=Gery_Wolf_Fit_Fun(X)

if FX_new<FX_old:

Is_Legal=In_Constraint_Range(X_new)

if Is_Legal==True:

Pop_Arr[i]=X_new

# Calculate Fitness Function Array

Fitness_Array=[]

for i in range(pop_no):

X = Pop_Arr[i]

FX = Gery_Wolf_Fit_Fun(X)

Fitness_Array.append(FX)

print('Fitness_Array:',Fitness_Array)

# Calculation of X_alpha , X_Beta , X_delta

temp_Fit_Arr = np.sort(Fitness_Array)

temp_Fit_Index_Arr = np.argsort(Fitness_Array)

# X_Alpha

alpha = temp_Fit_Arr[0]
```

```
alpha_Index = temp_Fit_Index_Arr[0]
X_Alpha = Pop_Arr[alpha_Index]
beta = temp_Fit_Arr[1]
beta_Index = temp_Fit_Index_Arr[1]
X_Beta = Pop_Arr[beta_Index]
delta = temp_Fit_Arr[2]
delta_Index = temp_Fit_Index_Arr[2]
X_Delta = Pop_Arr[delta_Index]
GWA_Best_Index_Arr.append(alpha_Index)
GWA_Best_Fit_Arr.append(alpha)
print('Final U_Array:')
for i in range(pop_no):
print(Pop_Arr[i])
print()
Best_Sol_Index=GWA_Best_Index_Arr[len(GWA_Best_Index_Arr)-1]
Best_Sol=Pop_Arr[Best_Sol_Index]
print ('Best Solution Vector U:',Best_Sol)
plot_x=np.arange(1, Max_Iterator, 1).tolist()
plot_y=GWA_Best_Fit_Arr
# plotting the points
plt.plot(plot_x, plot_y)
# naming the x axis
plt.xlabel('x - axis Iterations')
# naming the y axis
plt.ylabel('y - axis Fitness')
# giving a title to my graph
plt.title('Fitness Convergence!')
plt.show()
```

5.4 Analysis and Discussion

1. The following Tables [4.3.1 , 5.1 , 5.2 , 5.3 , 5.4 , 5.5] are the final results after the execution of the custom codes for each of the dynamic optimization systems where (ipopt solver) was used, and therefore we note that each table is divided into 10 columns, and the columns of output are defined as follows:

   - ❄ iter: The current iteration count.
   - ❄ objective: The unscaled objective value at the current point.
   - ❄ inf-pr: The unscaled max constraint violation at the current point.
   - ❄ inf-du: The max scaled dual infeasibility at the current point.
   - ❄ lg(mu): log 10 of the value of the barrier parameter μ
   - ❄ ||d||: The infinity norm (max) of the primal step
   - ❄ lg(rg): log 10 of the value of the regularization term for the Hessian of the Lagrangian. Dash (-) indicates that no regularization was done.
   - ❄ alpha-du: The stepsize for the dual variables
   - ❄ alpha-pr: The stepsize for the primal variables
   - ❄ ls: The number of backtracking line search steps

2. Analysis of Figures

   - ❄ From Figure (5.1) note there is our optimal value of (u) and there is χ_1 start from 2 and then decreases and the objective to maximize the final value of χ_2.
   - ❄ From Figure(5.2) see there is our optimal value of (u) and there is χ_1 start at 2 goes down, there is χ_2 so we are trying to maximize to the value of χ_2 and the value of u changed a little bit over horizontal.
   - ❄ The values of (u) can see it's as a certain point about 0.2 that transition from an upper value of ten down to a value of zero and then goes back up to a value

of ten and can see the arc in the Figure 5.3. The goal was to minimize the value of χ_4 at final time , so that the value of χ came up and then was minimized in the way a little bit but this is the optimum optimal results there .

❊ Figure (5.4) express the solution of values of (u) , χ_1 and χ_2 then notice the value of χ_1 decreases to zero at the final time equal 1 also the value of χ_2 will increasing at the final time, so the goal is to maximize the value of χ_2 at t_f.

❊ The Figure 5.5 shows that the value of the variable u gradually begins to decrease, and then at the value of 0.2 it begins to stabilize in the same way during the expired time until it starts to decrease also at the value of the expired time $u = 11.8$ until it reaches the zero value of the values of the variables, as for the variable χ_1 at the initial point equal to 1, then it descends until it settles at the expired time equal to 12 for the value of 0.5 approximately, We note for the variable χ_2, where it starts from zero as the initial value at time $t_f = 0$ and then escalates at time $t = 0.5$ and over time settles on the same pattern and takes the value 0.1 even when $t_f = 12$

Conclusions

1. In this thesis, the researcher developed a model consisting of ordinary and nonlinear differential equations, which is an improvement model where the equations of this system are objective functions and the restrictions are imposed in Iraq, especially Karbala governorate, where real data were taken from the Karbala health department on the numbers of people infected and recovering from COVID-19 for 2021, where these data were considered the starting point in the development of the SIR model and the study of the survival period of the epidemic within 365 days and after numerical analysis through the use of Python codes to show the results and good results and high accuracy .

2. New approaches were also used in solving selected systems of dynamic optimization systems for chemical processes, where two algorithms were employed to find solutions to these models of systems, and those two algorithms are both the algorithm for finding the inner point and the gray wolves algorithm, and the results were found using Python codes to find numerical results that give the optimal solution.

3. The reason for choosing the Python programming language to find solutions is because it is a language that is easy to use and deal with in entering data or target functions and restrictions as a result of the availability of its libraries and online libraries, which is one of the languages that takes up little space in computer memory also have highest rank.

4. In the future, the thesis work will be developed through the expansion of the use of new systems of dynamic optimization and optimal control systems, as well as the use of other algorithms in showing the results, whether in the systems used in this study or systems from different fields from the field of chemical processes, and a comparison can also be made with the results.

REFERENCES

[1] Hassan Mohamed Abdelalim Abdalla and Daniele Casagrande. Direct transcription approach to dynamic optimization problems in engineering. *Journal of Applied and Computational Mechanics*, 8(2):605–616, 2022.

[2] Mohamed Abdel-Basset, Doaa El-Shahat, Ibrahim El-henawy, Victor Hugo C de Albuquerque, and Seyedali Mirjalili. A new fusion of grey wolf optimizer algorithm with a two-phase mutation for feature selection. *Expert Systems with Applications*, 139:112824, 2020.

[3] Ban O Abdulsattar, Ian M Jones, et al. The pandemic of covid-19: Current scheme of iraq (24 february-8 august 2020). *Rafidain Journal of Science*, 30(1):11–17, 2021.

[4] A Aboulenein and R Levinson. The medical crisis that's aggravating iraq's unrest. a reuters special report. march 2, 2020.

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

[6] Raghda Alsayed, Ali Abd Ali, Raghda Makia, Mohammed Kadhom, Rasha Raheem, Omar Al-Obaidi, Angham Hadi, Dana Khdr Sabir, and Emad Yousif. Challenges

facing iraq to tackle the spread of covid-19: An overview. *Journal of university of Anbar for Pure science*, 14(2):22–27, 2022.

[7] Muhammad H Alsuwaiyel. *Algorithms: design techniques and analysis*, volume 15. World Scientific, 2021.

[8] Niclas Andréasson, Anton Evgrafov, and Michael Patriksson. An introduction to optimization: Foundations and fundamental algorithms. *Chalmers University of Technology Press: Gothenburg, Sweden*, 1:1–205, 2005.

[9] S Andriani, H Suyitno, I Junaidi, et al. The application of differential equation of verhulst population model on estimation of bandar lampung population. In *Journal of Physics: Conference Series*, volume 1155, page 012017. IOP Publishing, 2019.

[10] Rajesh Kumar Arora. *Optimization: algorithms and applications.* CRC Press, 2015.

[11] A Arpornwichanop, P Kittisupakorn, and IM Mujtaba. On-line dynamic optimization and control strategy for improving the performance of batch reactors. *Chemical Engineering and Processing: Process Intensification*, 44(1):101–114, 2005.

[12] Adil Bagirov, Napsu Karmitsa, and Marko M Mäkelä. *Introduction to Nonsmooth Optimization: theory, practice and software*, volume 12. Springer, 2014.

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

[14] Soufiane Bentout, Abdennasser Chekroun, and Toshikazu Kuniya. Parameter estimation and prediction for coronavirus disease outbreak 2019 (covid-19) in algeria. *AIMS Public Health*, 7(2):306, 2020.

[15] Dimitri Bertsekas. *Convex optimization algorithms.* Athena Scientific, 2015.

[16] David Betounes. *Differential Equations: theory and applications.* Springer, 2010.

[17] Titus A Beu. *Introduction to numerical programming: a practical guide for scientists and engineers using Python and C/C++*. CRC Press, 2014.

[18] Lorenz T Biegler. Optimization of differential-algebraic equation systems. *Chemical Engineering Department Carnegie Mellon University Pittsburgh, http://dynopt. cheme. cmu. edu*, 2000.

[19] Lorenz T Biegler. An overview of simultaneous strategies for dynamic optimization. *Chemical Engineering and Processing: Process Intensification*, 46(11):1043–1053, 2007.

[20] Nicolas Bourbaki. *Topological vector spaces: Chapters 1–5*. Springer Science & Business Media, 2013.

[21] William E Boyce, Richard C DiPrima, and Douglas B Meade. *Elementary differential equations and boundary value problems*. John Wiley & Sons, 2021.

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

[23] Omid Bozorg-Haddad. *Advanced optimization by nature-inspired algorithms*, volume 720. Springer, 2018.

[24] Charles L Byrne. *A first course in optimization*. CRC Press, 2014.

[25] Arturo Cervantes and Lorenz T Biegler. Optimization strategies for dynamic systems. *Encyclopedia of optimization*, 4:216–227, 2009.

[26] Xiao Dong Chen. Cooking potatoes: experimentation and mathematical modeling. *Chemical Engineering Education*, 36(1):26–31, 2002.

[27] Edwin KP Chong and Stanislaw H Zak. *An introduction to optimization*. John Wiley & Sons, 2004.

[28] Thomas H Cormen, Charles E Leiserson, Ronald L Rivest, and Clifford Stein. *Introduction to algorithms.* MIT press, 2022.

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

[30] Paul Dawkins. Systems of differential equations. 2007.

[31] Etienne De Klerk. *Aspects of semidefinite programming: interior point algorithms and selected applications*, volume 65. Springer Science & Business Media, 2006.

[32] Moritz Diehl, KU ESAT-SCD, Jan Bouckaert, David Ariens, Carlo Savorgnan, and Laurent Sorber. B-kul-h03e3a. 2009.

[33] Urmila M Diwekar. *Introduction to applied optimization*, volume 22. Springer Nature, 2020.

[34] Thomas F Edgar, David Mautner Himmelblau, Leon S Lasdon, et al. *Optimization of chemical processes*, volume 2. McGraw-Hill New York, 2001.

[35] Ahmed AM El-Gaafary, Yahia S Mohamed, Ashraf Mohamed Hemeida, and Al-Attar A Mohamed. Grey wolf optimization for multi input multi output system. *Universal Journal of Communications and Network*, 3(1):1–6, 2015.

[36] Jacob Engwerda. *LQ dynamic optimization and differential games.* John Wiley & Sons, 2005.

[37] Mohammad Fathi and Hassan Bevrani. *Optimization in electrical engineering.* Springer, 2019.

[38] Zhong-kai Feng, Wen-jing Niu, and Shuai Liu. Cooperation search algorithm: A novel metaheuristic evolutionary intelligence algorithm for numerical optimization and engineering optimization problems. *Applied Soft Computing*, 98:106734, 2021.

[39] Grégory Francois and Dominique Bonvin. Measurement-based real-time optimization of chemical processes. In *Advances in Chemical Engineering*, volume 43, pages 1–50. Elsevier, 2013.

[40] Franco Giannessi, Antonino Maugeri, and Panos M Pardalos. *Equilibrium problems: nonsmooth optimization and variational inequality models*, volume 58. Springer Science & Business Media, 2006.

[41] Henryk Górecki. Optimization and control of dynamic systems. *AGH University of Science and Technology Press, Krakow*, 2006.

[42] NIM Gould, D Orban, and PL Toint. Numerical analysis and optimization: Nao-iii, muscat, oman, january 2014, chap. *An interior-point l1-penalty method for nonlinear optimization. Springer, Cham*, pages 117–150, 2005.

[43] Elvia Karina Grillo Ardila, Julián Santaella-Tenorio, Rodrigo Guerrero, and Luis Eduardo Bravo. Mathematical model and covid-19. *Colombia Médica*, 51(2), 2020.

[44] Nicola Guglielmi, Elisa Iacomini, and Alex Viguerie. Delay differential equations for the spatially resolved simulation of epidemics with specific application to covid-19. *Mathematical Methods in the Applied Sciences*, 2021.

[45] Linus Hägg and Eddie Wadbro. Nonlinear filters in topology optimization: existence of solutions and efficient implementation for minimum compliance problems. *Structural and Multidisciplinary Optimization*, 55(3):1017–1028, 2017.

[46] A Harir, S Melliani, H El Harfi, and LS Chadli. Research article variational iteration method and differential transformation method for solving the seir epidemic model. 2020.

[47] John D. Hedengren, Reza Asgharzadeh Shishavan, Kody M. Powell, and Thomas F. Edgar. Nonlinear modeling, estimation and predictive control in APMonitor.

Computers & Chemical Engineering, 70:133 – 148, 2014. Manfred Morari Special Issue.

[48] Leonhard Held, Niel Hens, Philip D O'Neill, and Jacco Wallinga. *Handbook of infectious disease data analysis.* CRC Press, 2019.

[49] Eligius MT Hendrix, G Boglárka, et al. *Introduction to nonlinear and global optimization*, volume 37. Springer, 2010.

[50] Harry Hochstadt. *Differential equations.* Courier Corporation, 2014.

[51] Mark H Holmes. *Introduction to numerical methods in differential equations.* Springer, 2007.

[52] John N Hooker et al. *Integrated methods for optimization*, volume 170. Springer, 2012.

[53] Salisu Ibrahim. Numerical approximation method for solving differential equations. *Eurasian Journal of Science & Engineering*, 6(2):157–168, 2020.

[54] Song Jing and Cao Zhu'an. Application of iterative dynamic programming to dynamic optimization problems. *JOURNAL OF CHEMICAL INDUSTRY AND ENGINEERING-CHINA-*, 50(6):125–129, 1999.

[55] JV Kadam, M Schlegel, W Marquardt, RL Tousain, DH Van Hessem, J van den Berg, and OH Bosgra. A two-level strategy of integrated dynamic optimization and control of industrial processes—a case study. In *Computer Aided Chemical Engineering*, volume 10, pages 511–516. Elsevier, 2002.

[56] Kenneth Lange. *Optimization*, volume 95. Springer Science & Business Media, 2013.

[57] Evgeny Lazutkin, Abebe Geletu, and Pu Li. An approach to determining the number of time intervals for solving dynamic optimization problems. *Industrial & Engineering Chemistry Research*, 57(12):4340–4350, 2018.

[58] Danny J Lohan, Ercan M Dede, and James T Allison. Topology optimization for heat conduction using generative design algorithms. *Structural and Multidisciplinary Optimization*, 55(3):1063–1077, 2017.

[59] David G Luenberger and Yinyu Ye. Linear and nonlinear programming, 2nd and 4th editions, 2016.

[60] Rein Luus. Optimization of fed-batch fermentors by iterative dynamic programming. *Biotechnology and Bioengineering*, 41(5):599–602, 1993.

[61] Lucija Mihalj. *Penalty methods in constrained optimization.* PhD thesis, University of Zagreb. Faculty of Science. Department of Mathematics, 2019.

[62] Somanath Misra. *Global optimization with application to geophysics.* 2008.

[63] Bijan Mohammadi and Olivier Pironneau. Shape optimization in fluid mechanics. *Annu. Rev. Fluid Mech.*, 36:255–279, 2004.

[64] Katta G Murty. *Operations research: deterministic optimization models.* Prentice-Hall, Inc., 1994.

[65] Igor Nesteruk. *COVID-19 Pandemic Dynamics: Mathematical Simulations.* Springer Nature, 2021.

[66] Trung Thanh Nguyen. *Continuous dynamic optimisation using evolutionary algorithms.* PhD thesis, University of Birmingham, 2011.

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

[68] Jorge Nocedal and Stephen J Wright. *Numerical optimization.* Springer, 1999.

[69] Maxwell Noton. *Modern Control Engineering: Pergamon Unified Engineering Series.* Elsevier, 2014.

[70] James M Ortega and Werner C Rheinboldt. *Iterative solution of nonlinear equations in several variables*. SIAM, 2000.

[71] Gaurav Pandey, Poonam Chaudhary, Rajan Gupta, and Saibal Pal. Seir and regression model based covid-19 outbreak predictions in india. *arXiv preprint arXiv:2004.00958*, 2020.

[72] Konstantinos E Parsopoulos and Michael N Vrahatis. *Particle swarm optimization and intelligence: advances and applications: advances and applications*. IGI global, 2010.

[73] Nathaniel Peters, Martin Guay, and Darryl DeHaan. Real-time dynamic optimization of batch systems. *Journal of Process Control*, 17(3):261–271, 2007.

[74] Elijah Polak. *Optimization: algorithms and consistent approximations*, volume 124. Springer Science & Business Media, 2012.

[75] GP Pollard and RWH Sargent. Off line computation of optimum controls for a plate distillation column. *Automatica*, 6(1):59–76, 1970.

[76] Iman Rahimi, Amir H Gandomi, Panagiotis G Asteris, and Fang Chen. Analysis and prediction of covid-19 using sir, seiqr and machine learning models: Australia, italy and uk cases. *Information*, 12(3):109, 2021.

[77] Karthik Raman. *An Introduction to Computational Systems Biology: Systems-Level Modelling of Cellular Networks*. Chapman and Hall/CRC, 2021.

[78] RT Rockafellar. Fundamentals of optimization. *Lecture Notes 2007*, 2006.

[79] Cornelis Roos. A full-newton step o (n) infeasible interior-point algorithm for linear optimization. *SIAM Journal on Optimization*, 16(4):1110–1136, 2006.

[80] Shahrzad Saremi, Seyedeh Zahra Mirjalili, and Seyed Mohammad Mirjalili. Evolutionary population dynamics and grey wolf optimizer. *Neural Computing and Applications*, 26(5):1257–1263, 2015.

[81] Adil R Sarhan, Mohammed H Flaih, Thaer A Hussein, and Khwam R Hussein. Novel coronavirus (covid-19) outbreak in iraq: the first wave and future scenario. *medRxiv*, 2020.

[82] Christopher Y Shen. Logistic growth modelling of covid-19 proliferation in china and its international implications. *International Journal of Infectious Diseases*, 96:582–589, 2020.

[83] Jan A Snyman and Daniel N Wilke. Practical mathematical optimization: Basic optimization theory and gradient-based algorithms, oλ. 133, 2018.

[84] Suvrit Sra, Sebastian Nowozin, and Stephen J Wright. *Optimization for machine learning*. Mit Press, 2012.

[85] Balasubramaniam Srinivasan, Srinivas Palanki, and Dominique Bonvin. Dynamic optimization of batch processes: I. characterization of the nominal solution. *Computers & Chemical Engineering*, 27(1):1–26, 2003.

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

[87] Hamdy A Taha. *Operations research: an introduction*, volume 790. Pearson/Prentice Hall Upper Saddle River, NJ, USA, 2011.

[88] KL Teo, Y Liu, WR Lee, LS Jennings, and S Wang. Numerical solution of an optimal control problem with variable time points in the objective function. *The Anziam Journal*, 43(4):463–478, 2002.

[89] EF Toro. Numerical methods for hyperbolic equations: Theory and applications.

[90] Fredi Tröltzsch. *Optimal control of partial differential equations: theory, methods, and applications*, volume 112. American Mathematical Soc., 2010.

[91] Robert J Vanderbei et al. *Linear programming*. Springer, 2020.

[92] Pandian M Vasant. *Meta-heuristics optimization algorithms in engineering, business, economics, and finance.* IGI Global, 2012.

[93] Elena Vázquez-Cendón, Arturo Hidalgo, Pilar Garcia Navarro, and Luis Cea. *Numerical methods for hyperbolic equations.* CRC Press, 2012.

[94] Andreas Wächter and Lorenz T Biegler. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical programming*, 106(1):25–57, 2006.

[95] Zhaoxuan Wang, Zhiyu Yu, and Changlin Tian. Cruz-rodriguez l, sanchez batista l and bin zhao.“. *Differential equation analysis on COVID-19*, 2020.

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

[97] Xin-She Yang. *Nature-inspired optimization algorithms.* Academic Press, 2020.

[98] Xin-She Yang, Gebrail Bekdaş, and Sinan Melih Nigdeli. *Metaheuristics and optimization in civil engineering.* Springer, 2016.

[99] Danial Yazdani. *Particle swarm optimization for dynamically changing environments with particular focus on scalability and switching cost.* Liverpool John Moores University (United Kingdom), 2018.

[100] Chunlei Zhang and Raúl Ordóñez. *Extremum-seeking control and applications: a numerical optimization-based approach.* Springer Science & Business Media, 2011.

[101] Bin Zhao and Jinming Cao. Statistical and equation model analysis on covid-19. 2020.

[102] Minghui Zhu and Sonia Martinez. On distributed convex optimization under inequality and equality constraints. *IEEE Transactions on Automatic Control*, 57(1):151–164, 2011.

[103] Mu Zhu, Yang Yang, James K Guest, and Michael D Shields. Topology optimization for linear stationary stochastic dynamics: applications to frame structures. *Structural Safety*, 67:116–131, 2017.

[104] Dennis G Zill. *Differential equations with boundary-value problems.* Cengage Learning, 2016.

الخلاصة

في هذه الدراسة، تم عرض مشاكل التحليل العددي والتحسين في فئة المعادلات التفاضلية الوظيفية، والتي تعد أدوات مهمة في الرياضيات التطبيقية. حيث تم تطوير النموذج الرياضي SIR (S = المعرضين للوباء، I =المصابين، R = المتعافين أو المتوفين) المستخدم مع كوفيد-19 كدالة موضوعية مع القيود المفروضة في العراق وفقا للبيانات المأخوذة من دائرة صحة كربلاء لعام 2021، وكذلك تم استخدام طريقة أويلر للعثور على النتائج من سرعة انتشار الوباء إلى سرعة الشفاء منه. تم إدخال طريقة عددية جديدة على أساس نهج جديدة لخوارزميات التحسين، ومشاكل التحسين الديناميكي العامة، والتحكم الأمثل، حيث تم استخدام كل من خوارزمية طريقة النقطة الداخلية وخوارزمية تحسين الذئاب الرمادية (GWO) لتحسين النظم في العمليات الكيميائية المعتمدة. كانت لغة البرمجة الرئيسية المستخدمة في هذه الدراسة هي لغة بايثون، حيث لوحظت النتائج التي تم الحصول عليها بشكل عام دقيقة وقوية وحققت الأهداف المرجوة التي تم الوصول إليها مقارنة بنتائج الدراسات السابقة.

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

نهج جديد للتحسين العددي مع المعادلات التفاضلية

رسالة

مقدمة الى مجلس كلية التربية للعلوم الصرفة / جامعة بابل وهي جزء من متطلبات نيل درجة الماجستير في التربية / الرياضيات

من قبل الطالبة

رباب عبد الستار محسن جواد

بإشراف

أ.م. د. احمد صباح احمد الجيلاوي

1444 هـ　　　　2022 م