

https://dspace.org/

Developing insight into the optimization element in a nonlinear model predictive control context

Dhoodhat, Zaheer

2014-09-03
http://hdl.handle.net/10539/15377

Downloaded from a DSpace repository. https://dspace.org/

Developing Insight into the Optimization Element in a Nonlinear Model Predictive Control Context

Zaheer Dhoodhat

A research report submitted to the Faculty of Engineering and the Built Environment, University of the Witwatersrand, Johannesburg, in partial fulfilment of the requirements for the degree of Master of Science in Engineering.

Johannesburg, 2014

The financial assistance of the National Research Foundation (NRF) towards this research is hereby acknowledged. Opinions expressed and conclusions arrived at, are those of the author and are not necessarily to be attributed to the NRF.

Declaration

I declare that this research report is my own unaided work. It is being submitted to the Degree of Master of Science to the University of the Witwatersrand, Johannesburg. It has not been submitted before for any degree or examination to any other University.

_____

(Signature of Candidate)

_____day of _____, _____

(day) (month) (year)

Abstract

Optimization is one of the fundamental components in Model Predictive Control (MPC) and Nonlinear Model Predictive Control (NMPC). In NMPC the optimization problem that is to be solved can be non-convex which is a challenging problem to solve. Having insight into the optimization component of the NMPC algorithm will offer value to the control engineers designing and using NMPC controlled systems. This study presents an approach, referred to as the Optimization Roadmap, that graphically provides insight or transparency into the optimization element within NMPC. The methodology was applied to several examples to ratify the insights gained. Two optimization algorithms, the gradient based Sequential Quadratic Programming (SQP) algorithm and the meta-heuristic Particle Swarm Optimization (PSO), were employed within the NMPC algorithm and their characteristics contrasted. The application of the methodology to the examples revealed that the Optimization Roadmap provides useful insights into the optimization problem to the user. These insights include the convexity or non-convexity of the problem and additionally regions of local minima, if present. The Optimization Roadmap additionally provides insights into areas in which initial conditions, for local optimization methods, could be chosen for the best results. Furthermore, the results show that the local optimization algorithm, SQP, performs much faster than the PSO algorithm. More importantly, by using the Optimization Roadmap to select favourable initial conditions for the SQP algorithm led to it producing results in the vicinity of, if not equal to, those obtained by the PSO algorithm.

Dedication

To my family.

Acknowledgements

Firstly, I would like to thank the Almighty for the innumerable blessings, the opportunity to undertake this study and the ability to complete this research report.

I would also like to sincerely thank my supervisor, Professor Brian Wigdorowitz, for his unrelenting assistance, guidance and encouragement during the tenure of this study.

I cannot thank my family enough for all their support and understanding during this study. My parents Shehra Banoo and Yusuf, my wife and son, Safiya and Uthmaan, and my parents-in-law Rashida and Ismail Vachiat.

Contents

Declaration i

Abstract ii

Acknowledgements iv

Contents v

List Of Figures viii

List Of Tables xi

1 Introduction 1

1.1 Background . . . 1
1.2 Literature Review . . . 2
1.3 Research Question . . . 3
1.4 Proposed Solution . . . 3
1.5 Research Results . . . 5
1.6 Report Structure . . . 5

2 Background Theory 6

2.1 Convexity . . . 6
2.2 Optimization . . . 9
2.2.1 Sequential Quadratic Programming . . . 10
2.2.2 Particle Swarm Optimization . . . 13
2.3 Model Predictive Control . . . 16
2.3.1 Simple Formulation of MPC . . . 20
2.4 Nonlinear Model Predictive Control . . . 21

3 Literature Survey 23

3.1 NMPC Formulations and Optimization . . . 23
3.2 NMPC with the PSO Algorithm . . . 26
3.3 NMPC with the SQP Optimization Algorithm . . . 27

3.4 Insight . . . 27
3.4.1 Nonlinearity Quantification and Measure . . . 27
3.4.2 Optimization Visualisation . . . 28

4 NMPC Implementation **29**
4.1 Formulation of NMPC . . . 29
4.2 Algorithm Discussion . . . 31
4.2.1 Discretization . . . 31
4.2.1.1 Full Discretization[1] (Simultaneous Method) . . . 32
4.2.1.2 Recursive Discretization[1] (Sequential Method) . . . 33
4.2.2 Optimization . . . 33
4.2.2.1 SQP (fmincon[2]) . . . 34
4.2.2.2 PSO . . . 34
4.3 NMPC Algorithm Implementation [1] . . . 37
4.3.1 Step 1: Initialisation . . . 37
4.3.2 Step 2: Solving The Optimal Control Problem . . . 37
4.3.3 Step 3: Applying The Control Input . . . 39

5 Optimization Roadmap Methodology and Results **42**
5.1 Computation Platform . . . 43
5.2 Static Optimization . . . 43
5.2.1 Unconstrained Static Optimization . . . 44
5.2.1.1 Optimization Roadmap For Unconstrained Static Optimization . . . 44
5.2.1.2 Example 1: Unconstrained Static Optimization of a 1-Dimensional Quadratic Function . . . 44
5.2.1.3 Example 2: Unconstrained Static Optimization of a 2-Dimensional Quadratic Function . . . 47
5.2.1.4 Example 3: Unconstrained Static Optimization of a 2-Dimensional Rastrigin Function . . . 49
5.2.2 Constrained Static Optimization . . . 56
5.2.2.1 Optimization Roadmap For Constrained Static Optimization . . . 56
5.2.2.2 Example 1: Nonlinear Inequality Constrained Static Optimization of a 1-Dimensional Quadratic Function . . . 56
5.2.2.3 Example 2: Nonlinear Inequality Constrained Static Optimization of a 2-Dimensional Quadratic Function . . . 59
5.2.2.4 Example 3: Nonlinear Equality Constrained Static Optimization of a 2-Dimensional Quadratic Function . . . 65
5.3 NMPC . . . 70
5.3.1 NMPC Optimization: Optimization Roadmap Methodology . . . 70
5.3.2 NMPC Example 1: Linear System with a Quadratic Cost Function . . . 75

5.3.3 NMPC Example 2: Nonlinear System with a Quadratic Cost Function 81
5.3.3.1 NMPC Example 2 with Additional Constraint 92
5.3.4 NMPC Example 3: Van der Pol System with a Quadratic Cost Function . . . 95
5.3.4.1 Van der Pol System with Asymptotically Stable Dynamics 96
5.3.4.2 Van der Pol System with Limit Cycle Characteristics 101
5.4 Results Overview 109

6 Conclusions **111**

Appendix A **114**

Appendix B **131**

List of Figures

2.1 Convex Sets . . . 7
2.2 Non-convex Sets . . . 7
2.3 Epigraph of a Function . . . 8
2.4 A Convex Function . . . 8
2.5 A Non-convex Function . . . 9
2.6 PSO Star Topology . . . 16
2.7 Overview of MPC Algorithm (Adapted from [3] and [4]) . . . 19
2.8 Overview of MPC Structure (Adapted from [5]) . . . 20

4.1 Local and Global Minima Test Function for SQP . . . 35
4.2 Sequential Discretization . . . 38
4.3 Shift Method . . . 39
4.4 NMPC Time Line (Adapted from [1]) . . . 40
4.5 NMPC Time Line (Adapted from [1]) . . . 40

5.1 1-Dimensional Quadratic Cost Function . . . 45
5.2 Unconstrained 1-Dimensional Quadratic Function - Cost Function Value vs. Independent variable (x) . . . 45
5.3 2-Dimensional Quadratic Cost Function . . . 48
5.4 2-Dimensional Unconstrained Quadratic Function - Cost Function Value vs. Norm . 49
5.5 2-Dimensional Rastrigin Function . . . 50
5.6 2-Dimensional Rastrigin Function- Cost Function Value vs. Norm with increments of 1 . . . 51
5.7 2-Dimensional Rastrigin - Cost Function Value vs. Norm Plot depicting curvature of the function . . . 52
5.8 2-Dimensional Rastrigin Function - Cost Function Value vs. Norm plot illustrating a local minimum trap . . . 52
5.9 2-Dimensional Rastrigin - Cost Function Value vs. Norm Plot Illustrating All Curves 53
5.10 2-Dimensional Rastrigin - Cost Function Value vs. Norm Plot Illustrating Overlapping Norms . . . 54

5.11 2-Dimensional Rastrigin - Cost Function Value vs. Norm Plot Illustrating a Local Minimum Trap 54
5.12 1-Dimensional Quadratic Cost Function With a Nonlinear Constraint Function . . . 58
5.13 1-Dimensional Quadratic Cost Function Inequality Constrained by a Nonlinear Function 58
5.14 1-Dimensional Constrained Quadratic Function - Cost Function Value vs. Norm . . 59
5.15 2-Dimensional Quadratic Cost Function with Nonlinear Constraint 61
5.16 2-Dimensional Quadratic Cost Function with Nonlinear constraint 62
5.17 2-Dimensional Quadratic Cost Function with Nonlinear Constraint - Cost versus Norm Plot 62
5.18 2-Dimensional Quadratic Cost Function with Nonlinear Constraint Function: Points must lie on constraint function 67
5.19 2-Dimensional Quadratic Cost Function with Nonlinear Constraint Function (Zoomed) 67
5.20 Example 1: Cost vs. Time Plot for SQP with Initial $u = -10$, $N = 2$ 80
5.21 Example 1: Cost vs. Time Plot for PSO with Initial $u = -10$ and Horizon $N = 2$ . . 80
5.22 NMPC Example 2: Plots showing the Global Minimum Path for 4 Time Intervals . . 85
5.23 Cost vs. Time Plot for SQP u stuck at $+5$, $N = 2$ 86
5.24 Cost vs. Time Plot for SQP with u stuck at -5, $N = 2$ 86
5.25 Cost vs. Time Plot for PSO with Initial $u = -5$, $N = 2$ 87
5.26 Cost vs. Time Plot for SQP with Initial $u = 2$, $N = 2$ 89
5.27 Cost vs. Time Plot for SQP with Initial $u = 1.2$, $N = 2$ 90
5.28 Cost vs. Time Plot for SQP with Initial $u = -5$, $N = 5$ 90
5.29 Cost vs. Time Plot for SQP with Initial $u = 5$, $N = 5$ 91
5.30 Cost vs. Time Plot for SQP with Initial $u = 1.2$, $N = 5$ 91
5.31 Cost vs. Time Plot for PSO with Initial $u = 5$, $N = 5$ 91
5.32 Cost vs. Time Plot for SQP with Additional Constraint with Initial $u = 1.2$, $N = 2$ 95
5.33 Phase Portrait For Asymptotically Stable Van der Pol Oscillator 96
5.34 Cost vs. Time Plot for Van der Pol SQP with Initial $u = -0.9$, $N = 2$ 100
5.35 Phase Portrait For Van der Pol Oscillator - Limit Cycle 101
5.36 Simulation For Van der Pol Oscillator - Limit Cycle 102
5.37 Cost vs. Time Plot for Van der Pol SQP with Initial $u = -1$ 106
5.38 Cost vs. Time Plot for Van der Pol PSO 106
5.39 Cost vs. Time Plot for Van der Pol SQP with Initial $u = 0.5$ 106
5.40 Cost vs. Time Plot for Van der Pol using SQP with Initial $u = -1$, $N = 5$ 107
5.41 Cost vs. Time Plot for Van der Pol using SQP with Initial $u = 0.2$, $N = 5$ 107
5.42 Cost vs. Time Plot for Van der Pol using SQP with Initial $u = 0.7$, $N = 5$ 108

A.1 Flow Chart of Static Unconstrained Optimization Roadmap Strategy 115
A.2 Flow Chart of Static Constrained Optimization Insight Strategy 116
A.3 Flow Chart of NMPC Algorithm [1] 118

A.4 Flow Chart of measureInitialValue Function [1] 119
A.5 Flow Chart of solveOptimalControlProblem Function [1] 120
A.6 Flow Chart of computeOpenloopSolution Function [1] 122
A.7 Flow Chart of costfunction Function [1] 123
A.8 Flow Chart of dynamic Function [1] 124
A.9 Flow Chart of SolveOptimizationProblem Function [1] 126
A.10 Flow Chart of nonlinearconstraints Function [1] 128
A.11 Flow Chart of applyControl Function [1] 129
B.1 Example 1: Cost vs. Time Plot for PSO 132
B.2 Example 1: Cost vs. Time Plot for SQP with Initial $u = -10$ 132
B.3 Cost vs. Time Plot for Van der Pol SQP with Horizon $N = 5$ Initial $u = 0.9$ 134
B.4 Phase Plot for Van der Pol depicting Limit Cycle $u = -1$ 135
B.5 Cost vs. Time Plot for Van der Pol SQP with Horizon 5 Initial $u = 0$ 135
B.6 Figure Showing the Path of the SQP algorithm in Limit Cycle Van der Pol with Initial $u = -1$ 136
B.7 Figure Showing the Path of the SQP algorithm in Limit Cycle Van der Pol with Initial $u = 1$ 137
B.8 Figure Showing the Path of the PSO algorithm in Limit Cycle Van der Pol 138

List of Tables

2.1 Table of MPC Vendors and Products [6] [5] 17
2.2 Examples of MPC strategies and Their Models [5] 18
2.3 Table of NMPC Vendors and Products [7] 21

4.1 Table showing the results of the SQP optimization of Figure 4.1 36

5.1 Optimization of 1-Dimensional Unconstrained Quadratic Function using SQP 46
5.2 Optimization of 1-Dimensional Unconstrained Quadratic Function using PSO 47
5.3 Optimization of 2-Dimensional Unconstrained Quadratic Function using SQP 49
5.4 Optimization of 2-Dimensional Unconstrained Quadratic Function using PSO 50
5.5 Optimization of 2-Dimensional Unconstrained Rastrigin Function using SQP 55
5.6 Optimization of 2-Dimensional Unconstrained Rastrigin Function using PSO 55
5.7 Optimization of 1-Dimensional Nonlinear Constrained Quadratic Function using SQP 60
5.8 Optimization of 2-Dimensional Nonlinear Constrained Quadratic Function using SQP 63
5.9 Table Showing x_1 and x_2 values Generating the Same Norm Value 64
5.10 Optimization of 2-Dimensional Nonlinear Constrained Quadratic Function using PSO 65
5.11 Optimization of 2-Dimensional Nonlinear Constrained (On Function) Quadratic Function using SQP 68
5.12 Optimization of 2-Dimensional Nonlinear Constrained (On Function) Quadratic Function using PSO 69
5.13 Illustration of the Optimization Roadmap 74
5.14 Table Depicting the Optimization Roadmap Methodology in Time Intervals for NMPC Example 1 78
5.15 NMPC Example 1: NMPC Algorithm Results using SQP with Initial $u = -10$ and Horizon $N = 2$ 79
5.16 NMPC Example 1: NMPC Algorithm Results using PSO with Initial $u = -10$, $N = 2$ 79
5.17 Table Depicting the Optimization Roadmap Methodology in Time Intervals for NMPC Example 2 83
5.18 NMPC Algorithm Results using SQP with Initial $u = 5$, $N = 2$ 85
5.19 NMPC Algorithm Results using SQP with Initial $u = -5$, $N = 2$ 86
5.20 NMPC Algorithm Results PSO with Initial $u = -5$, $N = 2$ 87

5.21 NMPC Algorithm Results PSO with Initial $u = 5$, $N = 2$ 88
5.22 NMPC Algorithm Results SQP with Initial $u = 2$, $N = 2$ 89
5.23 NMPC Algorithm Results using SQP with Initial Point for $u = 1.2$, $N = 2$ 89
5.24 NMPC Algorithm Results using SQP with Initial $u = 1.2$, $N = 5$ 92
5.25 Table Depicting the Optimization Roadmap Methodology in Time Intervals for NMPC Example 2 with an Additional Constraint 94
5.26 NMPC Example 2: NMPC Algorithm Results using SQP Optimization with Additional Constraint with Initial $u = 1.2$, $N = 2$ 95
5.27 Table Summarising the Optimization Roadmap Methodology for Asymptotically Stable Van der Pol Example with $\mu = -2$, State Initial Condition $= [0.2, 0.5]$, $T = 3$ . . 98
5.28 NMPC with SQP Optimization Algorithm with Asymptotically Stable Van der Pol System with Initial $u = -0.9$, $N = 2$ 99
5.29 NMPC with PSO Optimization Algorithm with Asymptotically Stable Van der Pol System with Initial $u = -0.9$, $N = 2$ 99
5.30 NMPC with SQP Optimization Algorithm with Asymptotically Stable Van der Pol System with Initial $u = -0.5$, $N = 2$ 99
5.31 Table Depicting the Optimization Roadmap Methodology in Time Intervals for NMPC Example with Van der Pol System with Limit Cycle and $\mu = 1$ State Initial Condition $[1, 1]$ and $T = 3$ 104
5.32 NMPC SQP Optimization of Quadratic Cost Function with Van der Pol (Limit Cycle behaviour) as constraint and Initial $u = -1$, $N = 2$ 105
5.33 NMPC Algorithm Results using PSO algorithm with Van der Pol System (Limit Cycle behaviour) as the constraint and Initial $u = -1$, $N = 2$ 105
5.34 NMPC Algorithm Results using SQP optimization with Van der Pol System (Limit Cycle behaviour) as the constraint and Initial $u = 0.5$, $N = 2$ 105
5.35 NMPC Algorithm Results using SQP optimization with Van der Pol System (Limit Cycle behaviour) as the constraint and Initial $u = -1$, $N = 5$ 107
5.36 NMPC Algorithm Results using SQP Van der Pol System (Limit Cycle behaviour) as the constraint and Initial $u = 0.7$ and $N = 5$ 108

B.1 NMPC Algorithm Results SQP with Initial $u = -10$, $N = 5$ 131
B.2 NMPC Algorithm Results PSO with Initial $u = -10$, $N = 5$ 131
B.3 NMPC Algorithm Results using SQP Example 2 with Initial $u = 5$, $N = 5$ 133
B.4 NMPC Algorithm Results using SQP Example 2 with Initial $u = -5$, $N = 5$ 133
B.5 NMPC with SQP Optimization Algorithm of Quadratic Cost Function with Van der Pol Asymptotically Stable Dynamics with Initial $u = 0.9$, $N = 5$ 134
B.6 NMPC Algorithm Results using SQP For Van der Pol with Limit Cycle with Initial $u = 0$, $N = 5$ 135
B.7 NMPC SQP Optimization of Quadratic Cost Function for Van der Pol System with Limit Cycle behaviour with Initial $u = 1$, $N = 2$ 139

Chapter 1

Introduction

1.1 Background

The vast field of control systems with its ubiquitous applications, ranging from controlling traffic signals ([8]) to the launching of rockets ([9]) and to applications in socio-economic systems ([10]), plays an important role in our daily functioning. Within this vast array of applications, a number of control techniques or methodologies are used. Examples of these control techniques are Proportional Integral Derivative (PID) control, Adaptive Control, Optimal Control and Model Predictive Control (MPC).

MPC is an advanced control strategy that has been in existence (in some form) since the 1960s [11]. It has since become a widely used control technique in the process industry mainly due to a few attractive and fundamental features such as the explicit use of a system model and the ability to effectively handle constraints [5]. The basic concept of MPC is that a system model is used to generate a prediction of the system trajectory over a prediction horizon which is then followed by the optimization of a chosen objective function. This then generates a sequence of control inputs of which the first (control input) is applied at the next time interval. This procedure is then repeated. MPC has also found applications in several industries (other than the process industry) including the automotive and aerospace industries [6] but it is limited to the use of linear system models.

In [5] the MPC strategy is described as being analogous to the strategy of driving a car since using the knowledge of the desired (reference) trajectory and a *mental* model of the car, appropriate control inputs can be applied in order to direct the vehicle towards the reference trajectory. These control inputs being acceleration, braking or steering of the vehicle are applied in a receding horizon manner where only the first control input is applied at each time instant. The procedure is then repeated in order to obtain the control input at the next time instant.

Nonlinear Model Predictive Control (NMPC) is an extension of MPC which uses nonlinear system models and/or constraints. It has received much attention in recent years with a growing number

of applications in practise [7]. This increase in the number of NMPC applications is largely due to the inherent nature of systems being nonlinear coupled with the requirement for improved or more accurate system models. Other factors that have led to the increasing application of NMPC include economic factors where there is a constant push for increased profits and plants are required to perform at their limits in order to increase or improve output yield.

1.2 Literature Review

The solution of an optimization problem is an important step in MPC (in order to obtain the control input). A widely accepted fact is that a nonlinear system model has a significant effect on the optimization phase within the MPC algorithm ([7], [12],[5]). The optimization problem changes from an easily solved convex Linear Program (LP) or Quadratic Program (QP) in MPC, to a complex non-convex optimization problem in NMPC ([5],[7],[13],[14]). It should be noted that the non-convexity in NMPC could also occur as a result of the cost function having non-convex characteristics. The effect of the nonlinear system model has several important consequences which include:

- Increased complexity;
- Increased computational expense;
- Increased computation time;
- Local versus global solutions to the optimization problem (optima).

A quote from Mayne in [15], on the *"online solution of non-convex optimal control problem"* relating to the first two points in the list above states that *"failure to respond to this challenge will severely inhibit the use of Nonlinear Model Predictive Control"*. Although substantial developments have been made in NMPC and available computing hardware, the concept of the computational burden and optimal solution of the optimization problem remains an imperative concern [16][17]. The study by Borrelli et al. in [18] shows the effect that local optima have on the cost of a system (HVAC in this case). It was found in this study ([18]) that, for a specific configuration, the minima had considerably varying energy costs which translate into monetary costs thus leading to the conclusion that finding the global optimum has substantial financial consequences.

A significant amount of research has been undertaken in the field of NMPC to address the challenges faced by NMPC including that of the non-convex optimization problem and the consequent computational burden. Several different formulations of NMPC such as global NMPC ([19]), advanced-Step ([17]) NMPC and offline NMPC [20] have been formulated which have different approaches to solving the NMPC problem. A multitude of optimization algorithms have also been implemented

in NMPC formulations with the view of taking advantage of the inherent characteristics of the optimization method in order to expedite the solving of the optimization problem. Examples of some of the optimization algorithms that have been investigated with NMPC include interval analysis ([21]), Particle Swarm Optimization (PSO) [22], Sequential Quadratic Programming (SQP) [23] and Genetic Algorithms (GA) [24].

1.3 Research Question

The central and important component of the NMPC algorithm, optimization, is often considered as a "black box" system where users provide the required inputs and assume that the output received is reasonably close to the optimal value. In the context of NMPC the optimization problems that are to be solved are often challenging, non-convex problems. The traditional approach of accepting the optimization problem as a "black box" limits the understanding of the problem and what the best approach in solving the problem in an optimal manner would be. The best approach in solving the optimization problem could be the choice of optimization algorithm that is used or the identification of problem areas and how they are to be dealt with i.e. possibly by intelligently choosing initial conditions.

The research question that is to be addressed in this study is; Given the challenge of the (possibly non-convex) optimization problem and the benefits of finding an optimal solution in NMPC, how can one provide insight into the optimization problem in NMPC?

In a lecture about two and a half decades ago Dr. Gunter Stein made a statement that is still relevant today, if not more so. In his lecture Dr. Stein mentioned of the growing trend of *" increasing worship of abstract mathematical results in control at the expense of more specific examinations of their practical, physical consequences"* [25]. He mentioned this as one of the trends *"that threatens to undermine the achievement"* of the control research community. In this vein, this study is primarily concerned with providing insight and understanding into the optimization problem that is to be solved in NMPC while the underlying mathematical theory lies within the optimization and NMPC formulation.

1.4 Proposed Solution

Bearing the observations and statements by Dr. Stein in mind, the aim of this research is to address the problem of optimization (in the NMPC context) being regarded as a "black box" by providing a graphical means of presenting the control engineer or designer with insight or transparency into the optimization. This conceptually simple yet powerful approach is aimed at empowering designers

and engineers with insight into, and an understanding of, the optimization problem. Currently no references or research directly related to this study could be found in terms of graphically providing insight into the optimization problem to be solved in NMPC.

This study will be concerned with developing and proposing a methodology for single-input systems that furnishes graphical insight and transparency into the optimization component within NMPC such that control engineers and others working on the system will have a visual aid providing an understanding of the system and its operation in an NMPC context. The graphical tool will empower engineers in that they will have an *a priori* view into regions of non-convexity and local minima; and thus will be able to make educated decisions regarding the operation of the NMPC controller. The effect of constraints on the optimization element within NMPC can also be elicited from the methodology.

The Optimization Roadmap methodology is applied to several examples which prove the applicability and highlight various aspects of the proposed approach. The key aspect of the methodology is the use of the independent variable (control inputs) on one axis and the cost function value on the other to generate the graphs (2-Dimensional) such that the information is presented succinctly. Thus, the number of states of the system does not increase the complexity of the graphs but will influence the computation of the cost function value if the states are part of the cost function.

An extension of the car driving strategy analogy described earlier, which is aligned with this study, is the inclusion of optimization; it being a fundamental component of MPC and the driving force behind the control inputs. Thus, analogously, this study aims to provide an "*Optimization Roadmap*" or in other words a map that provides information (graphically) on the road/terrain topography or conditions. Relating this back to optimization, the conditions and topography of the road equates to the convexity or non-convexity characteristics of the optimization problem. Thus, rugged or off-road terrain with steep inclines and declines would be associated with a non-convex optimization problem which is difficult to solve while a smooth, flat, tarred highway would be associated with a convex optimization problem.

In addition to the Optimization Roadmap methodology, this study applies an NMPC algorithm to the examples as a means of evaluating the methodology. This investigation will use two optimization methods in the NMPC algorithm (separately); one *gradient-based* optimization method (local optimization) and one *meta-heuristic* optimization method (global optimization) in order to investigate the trade-offs between the two. The methodology will be used to aid or motivate for the selection of the most appropriate optimization algorithm (of the two) in the examples. The *gradient-based* optimization method that will be investigated is the Sequential Quadratic Programming (SQP) and the *meta-heuristic* optimization method is Particle Swarm Optimization (PSO)[1],

[1]PSO is used in this study due to its meta-heuristic nature and ability to escape local minima although it is not

both of which have been used in NMPC before (example [23],[27]). This investigation will be limited to single-input systems.

1.5 Research Results

The proposed methodology is applied to several examples to investigate its applicability, usefulness and contribution to obtaining insight into the optimization problem. The results firstly validate the applicability of the methodology and additionally demonstrate that clear and valuable information is presented. Important aspects related to the optimization such as local minima regions are made visible.

1.6 Report Structure

This report is structured as follows. Chapter 2 provides background and theoretical preliminaries for MPC, NMPC, SQP and PSO. In Chapter 3, a literature survey is presented on research that has been conducted relating to the graphical tools for optimization, quantification of nonlinearities and applications of NMPC with the two optimization algorithms, SQP and PSO. Chapter 4 addresses the implementation of the various components within the NMPC algorithm that was used in the investigation with a discussion of the software. Chapter 5 presents the Optimization Roadmap methodology proposed as a solution to the problem of graphically providing insight into the optimization as well as examples with results. Chapter 6 provides concluding remarks and possibilities for future work to build and improve on this work.

immune to local minima traps [26]

Chapter 2

Background Theory

In this Chapter, the theoretical preliminaries for the topics relevant to this study are presented. Firstly, the definition and concept of convexity is described since it underpins an important challenge faced by Nonlinear Model Predictive Control (NMPC) algorithms, which is the solution of a non-convex optimization problem. Secondly, a section on optimization, which is central to the Model Predictive Control (MPC) algorithm, is included. This section introduces the constrained optimization problem followed by a high level presentation of the theory behind the two optimization methods that were employed in this study (SQP and PSO). Finally, MPC and NMPC are introduced.

2.1 Convexity

Convexity is a significant concept in optimization and hence in NMPC since it provides a sense of the *difficulty* of the optimization problem. The benefits of this knowledge will become apparent in the next section on optimization when a fundamental property of convexity is described. [28] states that a convex set can be seen as *"a set with the property that all points on the line segment joining **any** two points in the set are also in the set"*. Additionally, [28] provides the mathematical equivalent of the aforementioned property of convexity which states that *"a set C is convex if*

$$x = \mu x_1 + (1-\mu)x_2 \in C \qquad \forall x_1, x_2 \in C \quad and \quad 0 \leq \mu \leq 1." \tag{2.1}$$

Figure 2.1 shows examples of convex sets, while figure 2.2 contains simple examples of sets that are non-convex. These examples can be easily inspected to verify their convexity and non-convexity characteristics by applying the intuitive *definition* provided.

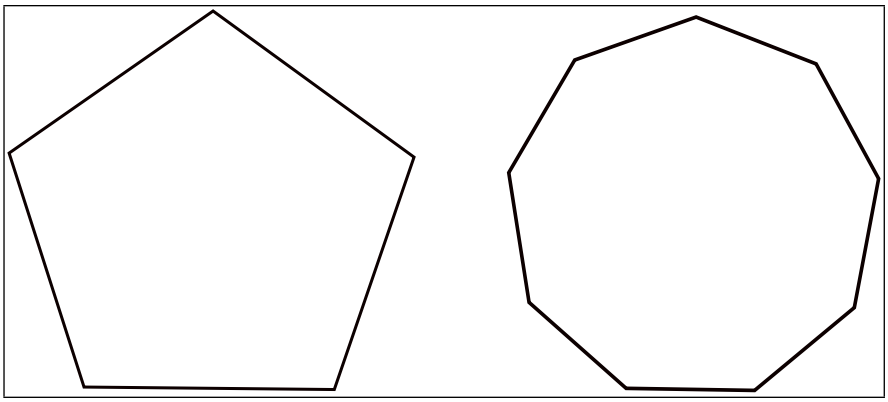


Figure 2.1: Convex Sets

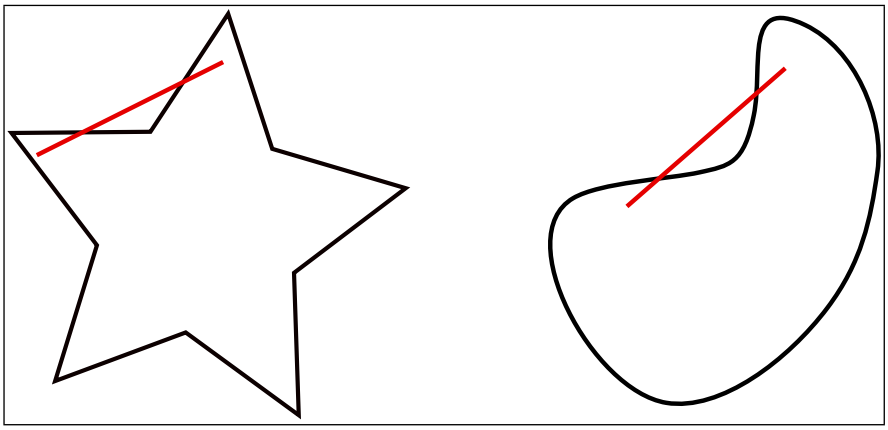


Figure 2.2: Non-convex Sets

Proceeding from sets to functions, the definition of a convex function is given in [29], as "*A function* $f : R^n \to R$ *is convex if the domain of* f ($\mathbf{dom}\, f$) *is a convex set and if for all* $x, y \in \mathbf{dom} f$, *and* θ *with* $0 \leqslant \theta \leqslant 1$, *we have*

$$f(\theta x + (1-\theta)y) \leqslant \theta f(x) + (1-\theta) f(y)". \tag{2.2}$$

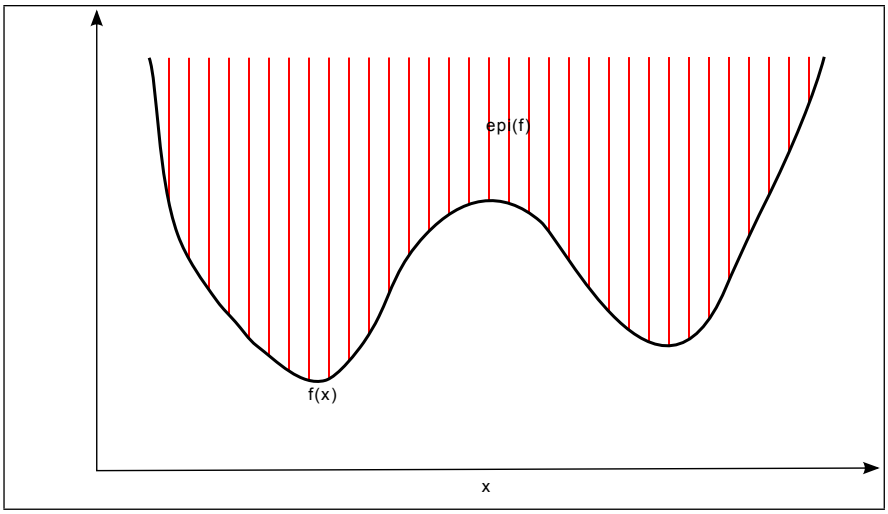


Figure 2.3: Epigraph of a Function

The concept of the *epigraph* of a function is often used to define the convexity of a function. The epigraph of a function f is mathematically defined as

$$epi(f) = \{(x, y) : f(x) \leqslant y\} \text{ where } f : \mathbb{R}^n \to \mathbb{R}. \tag{2.3}$$

However, it can be intuitively thought of as the *"points above the graph"* (shown in Figure 2.3). The rule which provides the link between a convex function and a convex set is that "*a function is convex if and only if its epigraph is a convex set*".

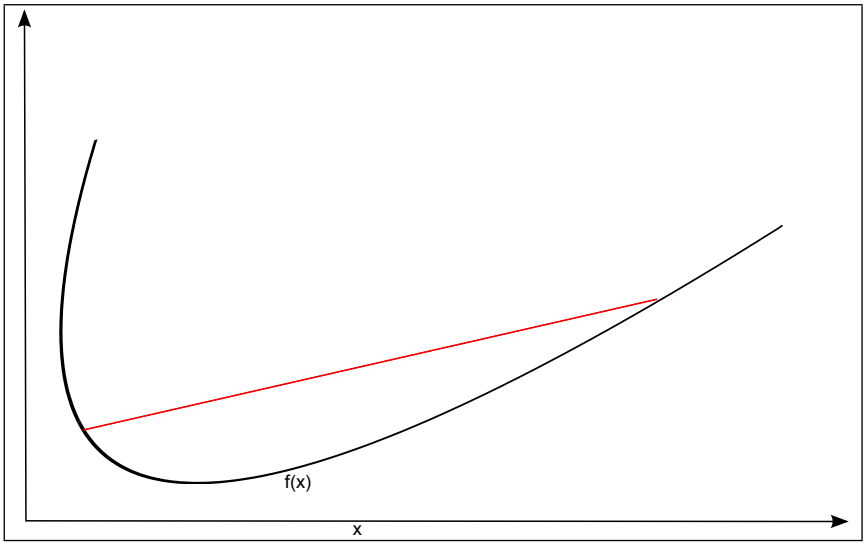


Figure 2.4: A Convex Function

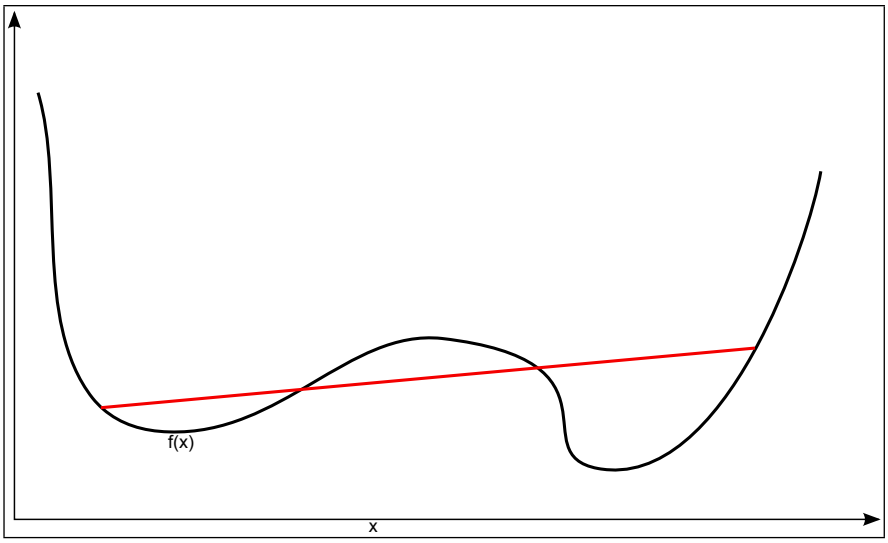


Figure 2.5: A Non-convex Function

2.2 Optimization

In this section, firstly the general constrained Nonlinear Programming (NLP) optimization problem is described with definitions for local and global minima. This is followed by concise background theory to the two optimization strategies (Sequential Quadratic Programming and Particle Swarm Optimization) used in this study. The goal of this section is not to present an exhaustive theoretical background into the optimization algorithms but rather to provide high level insight into the basic functioning of the algorithms.

The standard constrained NLP problem is formulated as:

$$\begin{aligned} & \min_{x \in \mathbb{R}^n} f(\mathbf{x}) \\ \text{subject to}: \quad & h_i(\mathbf{x}) = \mathbf{0}, \quad i = 1, \ldots, m \\ & g_j(\mathbf{x}) \leqslant \mathbf{0}, \quad j = 1, \ldots, p \end{aligned} \tag{2.4}$$

where $m < n$ and $f : \mathbb{R}^n \to \mathbb{R}$ is the objective function to be minimized, $h : \mathbb{R}^n \to \mathbb{R}^m$ are the equality constraints and $g : \mathbb{R}^n \to \mathbb{R}^p$ are the inequality constraints.

Classical optimization methods make use of the Lagrangian function and Lagrange multipliers in order to take the constraints into account. The Lagrangian $\mathcal{L} : \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^p \to \mathbb{R}$ for the constrained optimization in 2.4 is given by:

$$\mathcal{L}(x, \lambda, \mu) = f(\mathbf{x}) + \boldsymbol{\lambda}^\top \mathbf{h}(\mathbf{x}) + \boldsymbol{\mu}^\top \mathbf{g}(\mathbf{x}) \tag{2.5}$$

where the vectors $\boldsymbol{\lambda}$ and $\boldsymbol{\mu}$ are the associated *Lagrange multipliers.*

The definition of a local minimizer(minimum) is given in [30] as:
"*A point x^* is a local minimizer if there is a neighbourhood $\mathcal{N}$ of x^* such that $f(x^*) \leqslant f(x)$ for all $x \in \mathcal{N}$*"

while the definition (also in [30]) of a global minimum is given as:
"*A point x^* is a global minimizer if $f(x^*) \leqslant f(x)$ for all x, where x ranges over all $\mathbb{R}^n$ (or at least over the domain of interest)*"

One of the key or fundamental properties of a convex function is that if f is a convex function in a set S then **any** local minimum of S must be a global minimum as mentioned and proven in convex analysis and convex optimization resources such as ([29], [31] and [32]).

The Karush-Kuhn-Tucker (KKT) conditions, which are also known as first order necessary conditions for optimality, provide conditions for finding local optimizers (minima). The KKT optimality conditions are stated as follows: If x^* is a local minimum of the problem in 2.4 where f, g_j and h_i are continuously differentiable functions and the Linear Independence Constraint Qualifications (LICQ) [1] conditions are satisfied then there exists unique Lagrange multiplier vectors $\boldsymbol{\lambda^*} = (\lambda_1^*, \ldots, \lambda_m^*)$ and $\boldsymbol{\mu^*} = (\mu_1^*, \ldots, \mu_p^*)$ such that:

$$\begin{aligned}
\nabla_x \mathcal{L}(x^*, \lambda^*, \mu^*) &= 0, && && \text{(2.6a)}\\
g_j(x^*) &\leq 0, && i = 1, \ldots, p && \text{(2.6b)}\\
\mu_j^* &\geq 0, && j = 1, \ldots, p && \text{(2.6c)}\\
h_i(x^*) &= 0, && i = 1, \ldots, m && \text{(2.6d)}\\
\mu_j^* g_j(x^*) &= 0, && j = 1, \ldots, p && \text{(2.6e)}
\end{aligned}$$

Equation 2.6a is also referred to as *Lagrangian stationarity*, 2.6b and 2.6d as *Primal feasibility*, 2.6c as *Dual feasibility* and 2.6e as *Complementarity* conditions.

2.2.1 Sequential Quadratic Programming

Sequential Quadratic Programming (SQP) is a local optimization method which falls under the category of Nonlinear Programming (NLP) optimization strategies. It has been mentioned to be *"one of the most effective methods in nonlinearly constrained optimization"* in [30] and was found to be the *"most efficient"* [33] in the tests conducted by Schittkowski comparing several other strategies as presented in [33]. The description of the SQP algorithm that follows is based on the

[1] The LICQ conditions are said to hold if the gradients of the active constraints are linearly independent i.e for a point x, constraints $c_j(x)$ and active set $A(x)$; $\nabla c_i(x), i \in A(x)$ is *linearly independent.* [30]

texts [30] and [1] and aims to provide intuition into the operation and functioning of the algorithm. The details of the algorithm have not been included since it would not be possible to do justice to all the aspects of the SQP algorithm in this study. Texts such as [30] among several others provide details of the algorithm.

The SQP method that is to be described in this section is known as the *Active Set*[2] method [34]. This method is named as such since the approach in which the constraints are handled, is by means of a set (called a working set) which is updated at every iteration. The working set contains all of the equality constraints and a selection of inequality constraints at the current iteration; it is an approximation to the active set. All the constraints contained within the working set are treated as equality constraints while the constraints not contained by the working set are ignored (for the current iteration). The Linear Independence Constraint Qualifications (LICQ) are a requirement in the algorithm i.e. that the gradients of the constraints in the working set form a linearly independent set.

SQP solves optimization problems of the form 2.4 by forming a quadratic approximation of the objective function, solving this resulting Quadratic Programming (QP) subproblem and then using the result to progress to the next iteration.

In order to form the QP subproblem the Lagrangian Function is used. In this section the Lagrangian function defined in 2.5 is condensed by considering that the vector $\boldsymbol{\lambda}$ contains both the Lagrange multiplier vectors ($\boldsymbol{\lambda}$ and $\boldsymbol{\mu}$) associated with the equality and inequality constraints. Furthermore, for additional brevity, the inequality and equality constraints ($\mathbf{h}(\mathbf{x})$ and $\mathbf{g}(\mathbf{x})$ respectively) will be contained by $\mathbf{c}(\mathbf{x})$ where $c : \mathbb{R}^n \to \mathbb{R}^{m+p}$ (as done in [1]).

The updated Lagrangian function of 2.4 becomes:

$$\mathcal{L}(x, \lambda) = f(x) + \boldsymbol{\lambda}^\top \mathbf{c}(\mathbf{x})$$

A quadratic approximation of the Lagrangian function close to a point (x_k) takes the form of 2.7 after some manipulation and the use of identities (more detail to be found in [1]); where $x_k, f(x_k), d_k, \lambda_{\mathcal{W}k}$ are the current iterate, original objective function at the current iterate, search direction and the Lagrange multiplier iterate in the working set ($\mathcal{W}k$) .

$$f(x_k) + \nabla f(x_k)^\top d_k + \frac{1}{2} d_k^\top \nabla^2_{xx} L(x_k, \lambda_{\mathcal{W}k}) d_k \tag{2.7}$$

[2] An inequality constraint is considered to be active at x^* if $g_j(x^*) = 0$ and *inactive* if $g_j(x^*) < 0$.Equality constraints are always considered active and the *Active Set* is the set of indices of **all** active constraints. [34]

$\lambda_{\mathcal{W}k}$ is defined to be the vector of Lagrange multiplier iterates that are zero when the component is not part of the working set and equivalent to λ_k for the components which form part of the working set $\mathcal{W}k$. This means that

$$\begin{aligned} \lambda_{\mathcal{W}k} &= \lambda_k \qquad \forall i \in \mathcal{W}k \\ \lambda_{\mathcal{W}k} &= 0 \qquad \forall i \notin \mathcal{W}k \end{aligned}$$

The constraints from 2.4 are linearised close to iterate x_k to give:

$$c_i(x_k) + \nabla_x c_i x_k^\top d_k = 0 \qquad \forall i \in \mathcal{E} \tag{2.8a}$$

$$c_i(x_k) + \nabla_x c_i x_k^\top d_k \leqslant 0 \qquad \forall i \in \mathcal{I} \tag{2.8b}$$

where $\mathcal{E}$ and $\mathcal{I}$ are finite index sets of equality and inequality constraints [30]. Thus the resulting problem that is to be solved at each iteration takes the form of:

$$\begin{aligned} \min_{d_k} \; & f(x_k) + \nabla f(x_k)^\top d_k + \frac{1}{2} d_k^\top \nabla^2_{xx} L(x_k, \lambda_{\mathcal{W}k}) d_k \\ \text{subject to} : \; & c_i(x_k) + \nabla_x c_i x_k^\top d_k = 0 \qquad \forall i \in \mathcal{E} \\ & c_i(x_k) + \nabla_x c_i x_k^\top d_k \leqslant 0 \qquad \forall i \in \mathcal{I} \end{aligned} \tag{2.9}$$

The goal of solving the problem 2.9 for the optimal value of d_k and the corresponding Lagrange Multiplier ($\hat{\lambda}_{\mathcal{W}k}$) at every iteration is in order to progress and update the iterates x_k and λ_k by the following:

$$x_{k+1} = x_k + \alpha_k d_k \tag{2.10a}$$

$$\lambda_{k+1} = \hat{\lambda}_{\mathcal{W}k} \tag{2.10b}$$

where $\hat{\lambda}_{\mathcal{W}k}$ is the Lagrange Multiplier that corresponds to the solution of 2.9. α_k in 2.10a is the step size which needs to be obtained in such a way that constraints are not violated and there is sufficient decrease in the objective function. The step length is required to calculate the next iteration and in order to calculate it a function known as a *merit function* is used. The merit function provides a means of balancing between the violation and adherence to constraints and decrease in objective function. The *line search* methodology (which is discussed in this introduction to SQP) makes use of a merit function to manage the value of the step length α_k.

[1] and [30] both discuss the use of the merit function by employing an example merit function that uses the ℓ_1 norm (given in 2.11) and this will be the same merit function that is used in this

discussion . With the use of a merit function the inequality constraints are converted into equality constraints by means of slack variables(s) where the inequality constraints given by $c(x) \geqslant 0$ become $\bar{c}(x, s) = c(x) - s = 0$

$$M(x, \mu) = f(x) + \mu \|c(x)\|_1 \quad \text{where } \mu > 0 \tag{2.11}$$

The condition for acceptance of the step length (α_k) and hence the step ($\alpha_k d_k$) is given by the following

$$M(x_k + \alpha_k d_k, \mu) \leq M(x_k, \mu) + \eta \alpha_k D(M(x_k, \mu), d_k)$$

where D is the directional derivative and $\eta \in (0, 1)$.

Outline of Basic Active Set Sequential Quadratic Programming Algorithm (Adapted from [1])
Given initial values x_0, λ_0, initial working set $\mathcal{W}_0$ and $k = 0$.

1. Evaluate $f(x_k), \nabla f(x_k), \nabla^2_{xx} L(x_k, \lambda_k), c(x_k)$ and $\nabla c(x_k)$
2. [1] Obtain $d_k, \hat{\lambda}_{\mathcal{W}k}$ and $\mathcal{W}_{k+1}$
3. Update iterate x_{k+1} and λ_{k+1}
   i. Determine μ such that $D(M(x_k, \mu), d_k) < 0$
   ii. Obtain α_k such that $M(x_k + \alpha_k d_k, \mu) \leq M(x_k, \mu) + \eta \alpha_k D(M(x_k, \mu), d_k)$
   iii. Update $\lambda_{k+1} = \lambda_k + \alpha_k(\hat{\lambda}_{\mathcal{W}k} - \lambda_k)$
   iv. Update $x_{k+1} = x_k + \alpha_k d_k$
4. Update k $\rightarrow$ k+1

2.2.2 Particle Swarm Optimization

Particle Swarm Optimization (PSO) is a meta-heuristic optimization technique that falls under the category of Swarm Intelligence (SI). Swarm Intelligence is defined in [35] as *"the property of a system whereby the collective behaviours of (unsophisticated) agents interacting locally with their environments cause coherent functional global patterns to emerge".*

PSO is a population based algorithm that was developed in 1995 by Kennedy and Eberhart after researching the social system of birds. Thus, PSO was inspired by the flocking of birds and the

[1] Details of step 2 have not been included for brevity and can be found in Optimzation texts covering SQP (example [30]) or [1]

interaction between individuals within the flock.

At a high level, PSO operates on the notion of collaboration and interaction between the particles in the swarm (or individuals within the flock) such that the particles tend to gravitate or are attracted towards the success of neighbouring particles [26]. As particles, which represent candidate solutions, are *"flown"* through the hyper-dimensional search space, their movement is, to some extent, influenced by the experiences of neighbouring particles.

PSO has attracted the attention of many researchers since its beginnings, evolving along the way and has found applications across the spectrum from neural networks [36] to image processing [37] to electric power systems [38]. Some of the salient attractive features of PSO are that it is conceptually simple with relatively few parameters to change.

As mentioned previously, PSO has evolved over the time yet the basic fundamental principles of PSO still remain the same. Of the two original PSO algorithms (Local Best PSO and Global Best PSO) Global Best (gBest) PSO will be described in this section [26]. This is due to the two algorithms being very similar with the only major difference being the range of the collaborative neighbourhoods. The description of one algorithm should provide a good high level understanding of the overall PSO algorithm.

The neighbourhood for the Global Best PSO algorithm is the entire swarm which means that information is shared among all particles within the swarm. Figure 2.6 shows the star topology which is generally used for gBest PSO. In PSO there are two primary components, the *cognitive component* and the *social component*, that contribute to the optimization process. Particles are moved within the search space by adding a driving velocity vector that takes both the social and cognitive components into account. [26] describes the cognitive component as *"experiential knowledge"*, which is proportional to the particle's distance from its personal best position, and the social component as *"socially exchanged information"*.

It should be noted that PSO, due to its meta-heuristic nature, has the ability to escape local minima although it is not completely immune to, and may get trapped in, local minima. Many developments in PSO have addressed this issue and provide strategies to enhance the PSO algorithm such that it is less susceptible to local minima such as PSO with *escape velocity* in [39] and the use of *chaotic sequences* in [40], amongst many others.

Equation 2.12 describes how the velocity of a particle is updated which includes the social, cognitive and inertial components. The inertial component ($\omega_1 v_{ij}(t)$) provides a certain amount of momentum to the particles. $\alpha_1 r_1[pBest_{ij} - x_{ij}]$ is the *cognitive component*, with α_1 being a positive acceleration constant, $r_1 \in [0, 1]$ a uniformly distributed random number, $pBest_{ij}$ is the personal

best position of the particle i in dimension j and x_{ij} is the position of particle i in dimension j. Similarly, $\alpha_2 r_2[gBest_j - x_{ij}]$ is the *social component* with α_2 being a positive acceleration constant, $r_2 \in [0, 1]$ a uniformly distributed random number, $gBest_j$ is the global best position in dimension j and x_{ij} is the position of particle i in dimension j. The personal best (pBest) is the best position that a particle has encountered since the initial time and the global best (gBest) is the best particle that was encountered by the entire swarm.

$$v_{ij}(t+1) = \omega_1 v_{ij}(t) + \alpha_1 r_1[pBest_{ij} - x_{ij}] + \alpha_2 r_2[gBest_j - x_{ij}] \tag{2.12}$$

Following the calculation of the velocity in 2.12 the next step is to update the position using equation 2.13 [26].

$$\mathbf{x}_i(t+1) = \mathbf{x}_i(t) + \mathbf{v}_i(t+1). \tag{2.13}$$

[26] provides a formulation for calculating the personal best (pBest) at the time step t+1 as:

$$\mathbf{pBest}_i(t+1) = \begin{cases} \mathbf{pBest}_i(t) & \text{if} \quad f(\mathbf{x}(t+1)) \geqslant f(\mathbf{pBest}_i(t)) \\ \mathbf{x}_i(t+1) & \text{if} \quad f(\mathbf{x}(t+1)) < f(\mathbf{pBest}_i(t)) \end{cases} \tag{2.14}$$

where $f : \mathbb{R}^d \to \mathbb{R}$ is the objective or fitness function.

One method that the global best position can be obtained is by testing for the best of all personal best positions as given in [26]:

$$gBest(t) \in \{\mathbf{pBest}_0(t), \ldots, \mathbf{pBest}_n(t)\} \mid f(\mathbf{pBest}(t)) = \min\{f(\mathbf{pBest}_0(t)), \ldots, f(\mathbf{pBest}_n(t))\} \tag{2.15}$$

where n is the number of particles in the swarm.

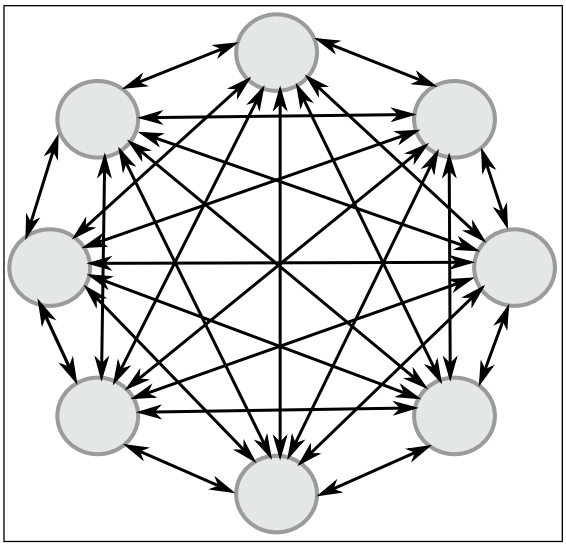


Figure 2.6: PSO Star Topology

Algorithm 2.1 Algorithm for gBest PSO [26]

```
Create d-dimensional swarm;
Initialize swarm;
repeat
    for each particle i = 1, ..., n do
        if f(x_i) < f(pBest) then pBest = x_i;
        end if
        if f(pBest_i) < f(gBest) then
     gBest=pBest_i
        end if
    end for
    for each particle i = 1, ..., n do
        update velocity;
        update position;
    end for
until stopping condition is true
```

2.3 Model Predictive Control

Linear Model Predictive Control (MPC) has a lengthy history dating to the late 1970's when Richalet et al. published papers on Model Predictive Heuristic Control (MPHC) and Cutler and Ramaker on Dynamic Matrix Control (DMC) which later progressed to being an extremely successful commercial product[41][5]. However, as mentioned in [41], the true beginnings of MPC may have been earlier with industrial implementations of predictive control. One of the key concepts of MPC, the receding horizon, was described in the context of optimal control, by Propoi over a decade earlier (1963) [41][5]. A second key aspect of MPC is the use of a model of the system and

work by Garcia and Morari in 1982 [42] encapsulated this in the form of Internal Model Control (IMC) [43].

Following these early beginnings, MPC has received a significant amount of attention from industry and academics which led to it becoming increasingly popular. Thus, MPC has enjoyed great success in being employed by industry with the majority of applications being in the refining industry followed by the petrochemical and process industries as provided by Qin and Badgwell in [6]. Although MPC was largely employed by the refining and petrochemical industries, it has been increasingly applied in the automotive, aerospace and pulp industries. Consequently, MPC has seen several vendors include variants of the MPC methodology as part of their product offering. Table 2.1 below lists some of these MPC products.

Table 2.1: Table of MPC Vendors and Products [6] [5]

Company	**Product(s)**
AspenTech	Dynamic Matrix Control (DMC)
Adersa	Identification and Command (IDCOM)
Honeywell Hi-Spec	Robust Model Predictive Control Technology
Treiber Controls	Optimum Predictive Control (OPC)
ABB	3dMPC
Invensys	Connoisseur
Shell Global Solutions	SMOC-II

Reasons that have been cited for the success of MPC include [5]:

- Intuitive and easily understood concepts
- Ability to handle multivariable processes
- Applicability to a wide spectrum of processes
- Ability to take constraints into account
- Relatively simple implementation of control law

A statement by Mayne in [44] reads *"Model predictive control is the only advanced control technology that has made a substantial impact on industrial control problems: its success is largely due to its almost unique ability to handle hard constraints."* Although the statement makes a bold claim, it is unanimous (in MPC literature) that MPC has played a significant role by its successful application in industry.

MPC refers to a family or group of control strategies which include Model Algorithmic Control (MAC), Generalised Predictive Control (GPC), Predictive Functional Control (PFC) and Quadratic

Dynamic Matrix Control (QDMC), amongst others. Thus, MPC represents a control methodology with common key elements rather than a specific control technique.

The key elements within the MPC methodologies are [5]:

1. The use of a model to predict future outputs
2. Obtaining a control input that optimizes an objective or cost function
3. Use of a receding horizon strategy
4. Application of only the first control input at each step

The first key element of MPC is the model of the plant or system in order to do prediction and as such, a model that encapsulates the plant dynamics is used. The different variations of MPC use differing model types. Table 2.2 summarizes some of the MPC strategies and the models that they use.

Table 2.2: Examples of MPC strategies and Their Models [5]

MPC Strategy	**Model**
Model Algorithmic Control (MAC)	Impulse Response
Dynamic Matrix Control (DMC)	Step Response
Generalised Predictive Control (GPC)	Transfer Function
Predictive Functional Control (PFC)	State Space

The second constituent of MPC is an objective function that is minimized or optimized to generate the control input. The objective function also varies with the different MPC schemes but the general idea is to minimize the deviation or error from a reference or set-point.

The receding horizon strategy that MPC schemes make use of can be intuitively understood (as explained in [41]) by thinking of the Earth's horizon. The concept is that by moving towards the horizon it seems to recede or move away and thus always remains a fixed distance away. This concept is applied in MPC in that there is a "prediction horizon" and "control horizon" which are fixed and with every time step these horizons *recede* as well.

The fourth fundamental principle of MPC is that once the control input or control law has been calculated (by minimizing the objective function), only the first input in the sequence is applied which implicitly tells us that at every time step a control input needs to be calculated.

Figure 2.7 above provides an overview of MPC where t is the current time with the past to the left and future to the right. Thus to the left is the closed-loop and to the right the open-loop prediction

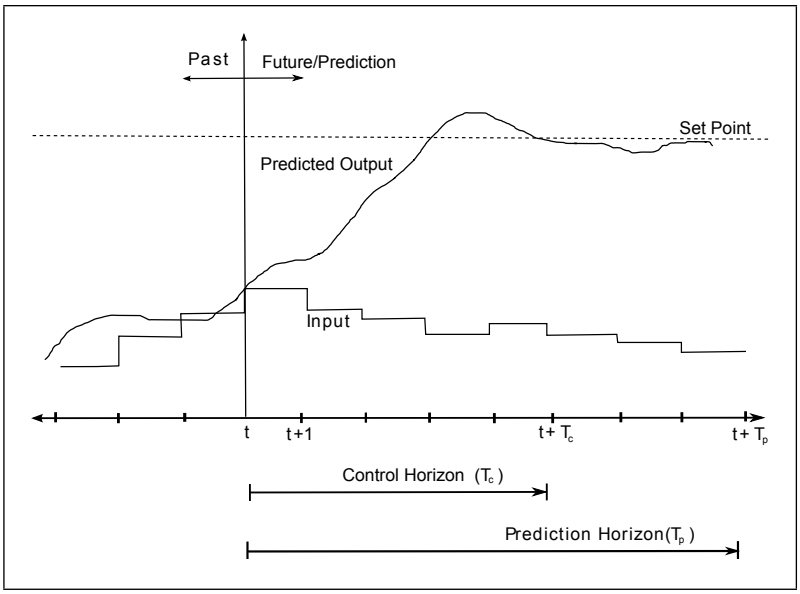


Figure 2.7: Overview of MPC Algorithm (Adapted from [3] and [4])

is depicted.

The first step in the MPC strategy is to obtain a prediction (using a model) for a specific time or horizon which is known as the prediction horizon, T_p. The prediction of the future output trajectory is dependent on the current state values and future input values over the prediction horizon.

Following the prediction, the control inputs for a chosen control horizon, T_c, are calculated. The control and prediction horizons need not be different and can be equal to each other. Calculation of the control inputs entail optimizing an objective function or, in other words, finding the optimal value of the control inputs that will minimize the error between the predicted output and the set-point trajectory. Generally, the objective function is a quadratic function.

Once the input trajectory has been obtained, only the first component or element is applied to the system and the rest are discarded. After applying the input, at the next time instant, the process repeats where another prediction over the prediction horizon is generated and the input values calculated for the control horizon and then the first element of the input is applied to the system.

Figure 2.8, succinctly encapsulates the major components of MPC, being the system model and the optimization algorithm. It also demonstrates that the past and current values are used with the model in order to produce the prediction. The optimization algorithm solves the optimization problem, using the cost function and constraints, to produce the required input sequence values. It is noticed in Figure 2.8 that the optimization algorithm block is highlighted in yellow to highlight the domain of focus in this study. To recap what was mentioned in the introduction in a nutshell,

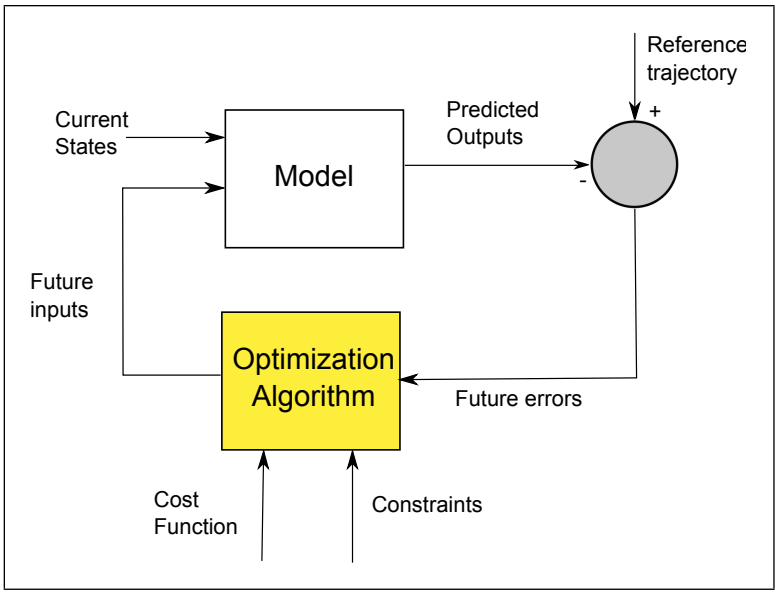


Figure 2.8: Overview of MPC Structure (Adapted from [5])

the aim of this study is to graphically represent the optimization problem to be solved by the optimization algorithm thereby providing insight.

2.3.1 Simple Formulation of MPC

A simple formulation of MPC is now presented for a discrete, linear time-invariant (LTI) system [4]. The state space model is given:

$$x(k+1) = Ax(k) + Bu(k) \qquad x(k) \in \mathbb{X}, u(k) \in \mathbb{U} \tag{2.16a}$$

$$y(k) = Cx(k). \tag{2.16b}$$

Using this model the output prediction y is obtained. A generic cost or objective function,J, is given by equation 2.17:

$$J_N(x(k), u(k)) = \sum_{i=0}^{N-1} \ell(x(k+i), u(k+i)), \tag{2.17}$$

where N is the prediction horizon. ℓ could take the form of $\ell = x^\intercal Qx + u^\intercal Pu$ where P and Q are chosen symmetric positive definite matrices. Other suitable definitions of ℓ may also be used.

The cost function is then minimised or optimised in order to obtain the optimal control input sequence ($\mathbf{u}^*(k)$) as shown in equation 2.18

$$
\begin{aligned}
u^*(k) &= \min_{u \in U} \quad J_N(x(k), u(k)) \\
\textit{subject to} \quad x(k+i+1) &= Ax(k+i) + Bu(k+i) \\
x(k+i) \in \mathbb{X} \quad &, \quad u(k+i) \in \mathbb{U}, \qquad i = 0, \ldots, N-1,
\end{aligned} \tag{2.18}
$$

where $\mathbb{X}$ and $\mathbb{U}$ are vector spaces. Once the optimal input sequence ($\mathbf{u}^*(k)$) has been calculated then the first element of ($\mathbf{u}^*(k)$) is applied to the system.

2.4 Nonlinear Model Predictive Control

As previously alluded to, Linear Model Predictive control is a mature field and has been successfully employed in industry with 2200 industrial applications by 1997 [45]. However, situations arise whereby linear models are not sufficiently accurate or where models are inherently nonlinear. Additionally, as industries become increasingly competitive, productivity demands increase alongside more stringent quality requirements, hence models that more accurately describe the systems are required [46]. This led to nonlinear models being investigated in the context of MPC and led to what is now known as Nonlinear Model Predictive Control (NMPC).

NMPC, the extension of MPC into the nonlinear domain, still comprises the key elements making up MPC as mentioned in section 2.3 i.e. explicit use of model, optimising of an objective function, receding horizon strategy and the application of the first control input. However, the distinguishing factor is that NMPC uses system models that are nonlinear. Although not to the same extent as MPC, in recent years NMPC has also attracted the attention of industry and researchers and as such several vendors have NMPC products available some of which are presented in Table 2.3.

Table 2.3: Table of NMPC Vendors and Products [7]

Company	**Product(s)**
AspenTech	Aspen Target
Adersa	Predictive Functional Control (PFC)
Continental Controls	Multivariable Control
DOT Products	NOVA Nonlinear Controller (NOVA-NLC)
Pavilion Technologies	Process Perfecter

The introduction of a nonlinear system model in NMPC brings several challenges along with it, many of which are still under investigation. [47] makes mention that NMPC faces challenges in all facets some of which are modelling, identification, stability and real-time implementation. It is also mentioned by Mayne that the *"online solution of non-convex optimal control problems"* needs to be addressed and *"failure to respond to this challenge will severely inhibit the use of*

NMPC" in [15]. This challenge has been taken on by researchers by the multitude of proposed approaches in dealing with the problem such as [48], [49], [50], to cite a few. Section 4.1 discusses the formulation of the NMPC, while section 4.2 discusses the NMPC algorithm and section 4.3 discusses the implementation of NMPC employed in this study.

Chapter 3

Literature Survey

This section presents some of the research that has been carried out, in and around the problem domain of this study. NMPC is an area of considerable research activity and this is by no means meant to be an exhaustive survey. Rather, the goal of this section is to illustrate some of the approaches taken by researchers relating to the topic of this study. Firstly, various approaches to the NMPC problem and the constituent optimization problem are presented. This is followed by the NMPC implementation of the two optimization algorithms employed in this study, i.e. SQP and PSO, and its applications. Although literature directly related to the focus of this study (graphical insight into optimization in NMPC) has not been found, approaches to the quantification of nonlinearities, which are somewhat related to the topic, are discussed. Additionally, visualisation relating to optimization is briefly covered.

3.1 NMPC Formulations and Optimization

The work in [51] presents an approach that uses Volterra series models (second order) in NMPC and also uses a cost function that is a convex approximation of the original (possibly non-convex) cost function. This was done by approximating the original convex function by its convex hull, also known as a convex envelope. In this implementation SQP was also used as the optimization algorithm although it was highlighted that any standard convex optimization algorithm would be suitable. Results of the algorithm being applied to a CSTR are presented and showed the algorithm's ability to effectively track the set-point while enduring disturbances.

The study in [19] presents a formulation of an NMPC algorithm that also uses a convexification strategy. This strategy makes use of variable transformations to transform the non-convex Nonlinear Programming (NLP) problem into a convex NLP problem which is then further converted into a Linear Programming (LP) problem by means of linearisation. The resulting LP is then solved by making use of a modified branch-and-reduce optimization technique (based on branch-and-bound) which is used in conjunction with interval analysis [1] in order to speed up the convergence. This

[1] Interval analysis is based on the concept of using intervals of real numbers rather than real numbers. It makes

method is demonstrated on a CSTR. [52] presents a very similar approach to [19].

The investigation in [13] extends the approach undertaken in [19] for application to systems with *hybrid dynamics*, which are described as systems with *"discrete switching between multiple continuous regimes of operation"*. These problems take the form of Mixed Integer Nonlinear Programming (MINLP) problems which are approximated or *relaxed* at each iteration to Mixed Integer Linear Programming (MILP) problems to obtain the lower bounds and the *binary realization*. The problem is then converted to an NLP by using the *binary realization* in the original MINLP. As was done in [19] the branch-and-reduce technique is used to obtain the upper bound. Another binary realization is obtained by solving the MILP again after excluding the previous binary realization from the next major iteration. The rest of the algorithm follows a very similar pattern to that of [19] (and [52]).

[53] describes an approach that uses neural network optimization in NMPC for the control of autonomous autorotation of small unmanned helicopters. Autorotation, as explained in [53], is a phenomenon in helicopters that allows them to make use of the rotor energy in the event of an engine failure (or similar scenario) to control the descent of the helicopter. This is a system with fast dynamics and this implementation of NMPC also bears testimony to the increasing application of NMPC, especially to systems with fast dynamics. The neural networks employed in this formulation have also been shown (in other works such as [54]) to have the ability to solve convex constrained nonlinear optimization as well as a class of non-convex constrained nonlinear optimization problems [53]. The neural networks solve the constrained nonlinear optimization problem by modelling an ordinary differential equation (ODE) whose equilibrium point corresponds to the solution of the optimization problem. The paper also compares the computational complexity of the Recurrent Neural Network (RNN) method to that of gradient descent based methods such as SQP and was shown to be much lower by the avoidance of having to calculate the Hessian.

An interesting statement in the paper [53] is that the neural network has been proven to be applied to a certain class of non-convex problem and mention has also been made of the possibility of entering into non-convex regions. The Optimization Roadmap methodology presented in this study providing insight into the optimization problem may prove to be useful in this case in order to provide the user with a better understanding of the problem.

In [55] an aircraft control scheme that is based on "*Fast*" NMPC is presented. The aircraft control problem is set in an NMPC form such that a reference trajectory is to be tracked while adhering to constraints such as a flight envelope and managing hardware faults or failure conditions. The NMPC formulation that is employed (similar to the approach used in this study) is based on the repeated solution of an Optimal Control Problem. The problem is then discretized using a Si-

use of interval arithmetic and analytical techniques and can be used for global optimization.

multaneous Discretization method called *Multiple Shooting* which transforms the problem into a Nonlinear Programming (NLP) problem. Methods such as Initial Value Embedding, where the initial values are based on the solution of the previous time instant, and *Real Time Iteration* (RTI) are used to speed up the solution. The solution to the NLP was obtained using the Sequential Quadratic Programming (SQP) algorithm. A very similar application of this approach to a gasoline engine is described in [56].

In [57], the non-convex NMPC problem is cast as a multi-parametric Nonlinear Programming problem where initialization or setting of initial conditions is important to find a "*close-to-global*" solution. The approach taken in obtaining good initial conditions is evaluating and comparing the local minima by making use of several varying initial guesses. The advantage of converting the NMPC problem to a parametric programming problem is that function evaluations are required for parametric programming which are far *"less computationally expensive than real-time optimization"* [57]. The study proposes an approach to obtaining *explicit Piecewise Linear solutions to nonconvex NMPC problems* by constructing an approximation of the optimal solution that is piecewise linear. During exploration and evaluation, the solution space (*hyper-rectangle*) is further divided into more hyper-rectangles which are then explored and evaluated until a feasible solution that is within the required tolerance value is found.

The approach presented in [20] is an offline methodology which is applied to systems described by parameter-varying polynomials (PPV) where the optimization is carried out offline. This reduces the dependency of the NMPC algorithm on the time taken for the optimization to be complete. The offline computation includes (i) the optimization, which involves using semi-definite programming to solve a convex optimization problem (the system is made to be bounded by a convex polytope) that is constrained by Sum-of-Squares (SOS) constraints, and (ii) nested *invariant sets*. These values are then stored in a lookup table. The lookup table is searched for the invariant set that contains the states at the current time instant and the associated feedback control law (computed from the optimization mentioned previously) that will produce the best results. The methodology was applied to a numerical example showing an improvement in convergence time from using online methods. [20] also makes mention of similar methodologies such as those described in [58] (NMPC using SOS constraints) and [59] (offline NMPC).

A suboptimal NMPC approach that employs Genetic Algorithms (GA) for the optimization is proposed in [60]. The essence of this approach is that instead of attempting to locate the global or even local optima via the optimization, it is sufficient if the control input that is to be applied reduces the cost value while adhering to the constraints and stability requirements. At the current time instant the cost for the next time instant is calculated and tested for compliance to the strategy i.e. reducing the cost. If the control input value does not comply, then the best value that decreases the cost is chosen. The Initial Value Embedding strategy is also used in the approach where the

previous results are used as the initial values for the genetic algorithm. The algorithm was simulated and tested experimentally showing comparable results (although the suboptimal control has a slightly longer transient phase) to conventional NMPC with GA. The advantage is that it has a significantly lower computational demand.

The advanced-step NMPC (asNMPC) is another approach to *Fast* NMPC and has been implemented and tested in several applications ([17], [50]). In brief, the gist of the asNMPC algorithm is that values for the states and control inputs are used to generate the values for the next time step and the optimal control problem solved for the control input, in advance, in the background. The fast online computation is handled by making use of Nonlinear Programming sensitivity analysis to obtain approximate solutions to the NLP and hence yield the approximate control input.

The above discussion of some of the approaches used in NMPC illustrates that many approaches exist in solving the NMPC problem with the aim of improving the computation time and/or the optimality of the solution of the optimization algorithm. The goal of this study is not to investigate a new method of solving the NMPC problem but rather to gain insight into the optimization problem which informs the user of the type of optimization problem that is to be solved (convex or non-convex) and regions of non-convexity, if applicable.

3.2 NMPC with the PSO Algorithm

NMPC adopting PSO as the optimization algorithm has been researched and implemented in several applications using both a standard PSO as well as a modified PSO algorithm. [22], [61] present a PSO based NMPC algorithm and its application to the well known inverted pendulum system with a comparison of the performance to an approach using numerical linearisation and convex optimization. In [62] the application of a PSO based NMPC algorithm to district heating networks is presented. NMPC with a modified PSO algorithm (integrated with a chaotic local search and roulette wheel mechanism) was applied to a pH Neutralization reaction in [63] in order to improve the performance of the standard PSO algorithm. The results proved the method to be feasible for further research and experimentation. [27] presented an NMPC algorithm based on PSO optimization and applied it to a *Greenhouse Climate* for energy efficiency. Furthermore, another NMPC algorithm with a modified PSO algorithm applied to another well known problem, the ball and plate, is presented in [64]. In this case, the modified PSO algorithm used was the Gaussian PSO (GPSO) while in [65] the modified PSO algorithm comprises a PSO and simulated annealing *hybrid* optimization algorithm.

3.3 NMPC with the SQP Optimization Algorithm

Some NMPC applications using SQP as the optimization algorithm have already been mentioned previously, however the Sequential Quadratic Programming optimization algorithm was found (in literature) to be an optimization algorithm that has found many applications in NMPC. [23] provides a comprehensive presentation of the solution of NMPC by considering the application of SQP as well as a second Newton-type optimization (Interior Point method). It also discusses the real-time implementation of the algorithms and provides a short survey of approaches suggested in literature. [66], [67] also present the implementation of SQP in NMPC and additionally considers the different approaches to the NMPC algorithm, for example the various discretization schemes and their effects, when applied to a CSTR case study. Similar to [23], [68] undertakes a comparison between Active Set SQP methods and Interior Point methods which each have their own advantages and disadvantages as discussed in [68], for example Interior Point methods display effectiveness in large size problems. A variation of the standard SQP algorithm called the *Feasibility-Perturbed SQP* is used in [69] in implementing NMPC and again applied to several systems including CSTR and the inverted pendulum. The work in [70] applied SQP based NMPC to a High-Density Poly-Ethylene (HDPE) plant and also presented a modification of the SQP algorithm to take advantage of the structure of the Hessian. [71] presents NMPC based on a modified SQP optimization algorithm (called the filter-trust-region method). The modifications are to improve efficiency and feasibility by taking advantage of the structure of the Hessian matrix.

Although this work will focus on the SQP and PSO, various other optimization methods have been implemented in NMPC, such as [72] and [24] where Genetic Algorithms were used. [21] made use of Interval Analysis while [73] made use of the Nested Partitions Method of optimization for the optimization within the NMPC.

3.4 Insight

3.4.1 Nonlinearity Quantification and Measure

Due to the nature of systems being nonlinear, work has been done in the testing and quantification of the nonlinearity of a system. A common approach to addressing the quantification of nonlinearities is to measure the nonlinear system against a linear version of itself and hence quantify the "degree of nonlinearity" of the system. [74] defines the nonlinearity measures of a dynamic system as "*the normalized largest difference between the nonlinear process and a linear time-invariant system*".

Alternatively, as noted in [75], the nonlinearity is measured against a straight line or the deviation of the system from a straight line. The work in [75] was aimed primarily at quantifying the nonlinearity in isolation, whereas [76] and [77] take the concept of quantifying nonlinearities

a step further by also taking the control action into account. [76] and [78] introduce the idea of an "*Optimal Control Structure (OCS)*" which, as the name suggests, is set in an optimal control context which is closely linked to MPC. The *Optimal Control Structure* takes the dynamics as well as performance objectives into account by making use of Lagrange multipliers and Lagrangian optimization while an open-loop nonlinearity test is done by using coherence estimation. This OCS method was applied to a practical chemical reactor problem in [77].

Building upon the work in [76], [78] and [77], [79] introduced the *"closed-loop Optimal Control Law (OCL) nonlinearity measure"* that uses closed-loop trajectories and the nonlinearity measure defined in [74], outlined in the first paragraph. This work improved upon previous research in terms of accuracy, computational performance and broader applicability. Many other approaches to obtaining a nonlinearity measure such as ([80],[81],[82],[83]) have been researched and documented.

The above mentioned methods are able to provide a quantitative measure of the nonlinearity of a system, however they do not provide a graphical view into the effects of the system nonlinearities on the optimization problem which affects the performance of the optimization algorithm and hence the control of the system (in Optimal Control and MPC).

3.4.2 Optimization Visualisation

Visualising or graphical views provide people with a platform from which to glean insight or information in a logical and clear layout. [84] presented a software package that provides the capability of visually analysing and comparing optimization methods in terms of, amongst others, rate of convergence and radius of convergence. A different outlook is presented in [85] where the visualization of the performance optimization algorithm is presented in real-time. A similar, more recent and specific application of this is found in [86] where the optimization or search process in Particle Swarm Optimization (PSO) is visually presented by displaying the movement of the particles within the search space. These tools are useful in obtaining a better understanding of the operation of optimization algorithms but are not suited to visually displaying the effects of nonlinearities on the performance of the optimization.

From the above discussion it can be deduced that not many tools are available for control engineers to gain insight into the optimization within NMPC. Such a tool, if made available, may provide engineers with confidence in their decisions and additionally enable them to base their decisions on a clear graphical representation of the behaviour of the system and the optimization (in terms of convexity etc.).

Chapter 4

NMPC Implementation

In this chapter the implementation of the NMPC algorithm is discussed. This includes some of the algorithm choices that were employed in [1] as this algorithm was used in this study. Firstly, a formulation of NMPC is presented which is followed by a discussion of the algorithm. The algorithm discussion presents the steps in the NMPC algorithm and some approaches to executing them, such as the discretization methods. The algorithm discussion also includes a section on the implementation of the optimization algorithms used in this study, which was done in MATLAB. A short test example showing the global vs. local optimization properties of the algorithms is shown. This is followed by a high level discussion of the MATLAB implementation of NMPC as carried out in [1] which was adopted in this study. Flow charts of the MATLAB implementation are available in Appendix A.

4.1 Formulation of NMPC

To outline the NMPC procedure, the first step is the use of a nonlinear model of the system dynamics to obtain a prediction of the system trajectory over a time horizon. This prediction is then used to calculate the required control input by optimizing an objective (cost) function over a control horizon. Once the control input sequence has been obtained, the first control input in the sequence is applied at the next time instance and the process is repeated.

This section presents the formulation of discrete time NMPC, primarily based on the algorithm presented in [1], beginning with the first step in the NMPC algorithm which is prediction using a system model. The discrete time nonlinear system (model):

$$
\begin{aligned}
x(k+1) = f(x(k), u(k)), \\
\text{where } x \ &\in \mathbb{X}, \\
u \ &\in \mathbb{U}, \\
k &\in \mathbb{N},
\end{aligned}
\tag{4.1}
$$

is used to calculate the trajectory of the system states ($\mathbf{x}$), given initial states $\mathbf{x}(0) = \mathbf{x}_0$ and a control sequence ($\mathbf{u}$). $\mathbb{U} \subset \mathbb{R}$, the set of all control inputs, and $\mathbb{X} \subset \mathbb{R}^n$, the set of all the states, will be considered as metric spaces which cater for the calculation of metrics or distances between elements.

In order to proceed to the next step of the NMPC algorithm, which is optimization, an objective or cost function is required. The objective function is the function that will be optimized subject to constraints which include the system model, the state constraints and input constraints. In general, the objective function provides a performance measure of how near or far the current states are from the desired reference state or trajectory. The objective function is, at times, used to penalise the control input (u) since it may be undesirable to have too large an input and is also often related to economic objectives ([7]). These are merely some of the more common examples of objective or cost functions but in general the cost function may be defined as warranted by the application.

Equation 4.2 presents a cost function based on the Euclidean distance or 2-norm which determines the distance between the state (x) and the reference trajectory, $\tilde{x}(k)$, as well as the input (u) and (if known) the reference control sequence, $\tilde{u}(k)$, with the parameter $\beta \geq 0$.

$$
\ell(x, u) = \|x(k) - \tilde{x}(k)\|^2 + \beta \|u(k) - \tilde{u}(k)\|^2 \tag{4.2}
$$

To continue with the second step of the NMPC algorithm, given a prediction horizon length ($T_p \geqslant 2$), an open-loop optimal control problem is formulated which is then solved to obtain the control input sequence.

$$
\begin{aligned}
&\min_{u} \quad J(x(k), u(k)) \\
&subject\,to: x(k+1) = f(x(k), u(k)), \\
&\text{with Initial Condition} \qquad x(0) = x_0, \\
&\qquad\qquad\qquad\qquad u \in \mathbb{U}, \\
&\qquad\qquad\qquad\qquad x \in \mathbb{X}, \\
&where \qquad J(x(k), u(k)) = \sum_{k=0}^{T_p - 1} \ell(x(k), u(k))
\end{aligned}
\tag{4.3}
$$

The closed-loop system is defined as $x_c = f(x, u_c(,x))$ where $u_c(x)$[1] is the control input feedback law which is the solution of the optimal control problem in 4.3.

Once the control input solution has been obtained the final step is to apply the first control input in the control input sequence at the next period.

4.2 Algorithm Discussion

The NMPC algorithm from [1], adopted in this work, is outlined below:

Step 1: Obtain measurement of the state $x(n)$ of the system.
Step 2: Set the initial conditions to the initial measurement and solve the Optimal Control Problem for the control input.

$$
\begin{aligned}
&\text{minimize} \quad J_{T_p}(x(k), u(k) : = \sum_{k=0}^{T_p - 1} \ell(x_u(k), u(k)) + F(x_u(T_p)) \\
&\text{with respect to} \quad u \in \mathbb{U}_{\mathbb{X}_0} \\
&\text{subject to} \quad x_u(0) = x_0, \\
&x_u(k+1) = f(x_u(k), u(k))
\end{aligned}
$$

Step 3: Set the first element of the control input sequence as the control input to be applied.

4.2.1 Discretization

Step 2 in the NMPC algorithm shows that an *Optimal Control Problem* needs to be solved in order to obtain the control input sequence. However, there are a few approaches to solving this problem namely [87]:

[1] Note the assumption that the solution to the optimization problem exists

1. Dynamic Programming and Hamilton-Jacobi-Bellman (HJB) partial differential equations.
2. Calculus of Variations and *Indirect* Methods
3. *Direct* methods which are based on the finite dimensional parameterisation of the controls

The *Direct* method in point 3 above is a solution to the *Optimal Control Problem* whereby it is transformed into a Nonlinear Programming optimization problem. This process is also known as *discretization* [1]. The basic idea behind the direct methods is that of first discretizing followed by the optimisation. The approach used in this study is one of the *Direct* methods called the *Sequential* method, *Recursive Discretization*[1] or the *Direct Single Shooting* method which is be discussed Section 4.2.1.2. A second *Direct method* called the simultaneous method is also briefly discussed.

This approach has several advantages since the problem is transformed into a finite dimensional NLP problem and then solved. Thus, the *Direct* methods allow for "state-of-the-art" NLP solution methods to be used. Further advantages of the two *direct* methods are discussed in Sections 4.2.1.1 and 4.2.1.2. The standard form of an optimization problem is presented below for ease of reference.

$$\begin{aligned} &\text{minimize} \quad F(z) \\ &\text{with respect to} \quad z \in \mathbb{R} \\ &\text{subject to} \quad G(z) = 0, H(z) \leq 0 \end{aligned}$$

4.2.1.1 Full Discretization[1] (Simultaneous Method)

The *Direct Simultaneous* or *Full* discretization method transcribes the original infinite dimensional problem into a finite dimensional NLP problem by discretizing the controls and states. The dynamics (for example discretized differential equations) are taken into account by being considered as constraints.

The equality constraints from the dynamics are given by:

$$x_u(k+1) - f(x_u(k), u(k)) = 0 \quad for \quad k \in \{0, \ldots, N-1\} \tag{4.4}$$

$$x_u(0) - x_0 = 0 \tag{4.5}$$

The optimization variable is given by

$$z := (x_u(0)^\top, \ldots, x_u(N)^\top, u(0)^\top, \ldots, u(N-1)^\top)^\top \tag{4.6}$$

and the cost function F is given by

$$F(z) := \sum_{k=0}^{N-1} \ell(x_u(k), u(k)) + F(x_u(N)) \tag{4.7}$$

The simplicity of the full discretization method makes it an attractive approach, however the side-effect is that the it suffers from the "curse of dimensionality" [1] since the optimization variable z takes on high dimensions and the number of constraints (Equality and Inequality) are escalated [1]. The *Simultaneous* method also has the advantage that the numerical integration of the dynamics or model equations are avoided [88] and it is able to handle unstable problems effectively [89]. The resulting NLP from the *Simultaneous* method is large and sparse [1].

4.2.1.2 Recursive Discretization[1] (Sequential Method)

The *Recursive discretization* approach or *sequential method* is the chosen discretization method for the NMPC implementation in [1] due to the dimensionality issue in full-discretization. The recursive discretization method splits the optimal control problem so that the dynamics are solved by one method and the optimization problem by another method and in every iteration the model equations or dynamics are solved. The sequential approach transcribes the problem into a finite dimensional NLP (as with the simultaneous method) but in the sequential method, the optimization variable only comprises the control variables.

In the recursive discretization method the optimization variable is:

$$z := (u(0)^\top, \ldots, u(N-1)^\top)^\top \tag{4.8}$$

where u represents a vector which could contain multiple inputs. Some advantages of the *Sequential Approach* are firstly that it results in a small NLP and the dynamics or model of the system are **always** adhered to. Two of the disadvantages of using the recursive discretization method are that firstly the solution of the numerical integration may become computationally expensive in large systems. The second disadvantage is that the sequential method may have difficulty in dealing with open-loop unstable systems [89]. In the case of an unstable system, the *simultaneous* method could be adopted or a "hybrid" method called the *multiple shooting* method could be used.

4.2.2 Optimization

The key operation in Step 2 of the NMPC algorithm in Section 4.2 is that of optimization where the goal is to obtain the control input sequence (u) that minimizes the cost or objective function. As mentioned previously two optimization methods (PSO and SQP) were employed in order to perform this vital step. Sections 4.2.2.1 and 4.2.2.2 provide an outline of the implementation of these algorithms in the NMPC algorithm as well as some performance characteristics relating to local and global optima, by means of an example.

4.2.2.1 SQP (fmincon[2])

The MATLAB function *fmincon*, which is part of the Optimization Toolbox, provides an implementation for constrained nonlinear optimization which includes the Sequential Quadratic Programming (SQP) algorithm, which was introduced in Section 2.2.1. The MATLAB *fmincon* implementation of SQP was used in the NMPC algorithm as one of the optimization algorithms. As mentioned in the MATLAB documentation, *fmincon* is a gradient based optimization implementation and thus suffers from being susceptible to local minima.

This characteristic will be verified by using a simple optimization problem containing global and local minima. The objective or cost function of the problem (shown in Figure 4.1) is:

$$f(x) = x^2 sin(x). \tag{4.9}$$

From Figure 4.1 it is evident that there are several minima. The goal of this example is to verify that SQP converges to a local minimum. A low dimensional example is used for visual confirmation of the performance of the SQP algorithm.

It is clear from Figure 4.1 that the global minimum is at $x \approx -8$ while there are several local minima, such as at $x \approx -2.3$. The optimization algorithm was applied to the objective function (Equation 4.9) using several starting points to illustrate the point of local convergence. The results of the optimization are shown in Table 4.1 and corroborate with the expectation of local convergence. Looking at the first three rows of Table 4.1, with starting points of $x = -10, -8, -6$, it is expected that the SQP algorithm converges to the global minimum at $x \approx 8$, which the results verify. Examining the next two rows in Table 4.1 where the initial point for the algorithm was $x = -4, -2$ shows that the SQP gets trapped in the local minimum at $x \approx -2.3$. Similar behaviour is found for the rest of the points although it is noticed that for the starting or initial point of $x = +8$ the SQP algorithm was trapped in the local minimum at $x \approx -2.3$ and not at $x \approx 5.1$ as expected which could have been as a result of the step size used in this case.

4.2.2.2 PSO

Particle Swarm Optimization (PSO), introduced in Section 2.2.2, was the second optimization algorithm employed in the NMPC algorithm. The MATLAB implementation of the PSO algorithm in "A Generic Particle Swarm Optimization Matlab Function" [90] was used in this study. PSO is a meta-heuristic algorithm as described in Section 2.2.2 and is considered a global optimization method [91], although it not completely immune to being trapped by local minima. This provides a good platform for comparison with a *gradient-based* optimization method (SQP) .

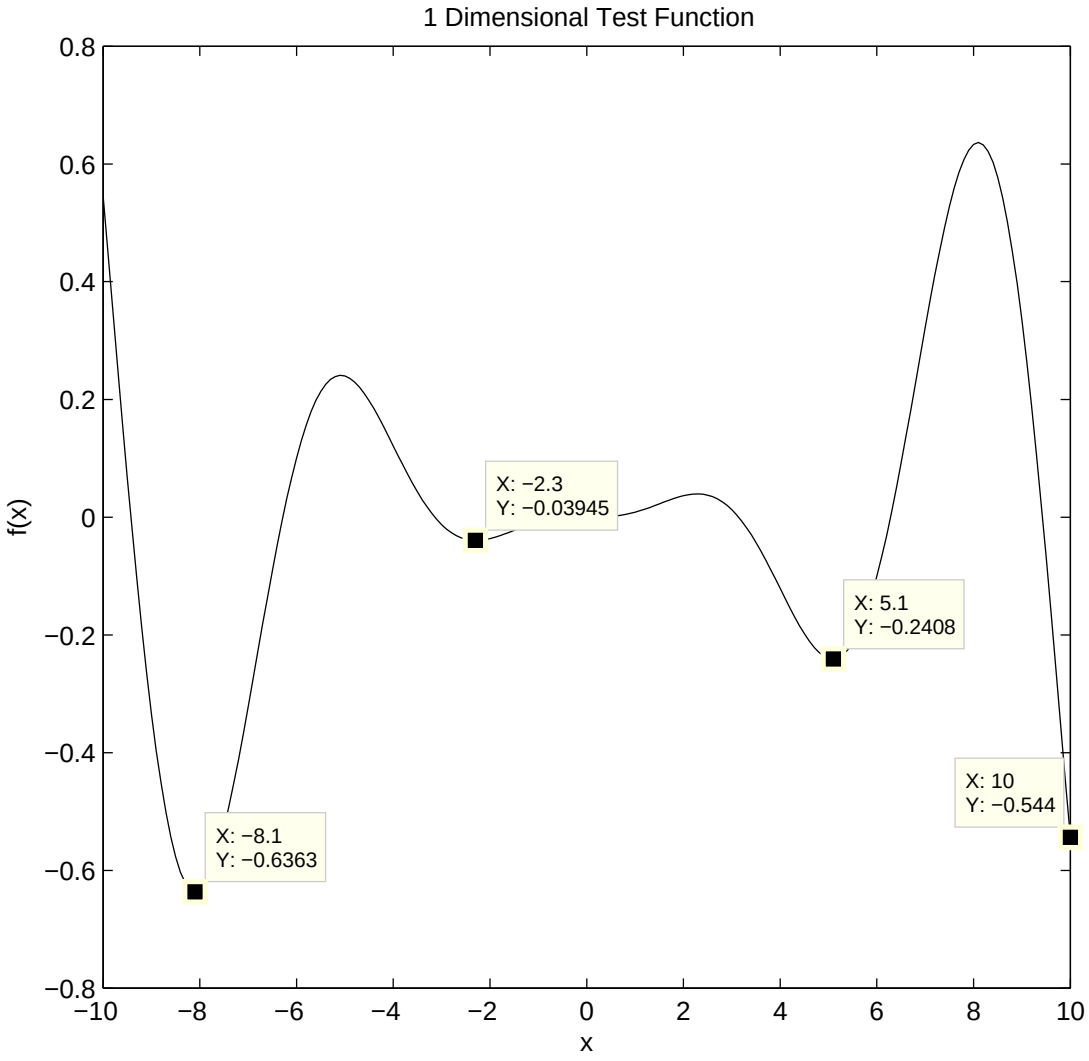


Figure 4.1: Local and Global Minima Test Function for SQP

Table 4.1: Table showing the results of the SQP optimization of Figure 4.1

Starting point	**x-value**	**Function Value**
-10	-8.0962	-0.63635
-8	-8.0962	-0.63635
-6	-8.0962	-0.63635
-4	-2.2889	-0.03945
-2	-2.2889	-0.03945
0	0	0
2	-2.2889	-0.03945
4	5.087	-0.24083
6	5.087	-0.24083
8	-2.2889	-0.03945
10	10	-0.54402

4.3 NMPC Algorithm Implementation [1]

In this section the implementation of the MATLAB NMPC algorithm in [1], which is used in this study, will be discussed. The flow charts of the software provide additional implementation information such as the functions used and their parameters. The goal of this discussion is to understand the implementation in order to be able to fully use and modify functions where required.

There are three main steps or stages in the NMPC algorithm, as mentioned in Section 4.2, which will be described in the following sections.

4.3.1 Step 1: Initialisation

The first step is that of initialisation whereby all the values required for the various variables are initialised before moving onto the solving of the optimal control problem, which is the predominant portion of the NMPC implementation. The variables that are to be initialised include the number of iterations, horizon length, sampling interval/length, initial state (x) values, initial control input (u) values, the required tolerances for the optimization and the type of system i.e. continuous or discrete.

The initial state and control input values (for the first iteration) are provided by the user; these are the initial conditions. In the consequent iterations the initial state values are those obtained after the control input has been applied (the last step in NMPC) and the sample instant is also updated at this point. The consequent initial control input sequences are obtained by using the "*shift technique*" [1] and the results of the optimization algorithm. Since the optimization algorithm calculates the control inputs for the entire horizon while only the first control input is actually applied to the system, the shift technique makes use of the calculated control input values that are not used as the initial control input sequence values for the next time interval. These control input values would otherwise be discarded. Figure, 4.3 outlines the shift technique which is further discussed towards the end of Step 2 (Section 4.3.2).

4.3.2 Step 2: Solving The Optimal Control Problem

Step 2 in the NMPC algorithm implementation, which entails solving the optimal control problem, is the bulk of the computation and comprises several steps. The flow chart in Appendix A.5 displays the high level steps comprising the solution of the Optimal Control Problem which is done by transforming it into a static (constrained nonlinear) optimization problem, as mentioned previously. The static nonlinear optimization problem is then solved using an optimization routine. In this study the SQP optimization algorithm is used as per the MATLAB implementation in the *fmincon* function for solving the optimization problem. The PSO optimization algorithm is also used for the optimization problem (for comparison) and the MATLAB implementation used is that of [90].

The recursive or sequential discretization method described in Section 4.2.1.2, is the method employed for transforming the optimal control problem into the static constrained nonlinear optimization problem. In this step, summarised by Figure 4.2, the solving of the dynamics of the system and the optimization problem are separated. Figure 4.2 shows that the dynamics of the system are computed, for example by an ODE solver if the system is continuous, for initial state and control input values. Following this, the NLP solver or optimization algorithm solves 4.10 for the optimal control sequence given the state values computed by the dynamics solver and the initial control input sequence. This process continues until the optimal control solution for u is obtained.

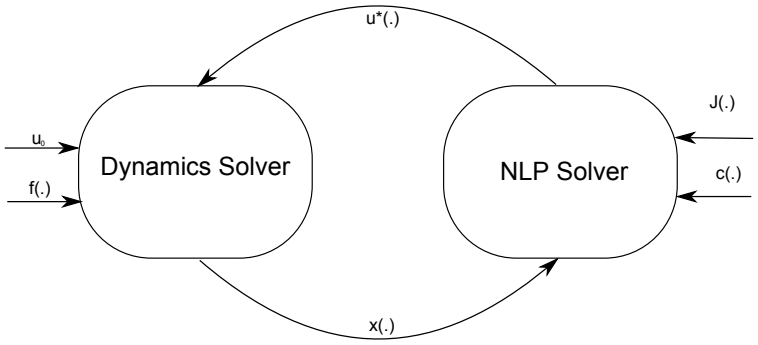


Figure 4.2: Sequential Discretization

The first major step in solving the optimal control problem is to obtain the open-loop solution or in other words the prediction step. This is computed for the entire prediction horizon using the initial state, control and sampling instant values. The implementation caters for systems of difference equations and differential equations. If the system provided is a differential equation then an ODE solver (ODE45 in this case) is used and the tolerances for the ODE solver is required (in Step 1). The next step is in preparation for the solving of the optimization problem (nonlinear constrained). This is where the linear equality and inequality constraints matrices (A, b, Aeq, Beq) which provide the restrictions as defined in the problem definition as well as upper and lower bound restrictions are generated. The MATLAB *fmincon* function solves the constrained optimization problem specified by:

$$\begin{aligned} &\min_{x\in\mathbb{R}^n} f(\mathbf{x}) \\ \text{subject to }: &c_i(\mathbf{x}) = \mathbf{0}, && i = 1,\ldots,m \\ &ceq_j(\mathbf{x}) \leqslant \mathbf{0}, && j = 1,\ldots,p \\ &A.\mathbf{x} \leq b \\ &Aeq.\mathbf{x} = beq \\ &lb \leq \mathbf{x} \leq ub \end{aligned} \tag{4.10}$$

The nonlinear constraints are handled by generating vectors (c, ceq) which contain the computed values of the constraints. The function uses the state values which are computed from the open loop prediction. The terminal constraints, if any, are appended onto the nonlinear constraint vectors.

The last and crucial component required for the optimization problem is that of the cost or objective function, which is provided in the problem definition. In this implementation the optimization function calls the function which calculates the cost function value over the horizon using the state values calculated using the open-loop prediction function, as done with the nonlinear constraints (terminal costs are added to a running cost to obtain the total cost). With this, all the requirements for the optimization function are met and the optimization problem can be solved. The optimization problem is the final major computational step in the solution of the optimal control problem and its solution generates the control input sequence which is used in Step 3.

Before proceeding onto applying the control input, a method of easing and preparing for the restart of the NMPC, the *shift method*, is that of using the portion of the control input sequence that is not applied to the system, which is all but the first control input. In order to compensate for the size of the vector the last control input is duplicated. The *shift method* is one of the "warm startup" methods used in order to provide relevant initial conditions for subsequent time intervals.

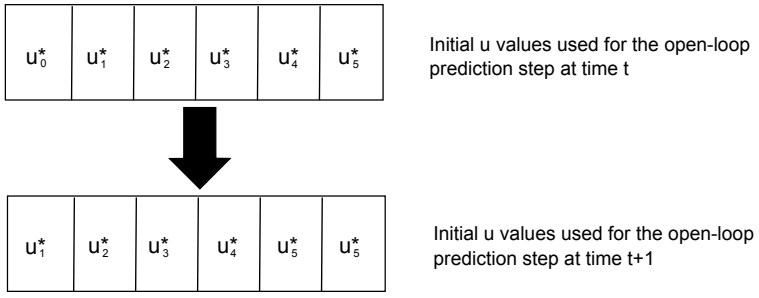


Figure 4.3: Shift Method

4.3.3 Step 3: Applying The Control Input

Once the optimization problem has been solved and the control input sequence has been obtained, the first control element is applied to the system for one sampling interval to obtain the closed-loop states. The closed-loop data is stored. The updated state values are then used in the next iteration after the NMPC iteration counter has been incremented. If the number of iterations are completed then the algorithm terminates.

Step 1: Initialise values u, initial input values, and initial states $x(t_0)$ along with other required parameters such as horizon length, tolerances for ODE etc.
Step 2: At t_0 simulate the system for the prediction horizon i.e. from t_0 to $t_0 + NT$.

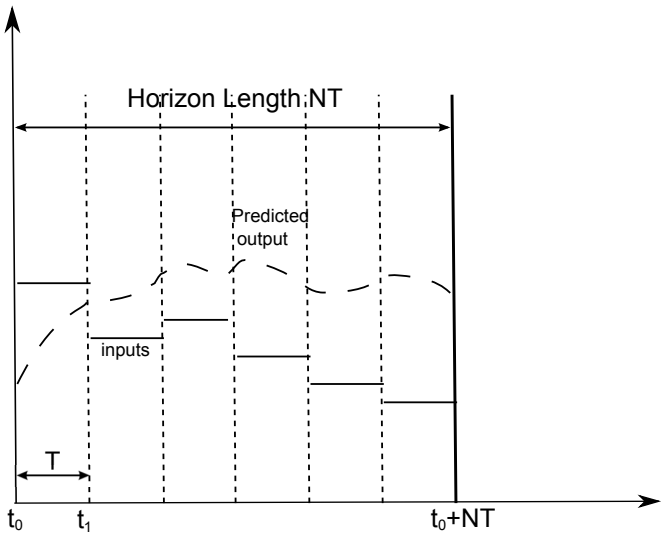


Figure 4.4: NMPC Time Line (Adapted from [1])

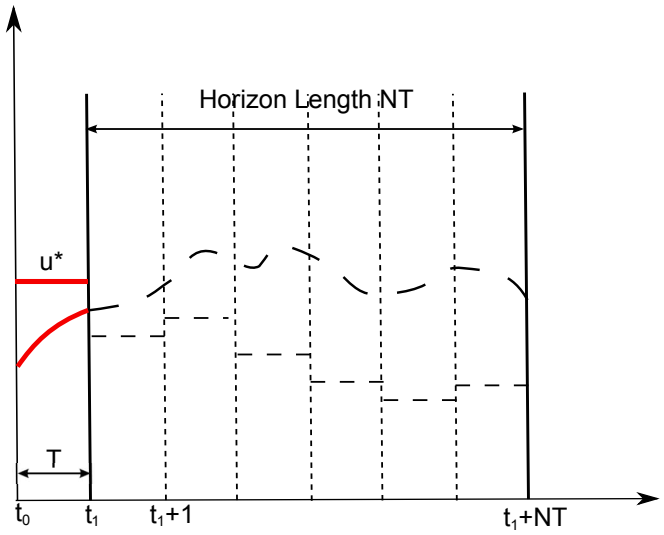


Figure 4.5: NMPC Time Line (Adapted from [1])

Step 3: Optimize problem 4.10 given state values $x(t_0) \dots x(t_0 + NT)$ (obtained from Step 2) to obtain optimal control input sequence $u(t_0) \dots u(t_0 + NT)$ within stopping criterion.
Step 4: Apply the first element of the calculated control input sequence from Step 3.
Step 5: Let new $t_0 = t_0 + 1$ (t_1) and go to Step 1 using the shift method for the initial u values.

Figures 4.4 and 4.5 provide an illustration of the NMPC procedure described in a time line setting. Figure 4.4 illustrates from Step 1-3 and Figure 4.5 covers Step 4-5.

Chapter 5

Optimization Roadmap Methodology and Results

This chapter discusses the approach taken in order to address the research question which, as described earlier, is to provide a methodology that would provide transparency and enable users to gain insight into the optimization problem in the context of NMPC.

The problem was approached in a progressive manner in that firstly the Optimization Roadmap was investigated out of the NMPC context on *unconstrained* **static** optimization problems which was then expanded into an investigation into *constrained* **static** optimization problems. Finally, after validating the methodology in the static optimization domain, the identified approach was then applied and investigated in the NMPC context which can be seen as taking a form of dynamic optimization. However, the underpinning key to the methodology as described in Chapter 4 is that the Optimal Control problem that is to be solved at every time interval is converted to a static constrained nonlinear optimization problem (NLP). The Optimization Roadmap proposed in this study takes advantage of this transformation of the Optimal Control problem into a static nonlinear optimization problem.

In order to attain the goal of insight into the optimization problem, a visual or graphical approach illustrating to the control engineer, in a 2-Dimensional graph, what is required of the optimization algorithm to obtain optimal results is followed. In other words, it presents a graphical representation that would reveal the nature of the optimization problem to be solved. Additionally, situations or regions where the algorithm may encounter problems in terms of local minima (due to the non-convexity of the problem) is illustrated. The graphical nature of this approach lends itself to the intuition of the user and thus presents the obfuscated nature of the (possibly non-convex) optimization problem in a clear intuitive manner.

The graphical representation eliciting the nature of the system, which would bring forth and high-

light the *topography* of the optimization problem, is that of plotting the cost or objective function value against the independent variable(s), in the system. In the case where more than one independent variable is present the norm of the independent variables are used[1].

5.1 Computation Platform

In this Chapter the two optimization algorithms (SQP and PSO) were used and the obtained results contrasted. The MATLAB Technical Computing Platform was used for the computation. The computation time is one of the metrics used when comparing the two optimization algorithms. The computation was done on a computer with the following specifications:

- Operating System: Windows 7 Professional
- Processor: Intel(R) Core(TM)2 Duo CPU E8500@3.16GHz
- Memory: 2.00GB
- Type: 32 Bit
- MATLAB Version: R2011b
- MATLAB Toolboxes:
  - Optimization Toolbox
  - PSO Algorithm [90]
  - NMPC [1]

5.2 Static Optimization

In this section several examples illustrating various aspects of the methodology of obtaining insight into the optimization problem to be solved will be presented. These examples will be in the static optimization domain. Initially, the unconstrained case will be considered followed by the constrained case such that the methodology is validated and can then be extended into the NMPC framework, in Section 5.3. The rationale behind first investigating static optimization examples and then extending into NMPC is that in NMPC, at every time interval, the Optimal Control Problem is converted to a static (nonlinear optimization) problem which is then solved, as mentioned earlier.

[1]This is briefly investigated in the static optimization examples. However, in this study the aim is to prove the concept for single input systems in an NMPC context

5.2.1 Unconstrained Static Optimization

The unconstrained optimization problem is commonly formulated as:

$$\text{Find}\,\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \text{ that minimizes } J(\mathbf{x}) \tag{5.1}$$

where $J(\mathbf{x})$ is the objective function or the function that is to be minimized.

5.2.1.1 Optimization Roadmap For Unconstrained Static Optimization

In the unconstrained static optimization examples presented in this section the following methodology was followed in order to obtain the graphs illustrating the optimization problem to be solved. A flowchart of this methodology is presented in Appendix A, Figure A.1.

1. Start with initial value(s), set the range for the variables and increment size.
2. Evaluate the cost/objective function value at the current point.
3. If there is only one independent variable proceed to step 4 or if there is more than one independent variable calculate the norm of the independent variables.
4. Check whether the range of variable values have completed. If the range of values are exhausted proceed to step 5. If the range of variables values are not exhausted, increment the variable values and return to step 2.
5. Plot the cost function value versus the independent variable (if there is only one independent variable) or the cost function value versus the norm (of the independent variables) if there is more than one independent variable.

5.2.1.2 Example 1: Unconstrained Static Optimization of a 1-Dimensional Quadratic Function

This is an example of unconstrained static optimization of a quadratic function. The aim of this example is to introduce the optimization roadmap methodology practically by beginning with an elementary example which is the 1-Dimensional quadratic function.
Figure 5.1 shows a plot of the 1-Dimensional quadratic function:

$$J(x) = 0.5x^2 - 20 \tag{5.2}$$

which will be considered as the cost or objective function $J(x)$ in this example.

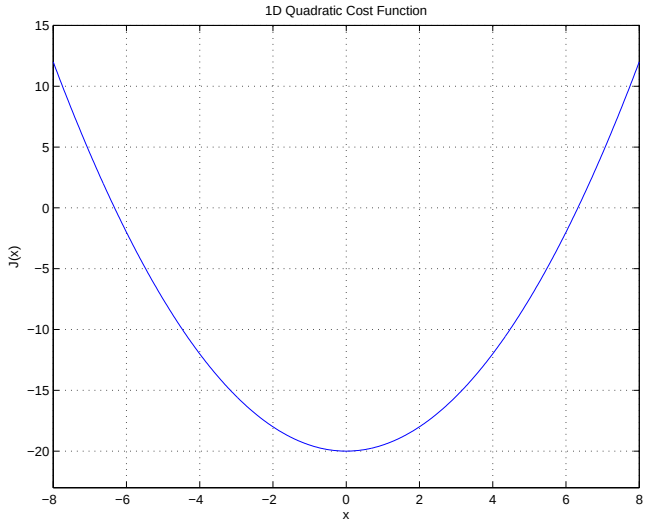


Figure 5.1: 1-Dimensional Quadratic Cost Function

It is clear that there is only one minimum (global minimum) at $x = 0$ in this example, (as is expected in a quadratic function) and as can be seen in Figure 5.2 which is a plot of the cost function value against the independent variable (x). In this example the Figures 5.1 and 5.2 display the same information since there is only one variable (x), which is the independent variable. However the fundamental concept that can be extracted from this plot is that the graphical representation of the system provides a multitude of information including the convexity of the function and the number of minima.

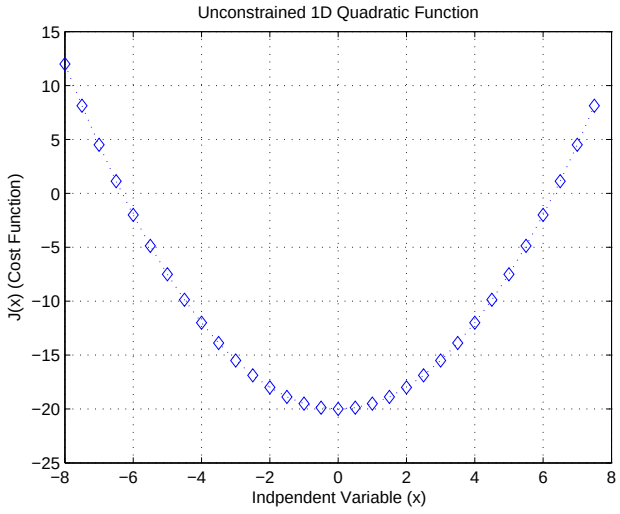


Figure 5.2: Unconstrained 1-Dimensional Quadratic Function - Cost Function Value vs. Independent variable (x)

The goal of the process is to gain insight or transparency into the problem that the optimization

algorithms (SQP and PSO) are required to solve. The insight that is being sought is what sort of issues might the optimization algorithm incur, whether the problems are confined to certain areas or regions for example a local minimum trap. This enables the user to make an informed choice as to what type of optimization algorithm to use and what initial values would yield the best results (in local optimization algorithms).

Returning to Example 1, of a 1-Dimensional quadratic function, it is evident that it is a straightforward optimization problem, due to its convexity, and that the local optimization algorithms (SQP in particular) should not experience any difficulties in finding the minimum. Tables 5.1 and 5.2 show the results obtained when applying the SQP and PSO algorithms, respectively, to the 1-Dimensional quadratic function. The SQP algorithm was run with several starting points or initial values (from $x = -8$ to $x = 8$) as shown in Table 5.1 and the PSO algorithm was applied several times since the algorithm randomly selects the initial points for the particles in the search space. It is noted from these results that both optimization algorithms were able to locate the minimum for every run of the algorithm. However, note should be taken of the duration of time taken by the algorithms. The SQP algorithm is noticeably quicker than the PSO algorithm in optimizing the function.

Table 5.1: Optimization of 1-Dimensional Unconstrained Quadratic Function using SQP

x **Initial Value**	**Minimum x Value**	**Function Value at Minimum**	**Time Taken by SQP (sec)**
-8	3.07E-08	-20	0.294000138
-7	-3.13E-08	-20	0.183917729
-6	-6.98E-08	-20	0.180077021
-5	3.86E-08	-20	0.223161127
-4	-8.65E-08	-20	0.17432275
-3	-2.93E-08	-20	0.169542239
-2	-2.38E-08	-20	0.164457731
-1	7.23E-08	-20	0.153200148
0	0	-20	0.083755318
1	-2.00E-08	-20	0.157072226
2	-4.44E-08	-20	0.190659343
3	-6.18E-10	-20	0.153961434
4	1.82E-07	-20	0.165365841
5	1.38E-08	-20	0.167920707
6	5.09E-08	-20	0.179298272
7	-8.76E-08	-20	0.170233347
8	-1.75E-08	-20	0.165358403

Table 5.2: Optimization of 1-Dimensional Unconstrained Quadratic Function using PSO

Minimum x Value	**Function Value at Minimum**	**Time Taken by PSO (sec)**
5.56E-08	-20	1.168204006
2.28E-08	-20	1.108084439
-1.96E-08	-20	1.154321379
2.71E-08	-20	1.206938694
5.48E-08	-20	1.170580679
5.72E-08	-20	1.201406923
-2.81E-08	-20	1.035409804
-7.85E-10	-20	1.171796666
5.87E-08	-20	1.172807941
-1.59E-08	-20	1.148387944
4.33E-08	-20	1.206499192
-5.01E-08	-20	1.187469964
-5.94E-08	-20	1.113293779
-2.49E-08	-20	1.203570798
-2.65E-08	-20	1.198897656
-5.37E-08	-20	1.162605938
-2.56E-08	-20	1.159195029

5.2.1.3 Example 2: Unconstrained Static Optimization of a 2-Dimensional Quadratic Function

This example extends Example 1 to two dimensions in order to illustrate the concept of the plotting of the cost value versus the norm of the independent variables. Although this concept will not be extended into the NMPC applications, the aim is to display that this concept (with additional work) may be extended to the multiple input case for NMPC. The methodology is not limited to two dimensions since the norm may be calculated for higher dimensions as well.

Extending Example 1 to 2-dimensions or $n = 2$ in the unconstrained static optimization problem 5.1:

$$\text{Find } \mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}, \text{ that minimizes } J(\mathbf{x}). \tag{5.3}$$

The plot of the objective function, $J(\mathbf{x})$, for the 2-Dimensional quadratic objective function is shown in Figure 5.3.

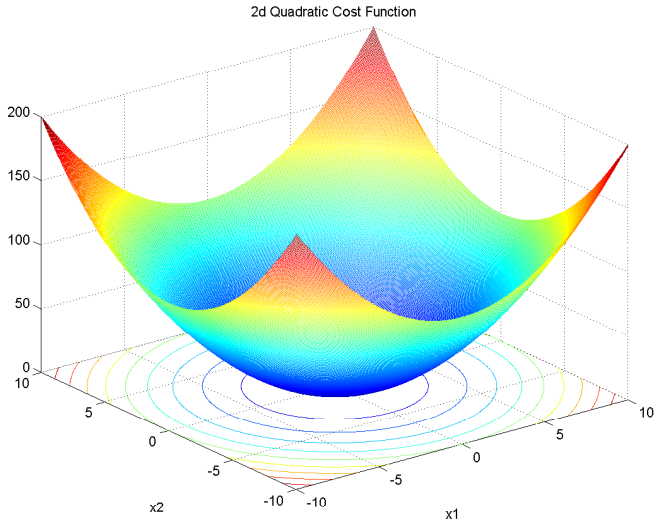


Figure 5.3: 2-Dimensional Quadratic Cost Function

Once again, it evident that the minimum (global) is located at $x_1, x_2 = 0$. The methodology proposed is that the independent variable is plotted against the corresponding cost or value of the objective function. However, it is noted that both x_1 and x_2 are independent variables which leads one to a problem: which variable should be used in the plot? The solution to this issue (which is not confined to two dimensions but to any dimension greater than or equal to two), is that the norm of the independent variables be used. The 2-norm or Euclidean norm which is defined by:

$$\|\mathbf{x}\|_2 = \left(\sum_{j=1}^{n} x_j^2 \right)^{1/2} = \sqrt{x_1^2 + \ldots + x_n^2}$$

will be used for this purpose.

Figure 5.4 shows the plot of the cost function value versus the norm (of the independent variables x_1 and x_2) for the unconstrained 2-Dimensional quadratic function. Despite being a rather simple example, some very useful or valuable information can be gleaned from the plot as mentioned previously. Although not being the focus or goal of the methodology, the first and possibly most obvious piece of information is that one can see that the minimum of the function is zero and the norm will only have a value of zero if both x_1 and x_2 are zero since each element is *squared*. Furthermore, and more importantly, it is noted that the plot reflects the curvature of the objective function i.e. a smooth descent to the minimum at $(0, 0)$.

As mentioned previously, from the graphs of the objective function in Figure 5.3 and more importantly from the cost versus norm plot in Figure 5.4, it is clear that no local minima are present in this function. Thus from an optimization point of view a gradient based algorithm such as SQP should not experience any problems in reaching the global minimum.

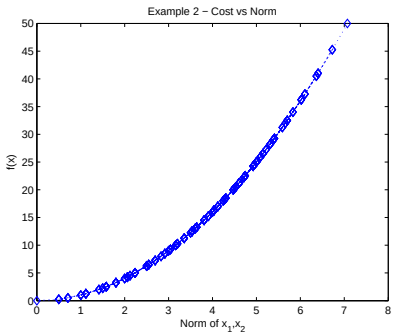


Figure 5.4: 2-Dimensional Unconstrained Quadratic Function - Cost Function Value vs. Norm

Table 5.3 displays the data from applying the SQP algorithm to the 2-Dimensional quadratic function (Figure 5.3) at varying initial values or starting points. The results of the SQP optimization validate the expectation that the global minimum should be found without any difficulty. It is noticed in Table 5.3 that the global minimum at the function value of $f(x) = -20$ is located by the SQP algorithm from all of the starting points.

Table 5.3: Optimization of 2-Dimensional Unconstrained Quadratic Function using SQP

x_1 **Initial Value**	x_2 **Initial Value**	x_1, x_2 **Norm Value**	**Minimum** x_1 **Value**	**Minimum** x_2 **Value**	**Function Value at Minimum**	**Time Taken by SQP (sec)**
-5	-5	7.0711	-2.49E-08	2.45E-08	-20	0.24306
1.5	-4.5	4.7434	-1.67E-08	7.34E-08	-20	0.11899
3.5	-4	5.3151	-7.69E-08	1.01E-07	-20	0.13093
0	-3.5	3.5000	0.0023997	2.99E-08	-20	0.11124
1.5	-1.5	2.1213	1.13E-08	3.48E-08	-20	0.14311
2	-1	2.2361	7.92E-08	-3.29E-09	-20	0.14119
-3	0	3.0000	-1.39E-08	0	-20	0.13149
1	0.5	1.1180	-8.06E-09	-3.30E-08	-20	0.16679
0.5	4	4.0311	-4.17E-09	-1.55E-08	-20	0.14246
1.5	4.5	4.7434	-1.67E-08	-7.34E-08	-20	0.13052

Table 5.4 displays data from applying the PSO optimization algorithm to the 2-Dimensional quadratic function shown in Figure 5.3. Similarly to the SQP algorithm, the PSO algorithm located the global minimum at every run (PSO assigns random starting points in the search space), as was expected.

5.2.1.4 Example 3: Unconstrained Static Optimization of a 2-Dimensional Rastrigin Function

As with Example 2, this example deals with a function in two dimensions. The function used in this example is the well known Rastrigin function [92]. This example also aims to illustrate and validate the norm concept in the static optimization setting while displaying the insight gained by the cost versus norm plot.

Table 5.4: Optimization of 2-Dimensional Unconstrained Quadratic Function using PSO

Minimum x_1 Value	**Minimum x_2 Value**	**Function Value at Minimum**	**Time Taken by PSO (sec)**
-1.49E-08	-3.92E-08	-20	2.060354937
2.83E-08	6.07E-08	-20	1.78774888
3.22E-09	3.47E-08	-20	2.047759202
-4.72E-09	2.36E-08	-20	1.925193021
2.02E-08	3.34E-08	-20	2.026585964
-1.57E-08	2.94E-08	-20	1.920733675
2.58E-07	-1.79E-07	-20	1.702805613
-3.45E-08	1.99E-08	-20	1.828653969
-1.38E-08	-3.77E-08	-20	1.966814115
-7.33E-09	2.49E-08	-20	1.934647583

The Rastrigin function, defined by Equation 5.4, is a popular function used in testing optimization algorithms. A plot of the 2-dimensional Rastrigin function is shown in Figure 5.5 and elucidates why this is so. It is a demanding function to optimize due to the abundance of local minima while its global minimum is $f(\mathbf{x}) = 0$ at $\mathbf{x} = 0$.

$$f(x) = 10n + \sum_{i=1}^{n}(x_i^2 - 10\cos(2\pi x_i)) \quad x_i \in [-5.12, 5.12] \tag{5.4}$$

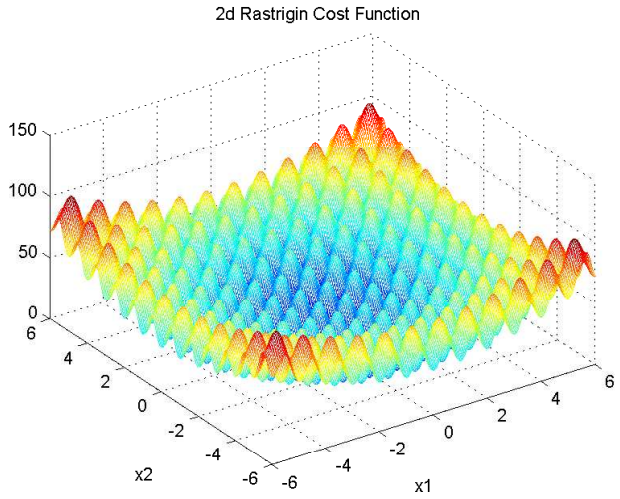


Figure 5.5: 2-Dimensional Rastrigin Function

Inspecting Figure 5.6, which is a representation of the cost versus norm (of x_1 and x_2) plot that is to be used to characterise the function and provide insight into the optimization of this function, it is noticed that the plot indicates (a deceiving) smooth curvature. However, as is clearly dis-

played by the plot of the Rastrigin function in Figure 5.5, this is not an accurate depiction. Upon investigation it is noted that an extremely important point is highlighted by this inaccuracy. The problem is due to the increment size, between the points being used to calculate the norm and plot the graph, being too large which led to the function being inaccurately reflected.

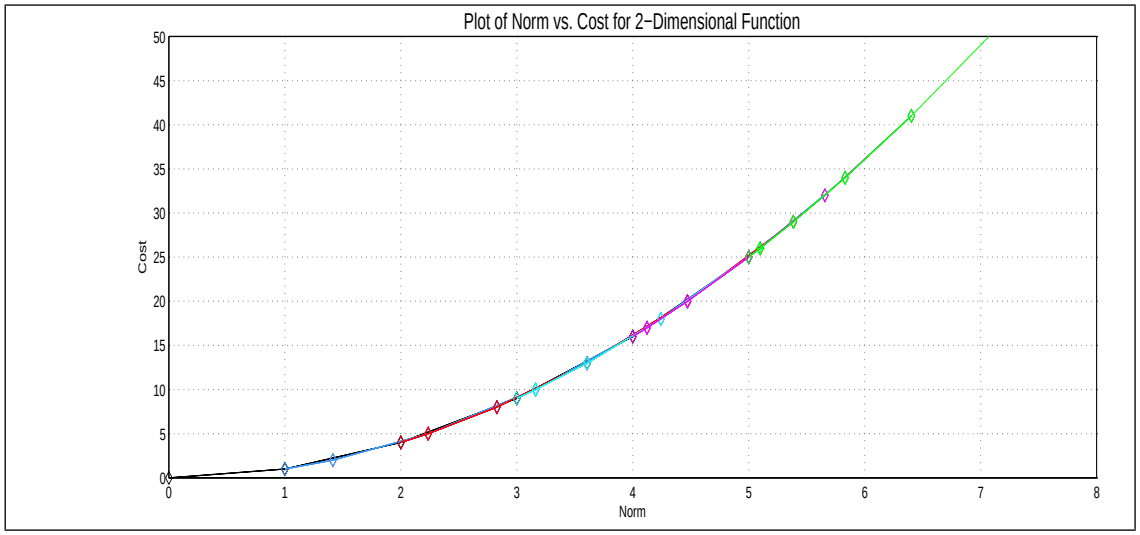


Figure 5.6: 2-Dimensional Rastrigin Function- Cost Function Value vs. Norm with increments of 1

Now, by decreasing the magnitude of the increment it is expected that the cost vs. norm plot should encapsulate and reflect the function more closely. This is depicted in Figure 5.7 (which has been simplified for clarity [2]) where it is now evident that the Rastrigin function is indeed a challenging function to minimize by virtue of the fact that there are several local minima within which the optimization algorithms may be trapped. The use of the norm (of x_1 and x_2) in this case is for dimension reduction so that a two dimensional plot is obtained since both x_1 and x_2 are independent variables. However, with this advantage of the simplified plot comes the disadvantage that more *housekeeping* work is required in the background in order to keep track of which x_1 and x_2 values correspond to the norm since differing values of x_1 and x_2 may work out to the same norm value. This example is presented for illustration purposes that this approach, of using the norm for multiple independent variables, may be used in this manner for insight into the optimization problem. This study will focus on the methodology for single input systems which, in the static optimization case, translates to one independent variable. The concept of this approach is fundamentally proven in this example and it is recommended for further exploration and further research for Multiple Input systems in NMPC.

Although this example is mostly for illustration purposes (of the norm concept) the optimization algorithms were applied to this problem. Tables 5.5 and 5.6 display the results obtained by using

[2]The red and blue lines differentiate the graphs obtained for the different initial values.

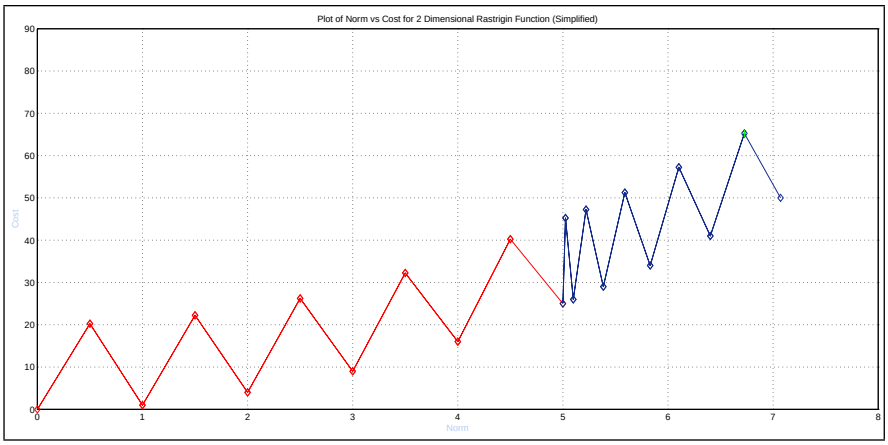


Figure 5.7: 2-Dimensional Rastrigin - Cost Function Value vs. Norm Plot depicting curvature of the function

the SQP algorithm and the PSO algorithm to optimise the 2-Dimensional Rastrigin function. The initial values selected for the SQP algorithm were chosen in order to illustrate the insight provided by the methodology. As mentioned in previous examples, the PSO algorithm randomly distributes the particles within the search space.

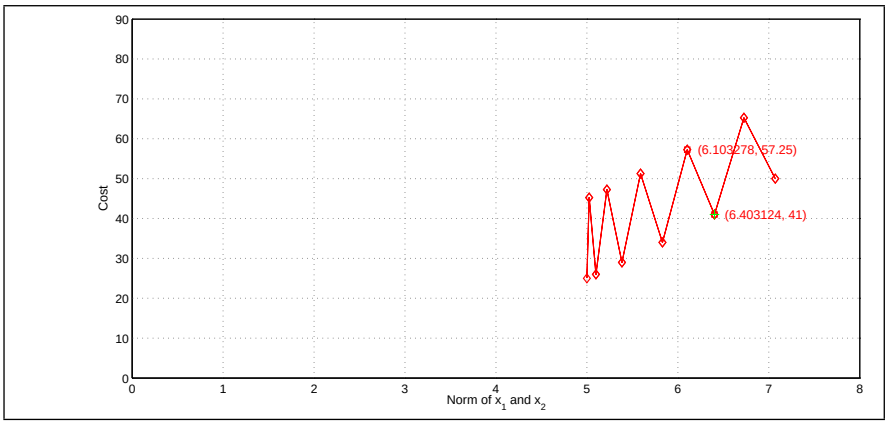


Figure 5.8: 2-Dimensional Rastrigin Function - Cost Function Value vs. Norm plot illustrating a local minimum trap

Figure 5.8 displays a simplified version of the cost versus norm plot which only displays the region of interest for this discussion. In the first row of Table 5.5, the initial conditions were chosen as $(3.6, 5)$ which corresponds to the norm value of 6.162. This falls within the range of the labelled points in Figure 5.8. When applying the SQP algorithm (row 1 in Table 5.5) it is noticed that it gets trapped at a local minimum i.e. it does not reach the global minimum of $f(\mathbf{x}) = 0$ at $x_1, x_2 = 0$. From Figure 5.8, this is what is expected and the cost value of the local minimum that SQP converged to (40.7929) is approximately that depicted in Figure 5.8 (i.e. ≈ 41).

Row 2 and 3 in Table 5.5 highlight the scenario of the norm whereby the norm magnitudes are the same (2.5 in this case) while the variable values (x_1 and x_2) are different. In this case the values of x_1 and x_2 are swapped around. In both these cases the SQP algorithm gets stuck in local minima however the local minima are different i.e. at different locations with differing function values. This is illustrated in Figure 5.10 which is also a simplified version of the cost versus norm plot depicting the region of interest.

The last row in Table 5.5 is another example where the illustration depicts the local minimum that SQP gets trapped in. Figure 5.11 illustrates this phenomenon where the algorithm gets trapped at the function value of $f(x) \approx 16.9$ when starting at $x_1 = -3, x_2 = 3.5$. These norm values also have several overlapping curves which have been omitted for clarity.

Figure 5.9 shows all the different cost vs. norm plots, and as can be seen the tracking of the coordinate values and additional *housekeeping* becomes vital in these scenarios of multiple independent variables. This is required in order to extract the valuable information providing insight from all the curves.

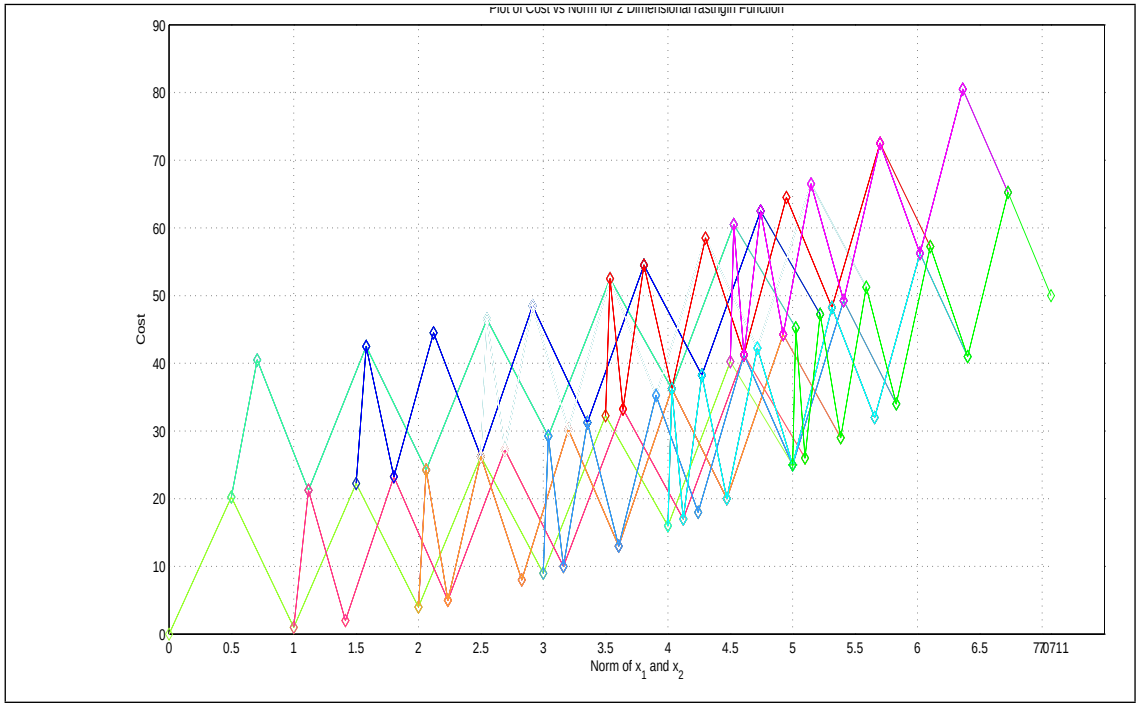


Figure 5.9: 2-Dimensional Rastrigin - Cost Function Value vs. Norm Plot Illustrating All Curves

Although, as mentioned previously, PSO is at times susceptible to getting trapped in local minima, it is noticed in Table 5.6 that for most runs of the PSO algorithm, it was able to get to the global minimum of $f(x) = 0$. There are instances in Table 5.6, in row 1 and 8, where the PSO algorithm got close to, but did not actually reach the global minimum. In both instances the algorithm

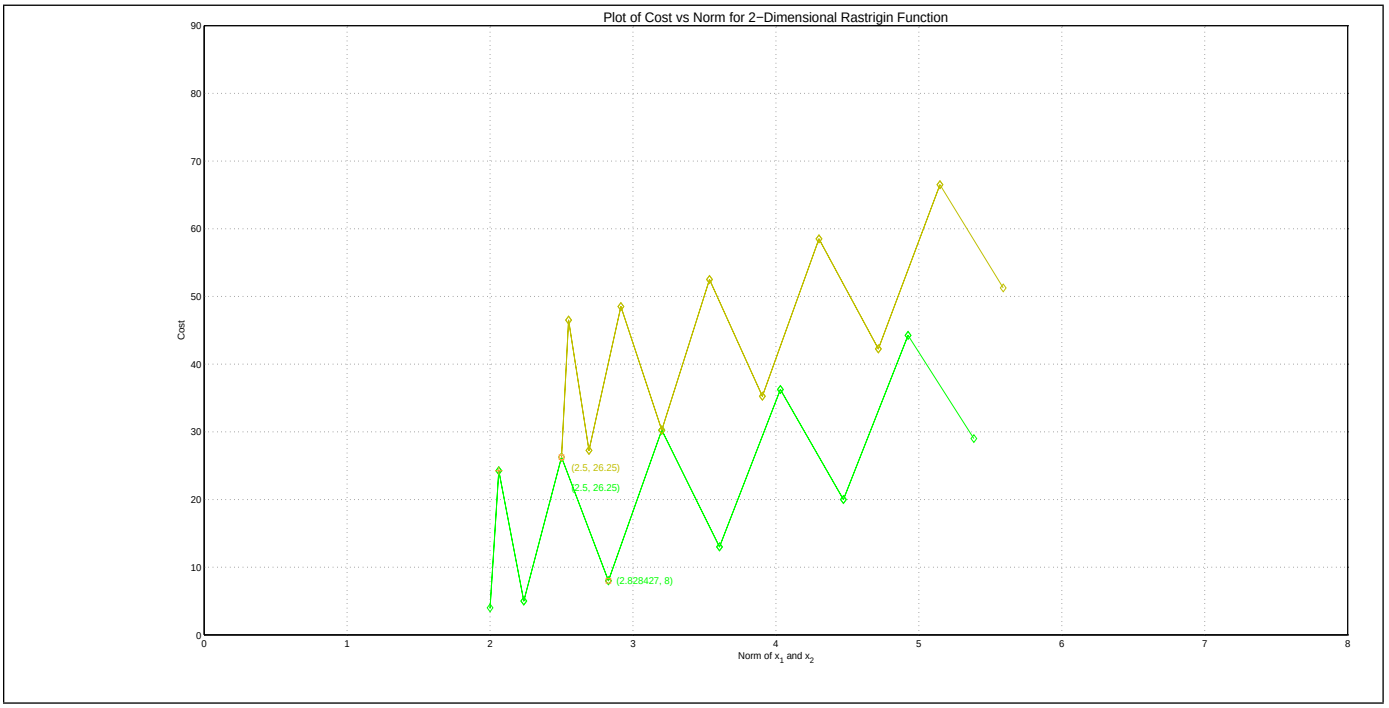


Figure 5.10: 2-Dimensional Rastrigin - Cost Function Value vs. Norm Plot Illustrating Overlapping Norms

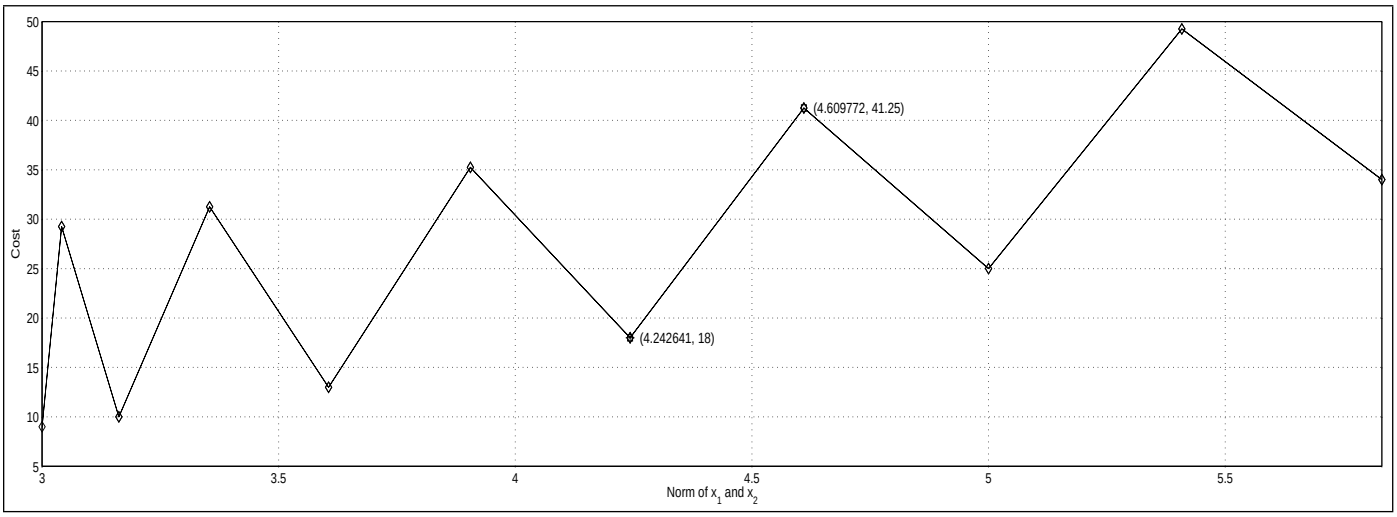


Figure 5.11: 2-Dimensional Rastrigin - Cost Function Value vs. Norm Plot Illustrating a Local Minimum Trap

seemed to converge to the same point. This could possibly be improved by increasing the number of particles although in this set of results for this problem, the PSO algorithm was able to locate the global minimum 80% of the time. [3]

Table 5.5: Optimization of 2-Dimensional Unconstrained Rastrigin Function using SQP

x_1 **Initial Value**	x_2 **Initial Value**	x_1, x_2 **Norm Value**	**Minimum** x_1 **Value**	**Minimum** x_2 **Value**	**Function Value at Minimum**	**Time Taken by SQP (sec)**
3.6	5	6.1612	3.9798	4.9747	40,7929	0.227403
1.5	2	2.5000	-1.9899	-3.9798	19.89907	0.212324
2	1.5	2.5000	-1.9899	-1.9899	7.959662	0.26142
-3	3.5	4.6098	-3.9798	0.9950	16.9142	0.19559

Table 5.6: Optimization of 2-Dimensional Unconstrained Rastrigin Function using PSO

Minimum x_1 **Value**	**Minimum** x_2 **Value**	**Function Value at Minimum**	**Time Taken by PSO (sec)**
-2.95E-09	0.994958629	0.994959057	3.478421542
-0.000283398	4.01E-05	1.63E-05	2.787206216
3.57E-08	-2.97E-08	4.28E-13	3.549892987
7.27E-08	-1.17E-07	3.77E-12	3.418357018
-8.55E-09	-9.82E-09	3.38E-14	3.437346992
1.34E-08	3.82E-08	3.25E-13	3.515967813
-1.38E-07	2.03E-07	1.20E-11	3.368682648
-3.61E-08	0.99495858	0.994959057	3.471443891
-3.72E-08	1.16E-07	2.95E-12	3.51076077
-7.10E-08	3.73E-08	1.28E-12	3.397381052

[3]It should also be noted that although, it is a challenging problem, it is not a high dimensional problem and the results for the PSO algorithm at higher dimensions could vary. However, the thrust of the results are for comparison purposes (with SQP) which show that PSO certainly does perform much better in terms of locating the global minimum than the SQP algorithm albeit with the drawback of taking more time.

5.2.2 Constrained Static Optimization

The previous section dealt with the unconstrained static optimization case. In this section the constrained static optimization problem will be investigated.

The constrained static optimization problem or NLP (as already defined in Section 2.2) is provided again below for ease of reference.

$$\begin{aligned} &\min_{x\in\mathbb{R}^n} f(\mathbf{x}) && && \text{(2.4 revisited)}\\ &\text{subject to :} h_i(\mathbf{x}) = \mathbf{0}, && i = 1,\ldots,m\\ &\qquad\qquad g_j(\mathbf{x}) \leqslant \mathbf{0}, && j = 1,\ldots,p \end{aligned}$$

where $m < n$ and $f : \mathbb{R}^n \to \mathbb{R}$ is the objective function to be minimized, $h : \mathbb{R}^n \to \mathbb{R}^m$ are the equality constraints and $g : \mathbb{R}^n \to \mathbb{R}^p$ are the inequality constraints.

5.2.2.1 Optimization Roadmap For Constrained Static Optimization

In the constrained static optimization examples presented in this section the following methodology was followed in order to obtain the graphs illustrating the optimization problem to be solved. A flowchart of this methodology is presented in Appendix A, Figure A.2.

1. Start with initial value(s), set the range for the variables and increment size.
2. Evaluate the cost/objective function value at the current point.
3. If there is only one independent variable proceed to step 4 or if there is more than one independent variable calculate the norm of the independent variables.
4. Evaluate the constraints and test if they are obeyed or violated.
5. Check whether the range of variable values have completed. If the range of values are exhausted proceed to step 6. If the range of variable values are not exhausted, increment the variable values and return to step 2.
6. Plot the cost function value versus the independent variable (if there is only one independent variable) or the cost function value versus the norm (of the independent variables) using appropriate markers to illustrate constraint adherence or violation.

5.2.2.2 Example 1: Nonlinear Inequality Constrained Static Optimization of a 1-Dimensional Quadratic Function

This example of static nonlinear constrained optimization aims to show how constraints can be incorporated into the optimization roadmap plots delineating the optimization problem.

The first static constrained optimization example that is investigated is a 1 dimensional quadratic cost function $f(x) = 0.5x^2 - 20$ which is constrained by the nonlinear inequality constraint $x^2 \sin(x)\cos(x) - 2 \leq 0$. Defining this problem in the standard form:

$$\begin{aligned} \min_{x} \quad & f(x) = 0.5x^2 - 20 \\ \text{subject to :} \quad & x^2 \sin(x)\cos(x) - 2 \leqslant 0 \end{aligned} \tag{5.5}$$

A plot of this problem is shown in Figure 5.12. The red curve in the plot is the objective or cost function to be minimized, the magenta curve is the nonlinear constraint function that needs to be adhered to and the blue line denotes the $y = 0$ line. Inspecting Figure 5.12, it is noticed that the constraint has limited the possible values that the solution to the optimization problem defined in 2.4 may take (as is expected of a constraint). Once again, by inspection, the approximate *feasible* regions in the plot or the values of x that satisfy the constraint are where the constraint function (the magenta curve) is below the $y = 0$ line which are:

$$x \lesssim -6.2, \tag{5.6}$$

$$-4.8 \lesssim x \lesssim -2.9, \tag{5.7}$$

$$-2.1 \lesssim x \lesssim +3.3, \tag{5.8}$$

$$+4.6 \lesssim x \lesssim +6.4. \tag{5.9}$$

In effect, discontinuities have been created by the constraint and as such it is expected that some problems will be experienced by the SQP optimization algorithm depending on the initial conditions chosen. This is significant since the objective function is quadratic with one (global) minimum.

Due to it being a one dimensional problem this example has only one independent variable and thus it will suffice to plot the independent variable (x) against the cost or objective function value, $f(x)$. This is displayed in Figure 5.13 which clearly brings out regions where constraints are violated by using red squares to demarcate them. The regions with blue diamonds are the feasible regions or regions where the constraints are met. The plot in Figure 5.13 is extremely useful in gaining insight into the problem in terms of where the feasible regions are and where a gradient based optimization algorithm may face problems in getting to the global minimum.

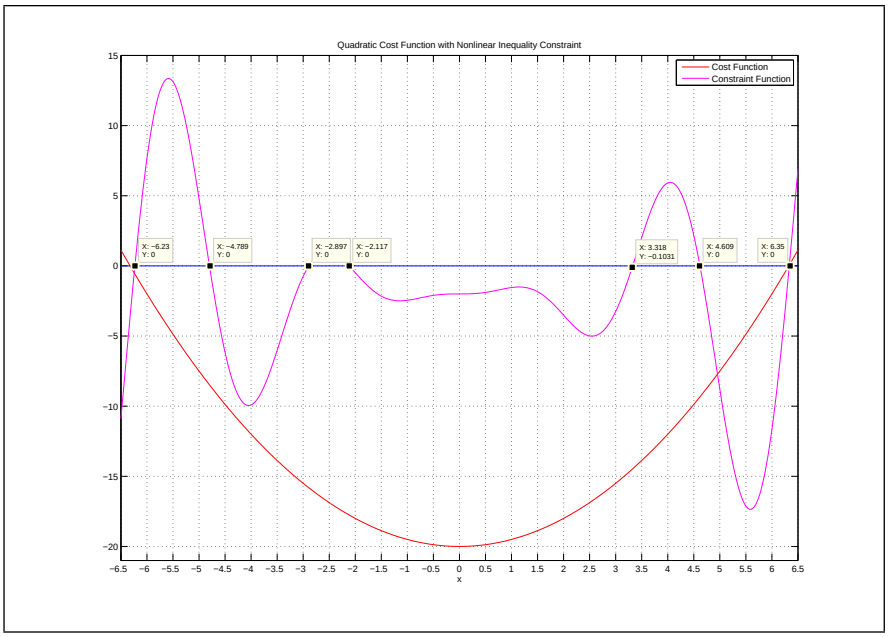


Figure 5.12: 1-Dimensional Quadratic Cost Function With a Nonlinear Constraint Function

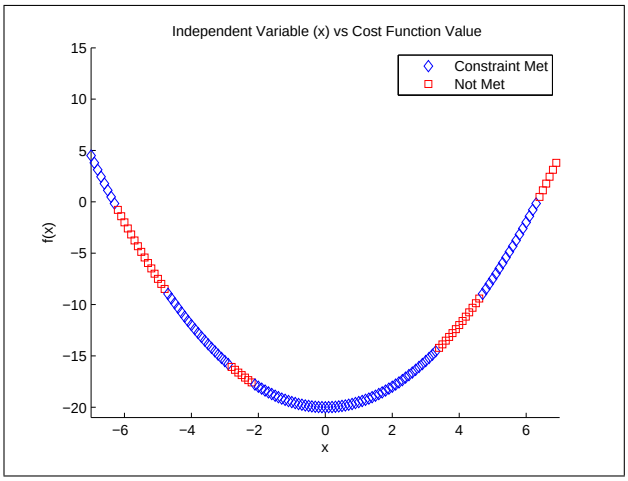


Figure 5.13: 1-Dimensional Quadratic Cost Function Inequality Constrained by a Nonlinear Function

Table 5.7 displays the results from applying SQP to the problem defined by 5.5. The SQP optimization algorithm was applied to the problem for various starting points covering the different regions in the problem. As expected the algorithm did not reach the global minimum when the initial values or the starting points were *above* the infeasible regions (shown in Figure 5.13). Examples of these are rows $1-3$. However, an interesting outcome is that when the initial value is not very *close* to the infeasible region the SQP algorithm is able to "*step over*" the infeasible region and locate the global minimum such as in rows $6, 7$ and $33, 34$. This is due to the step size calculated by the SQP algorithm. When the initial points were in the regions violating constraints (infeasible regions), the algorithm moves the initial value to the boundary of the infeasible region i.e. the

closest value that is feasible. Thus it is noticed that some initial conditions which are in infeasible regions are sometimes able to locate the global minimum (because of the above phenomenon of the algorithm "*stepping over*" the infeasible regions) or in other cases they get trapped by the upper bound of the infeasible region. Examples of the first case are rows $4, 5, 27, 28$ and the second case are rows $3, 29$.

Figure 5.14 is a plot of the norm of x ($\|x\|_2$) vs. the cost ($f(x)$). This plot conveys the same information as Figure 5.13 but in a condensed manner. It is noticed that in some regions the plot displays that constraints are both met and not met, which is valid (due to the nature of the norm), however one may need to go back to data to confirm exactly which points obey the constraints and which do not when there is an overlap. Essentially, the graph clearly displays where no problems should be encountered (such as the regions where the norm value is between 0 and 2). It also displays the *overlap* areas where the data may be consulted for confirmation of problematic points (such as between norm values 3.5 and 4.8).

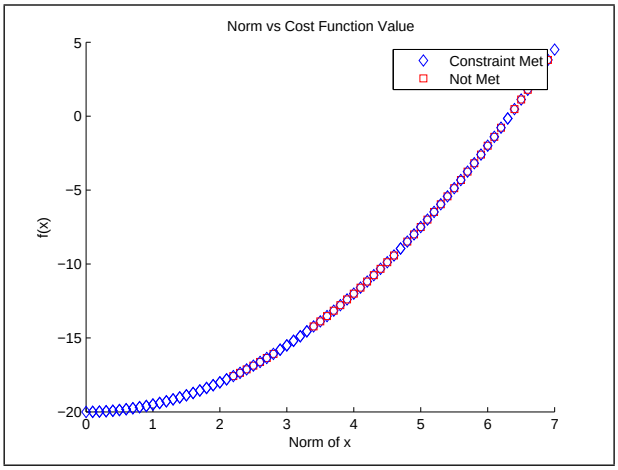


Figure 5.14: 1-Dimensional Constrained Quadratic Function - Cost Function Value vs. Norm

5.2.2.3 Example 2: Nonlinear Inequality Constrained Static Optimization of a 2-Dimensional Quadratic Function

Extending Example 1, this example investigates a 2-Dimensional quadratic cost function that is constrained by a nonlinear function. As before, with multi-dimensional problems the norm of the independent variables is used in the plot. This example also displays how the constraints may be incorporated into the cost versus norm plot. The problem is defined in 5.10 and depicted in Figures 5.15 and 5.16.

Table 5.7: Optimization of 1-Dimensional Nonlinear Constrained Quadratic Function using SQP

x **Initial Value**	**Minimum** x **Value**	**Function Value at Minimum**	**Time Taken by SQP (sec)**
-6.6	-6.2316	-0.5836	0.1468
-6.2	-6.2316	-0.5836	0.131
-5.8	-6.2316	-0.5836	0.145
-5.4	0	-20	0.0834
-5	0	-20	0.0962
-4.6	0	-20	0.0844
-4.2	0	-20	0.0853
-3.8	-2.8923	-15.8173	0.1935
-3.4	-2.8923	-15.8173	0.1463
-3	-2.8923	-15.8173	0.13
-2.6	-2.8923	-15.8173	0.1648
-2.2	0	-20	0.0842
-1.8	0	-20	0.0864
-1.4	0	-20	0.0882
-1	0	-20	0.0835
-0.6	0	-20	0.1
-0.2	0	-20	0.1027
0.2	0	-20	0.098
0.6	0	-20	0.0998
1	0	-20	0.0832
1.4	0	-20	0.12
1.8	0	-20	0.1501
2.2	0	-20	0.1352
2.6	0	-20	0.0863
3	0	-20	0.0875
3.4	0	-20	0.0862
3.8	0	-20	0.1015
4.2	4.618	-9.3368	0.1712
4.6	4.618	-9.3368	0.1167
5	4.618	-9.3368	0.133
5.8	0	-20	0.1012
6.2	0	-20	0.0883
6.6	0	-20	0.0893

$$\min_{x} \quad f(x) = x_1^2 + x_2^2 - 20 \tag{5.10}$$
$$\text{subject to :} \quad x_1^2 sin(x_1) - 1 \leqslant 0$$

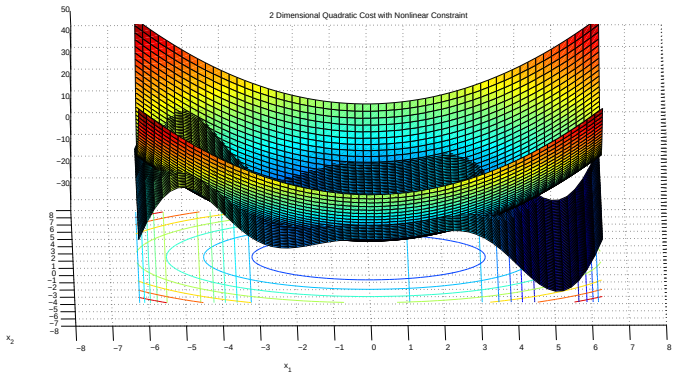


Figure 5.15: 2-Dimensional Quadratic Cost Function with Nonlinear Constraint

Examining Figures 5.15 and 5.16, it is noticed that the constraint imposed upon $f(\mathbf{x})$ i.e $c(\mathbf{x}) \leq 0$ is obeyed for the region $-3.5 \lesssim x_1 \lesssim 1$ which surrounds the origin (where the global minimum is situated). The constraint is also satisfied in the region $+3 \lesssim x_1 \lesssim 6$. Thus, it is expected that the Optimization Roadmap plot reflect this information. Figure 5.17(a), which is the Optimization Roadmap plot, follows the same convention as Example 1 where the region of only blue diamonds are feasible regions while the regions of only red squares are infeasible regions or regions where constraints are not obeyed. The regions of overlapping blue squares means that at that norm value, for some x_1 and x_2 values, the constraints are obeyed while at the same norm value and different x_1 and x_2 values the constraints are violated.

The cost versus norm (Optimization Roadmap) plot discloses that the minimum cost or objective function value $f(\mathbf{x}) = -20$. The plot also reveals that the minimum occurs at the point $x_1, x_2 = 0$ since one of the properties of the 2-norm is that $\|\mathbf{x}\|_2 = 0 \, \textit{iff} \, \mathbf{x} = 0$. As mentioned previously, although the actual solution of the optimization problem (finding the minimum) is not the goal of the methodology, it may be revealed as a "by-product" as in this example. Another very important insight that is provided by the plot is that the curvature of the function is smooth. Since the quadratic function used in this example is well known, this may be taken for granted. However, as shown in the unconstrained examples (Example 3) the curvature information provides immense insight and as such it shows that there are no local minima present in the objective function. The gradient based SQP optimization algorithm may experience problems due to the constraints as was the case in Example 1 of the constrained static optimization discussion.

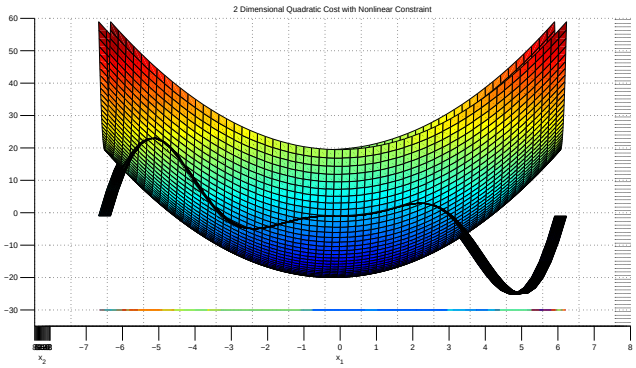
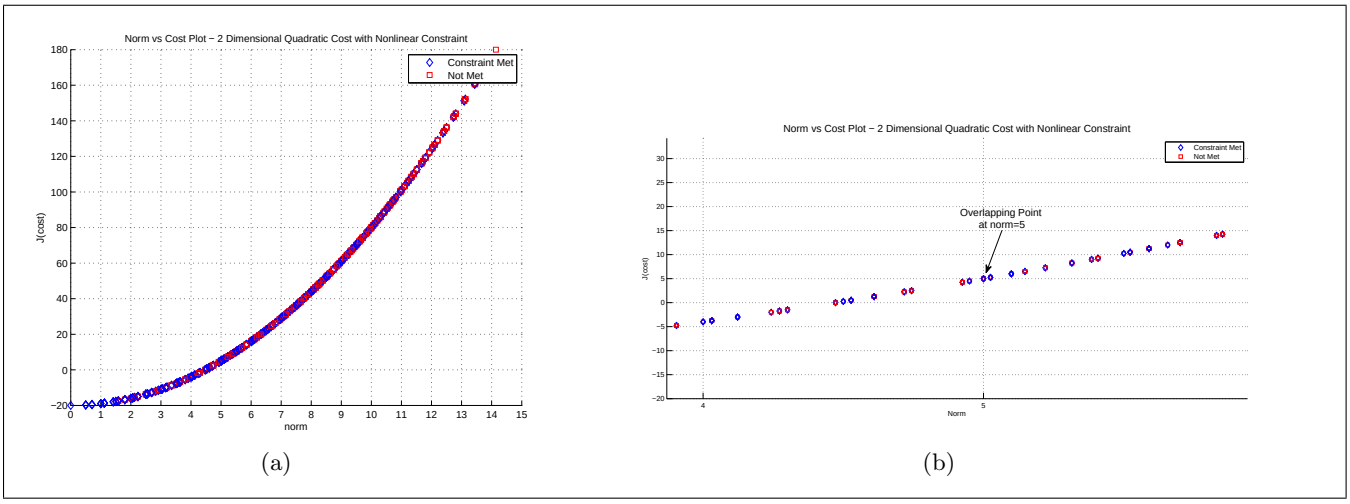


Figure 5.16: 2-Dimensional Quadratic Cost Function with Nonlinear constraint



(a) (b)

Figure 5.17: 2-Dimensional Quadratic Cost Function with Nonlinear Constraint - Cost versus Norm Plot

Table 5.8 displays the results of applying the SQP algorithm to the problem defined in 5.10 with varying initial points across the regions of the function. As was the result in Example 1, the SQP algorithm was also trapped by the "*discontinuities*" created by the constraint function. This is evident from the first 7 rows (with the exception of row 3) and the rows $17-22$. Row 3 and the last 2 rows in Table 5.8 were able to reach the global minimum since they are in regions of constraint violation and thus moved to the closest boundary. As expected (and shown in Figure 5.17(a)), the region surrounding the origin proved not be a problem for the SQP due to constraints not being violated and no local minima being present. This is verified by the SQP optimization results in Table 5.8.

The results obtained from applying the PSO algorithm to the problem defined in 5.10 is shown in Table 5.10. The results clearly show that the PSO algorithm was able to get very close to, if not reach, the global minimum at $x_1, x_2 = 0$. Once again, the time difference between the PSO

Table 5.8: Optimization of 2-Dimensional Nonlinear Constrained Quadratic Function using SQP

x_1 **Initial Value**	x_2 **Initial Value**	x_1, x_2 **Norm Value**	x_1 **Minimum Value**	x_2 **Minimum Value**	**Function Value at Minimum**	**Time Taken by SQP (sec)**
-6	-4	7.2111	-6.25765	1.48E-07	19.15812	0.012583
-5.5	3.5	6.5192	-6.25765	8.56E-08	19.15812	0.027315
-5	-2	5.3852	-6.58E-09	1.64E-08	-20	0.013968
-4.5	0	4.5000	3.032645	0	-10.8031	0.017622
-4	-4	5.6569	3.032645	9.00E-08	-10.8031	0.018073
-3.5	1	3.6401	3.032645	1.43E-08	-10.8031	0.022913
-3	-2.5	3.9051	-6.54E-08	-2.37E-08	-20	0.013797
-2.5	2	3.2016	-2.70E-08	5.57E-08	-20	0.01664
-2	2.5	3.2016	-1.27E-07	-4.44E-08	-20	0.020475
-1.5	3	3.3541	3.96E-08	-3.58E-08	-20	0.022304
-1	0.5	1.1180	5.90E-08	-7.10E-08	-20	0.021717
-0.5	-5	5.0249	2.14E-08	1.26E-09	-20	0.009078
0	-3	3.0000	0	1.14E-08	-20	0.010473
-0.5	5	5.0249	2.14E-08	-1.26E-09	-20	0.016103
0	-3	3.0000	0	1.14E-08	-20	0.010473
0.5	5	5.0249	-2.14E-08	-1.26E-09	-20	0.016299
1	-4	4.1231	1.22E-07	7.51E-10	-20	0.013006
1.5	4	4.2720	-9.11E-09	2.64E-07	-20	0.019676
2	0.5	2.0616	0	2.22E-16	-20	0.013893
2.5	6	6.5000	3.032645	-1.25E-08	-10.8031	0.023815
3	-5	5.8310	3.032645	3.29E-08	-10.8031	0.011589
3.5	-0.5	3.5355	3.032645	3.91E-08	-10.8031	0.016682
4	-5	6.4031	3.032645	3.21E-08	-10.8031	0.015633
4.5	5.5	7.1063	3.032645	1.59E-08	-10.8031	0.023338
5	-6	7.8102	3.032645	-1.74E-07	-10.8031	0.016289
5.5	3	6.2650	-2.79E-09	-1.62E-08	-20	0.019893
6	4	7.2111	2.84E-08	-2.24E-08	-20	0.017394

algorithm and the SQP should be noted since the PSO takes order of seconds longer than the SQP algorithm.

Figure 5.17(b) is a zoomed version of Figure 5.17(a). The non-uniqueness of the norm results in several different points having the same norm as noted with the overlap of constraints being met and not met at the same norm value. It should be noted that in the cost vs. norm plot, Figure 5.17(a), for this example, each norm point does not have multiple different associated costs which is due to the cost function. However, this situation may arise for other cost functions.

In this discussion, the norm value of 5 in Figures 5.17(a) and 5.17(b) is investigated. Figure 5.17(b) shows that at the norm value $\|\mathbf{x}\|_2 = 5$ there is an overlap of points; where the constraint has been met and where the constraint has not been met. In the range plotted, where $-6 \leqslant x_1 \leqslant 6$ and $-6 \leqslant x_2 \leqslant 6$, there are several combinations of the x_1 and x_2 values for which the norm will equal 5. The various combinations are shown in Table 5.9 as well as the constraint function value at each point. Since, the optimization problem is inequality constrained it is clear that (since some constraint function values are positive and some negative) the constraint is both obeyed and violated at the same norm value. This is the reason for the additional *tracking* of points/indices which is required when there are more than one independent variables and the norm is used. The *tracking* functionality would need to keep record of and provide users with the information of which

points are associated to the norm as well as the constraint information i.e. which points violate and which obey the constraints. The benefit of using the norm is that the dimension of the plot is reduced but in the process some information becomes obscure which requires the additional *tracking* of the points. This will not be covered in this study since the focus is on single-input systems, however with additional development this may be achieved. Note should also be taken that by using the norm (with the additional tracking), the dimension of the cost vs. norm plot remains 2 irrespective of the dimension of the optimization problem, which is a great advantage. Additional future work could include a filtering functionality whereby the user can select the *type* of points to be displayed. An example of this could be to display only points at which constraints were adhered to.

Table 5.9: Table Showing x_1 and x_2 values Generating the Same Norm Value

x_1 **Value**	x_2 **Value**	**Norm Value**	**Cost Function Value**
0	-5	5	-1
-3	-4	5	-2.270080073
3	-4	5	0.270080073
-4	-3	5	11.10883992
4	-3	5	-13.10883992
-5	0	5	22.97310687
5	0	5	-24.97310687
-4	3	5	11.10883992
4	3	5	-13.10883992
-3	4	5	-2.270080073
3	4	5	0.270080073
0	5	5	-1

Table 5.10: Optimization of 2-Dimensional Nonlinear Constrained Quadratic Function using PSO

Minimum x_1 Value	**Minimum x_2 Value**	**Function Value at Minimum**	**Time Taken by PSO (sec)**
8.16E-05	-9.01E-05	-19.99999999	8.840743009
-2.25E-05	-5.90E-06	-20	8.112345608
-9.14E-06	-0.000202853	-19.99999996	8.06676082
-5.59E-05	-0.000762453	-19.99999942	7.592120055
-0.000101168	-0.000129514	-19.99999997	7.822146755
0.000207065	-1.37E-05	-19.99999996	7.359671721
1.18E-05	-5.28E-05	-20	7.927088077
-3.10E-06	-9.15E-05	-19.99999999	7.84958611
3.16E-06	9.86E-06	-20	8.604202025
-8.79E-05	0.000224768	-19.99999994	7.357349049
2.07E-05	-7.33E-05	-19.99999999	7.885983375
1.44E-05	0.000129492	-19.99999998	7.794646276
-0.000145445	6.38E-05	-19.99999997	7.607393032
-0.000205135	-1.51E-05	-19.99999996	7.51992479
3.20E-05	-0.00015544	-19.99999997	7.453360922
-7.64E-05	-0.000112722	-19.99999998	8.145302846
-1.54E-06	-0.000124391	-19.99999998	8.10070217
-0.000142513	0.000354107	-19.99999985	7.707456483
3.83E-05	-8.76E-05	-19.99999999	7.501903867
2.14E-05	-5.39E-05	-20	7.870987876

5.2.2.4 Example 3: Nonlinear Equality Constrained Static Optimization of a 2-Dimensional Quadratic Function

This example is the last step before moving on to NMPC and as such the constraints are different to the previous examples. In NMPC the values that the states are able to take on, are constrained by the dynamics of the system. This constraint then influences the objective or cost function, if the objective function is a function of the states. The constraint imposed in this example requires that the values of x_1 and x_2 obey or lie on the constraint function while the cost function remains a 2-dimensional quadratic function.

As mentioned the cost function is once again a 2-Dimensional quadratic function given by:

$$f(x) = x_1^2 + x_2^2 - 20 \tag{5.11}$$

while the constraint in this problem is that the points (x_1 and x_2) should lie ***on*** the constraint function:

$$c(x) : x_1^2 sin(x_1) - x_2 = 0. \tag{5.12}$$

For illustration purposes this was accomplished by keeping x_1 independent and calculating x_2 with

the x_1 value i.e.

$$x_2 = x_1^2 sin(x_1).$$

Thus, the problem translates to:

$$\begin{aligned} \min_{x} \quad & f(x) = x_1^2 + x_2^2 - 20, \\ \text{subject to :} \quad & x_1^2 sin(x_1) - x_2 = 0. \end{aligned} \tag{5.13}$$

The plot of the problem is basically the same as that of constrained optimization Example 2 (Figures 5.15 and 5.16) with the constraint conditions changing. Since only x_1 is the independent variable in the problem, the plot of the cost function value versus x_1 is used to gain insight into the problem. To reiterate, it must be noted that the aim of the methodology is not to search for or ***find*** the minimum (although this may be a by-product). Rather, it is to extract useful information that enables the user to gain insight into the problem and hence empower the individual to make better or more informed choices to get to the solution. As mentioned previously, this may be by means of the choice of optimization algorithm or possibly the initial conditions used.

Figure 5.18 is a plot of the cost function value ($f(x)$) versus the independent variable, x_1, from which insight into the optimization problem is to be gleaned. Taking a look at Figure 5.18, it is noticed that the plot delineates some obvious problems that would be faced by the SQP algorithm. The most apparent or obvious issue that is noticed, are the peaks on the extremes. These peaks create trapping regions which may cause difficulties for the SQP algorithm to overcome. Now, inspecting Figure 5.19 which is the same as Figure 5.18 only zoomed in on the region between -4 and $+4$, it is noticed that there are local minima at $\approx \pm 3.2$. The final region (towards $x = 0$) is where the global minimum is found and is shown to be -20. This is known to be the actual global minimum and solution to the problem. Once again, it is emphasised that while finding the minimum is not the actual goal of the methodology, it is a by-product at times.

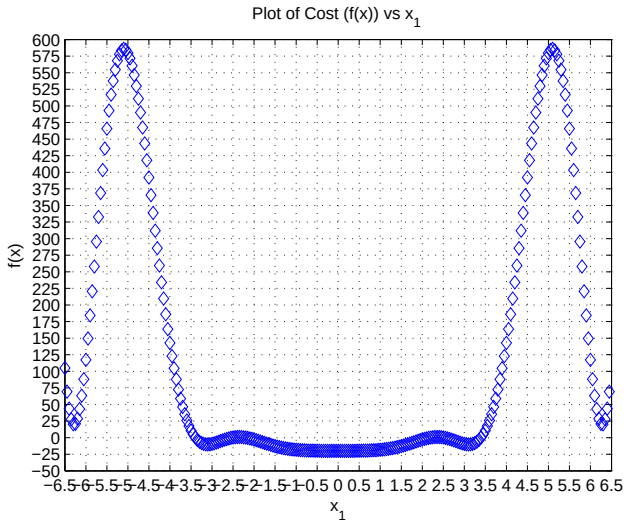


Figure 5.18: 2-Dimensional Quadratic Cost Function with Nonlinear Constraint Function: Points must lie on constraint function

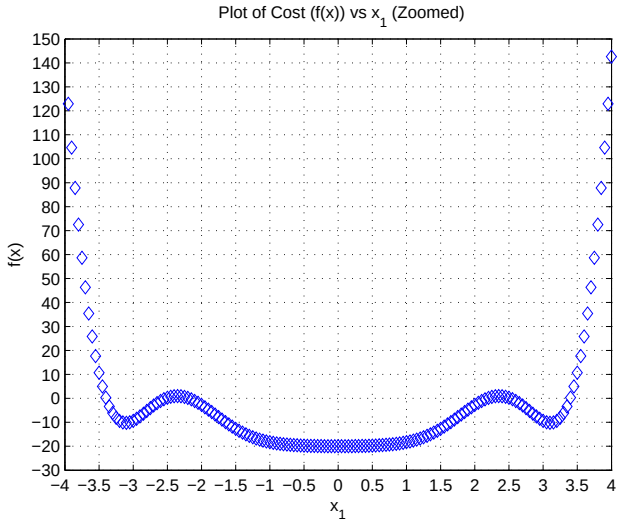


Figure 5.19: 2-Dimensional Quadratic Cost Function with Nonlinear Constraint Function (Zoomed)

Table 5.11 displays the results generated by applying the SQP algorithm to the problem defined by 5.13. Since, SQP is a local optimization algorithm it is expected to be trapped in local minima. Rows 1 and 2 in Table 5.11 clearly affirm the expectation since the SQP algorithm is trapped in the local minimum at $x_1 \approx -6.3$. The next 5 rows where the initial x_1 values range from $-5 \leqslant x_1 \leqslant -3$ (with the exception of the row where $x_1 = -4.5$) result in the SQP algorithm being trapped in the local minimum at $x_1 \approx -3.3$. The region surrounding $x_1 = 0$ is depicted, in Figures 5.18 and 5.19, as the region surrounding the global minimum, which is ratified by the SQP optimization results where the global minimum was reached in Table 5.11. It is noticed that the row where $x_1 = 4.5$ managed to locate the global minimum as was the case where $x_1 = -4.5$. This could be attributed

to the SQP algorithm "*stepping over*" the local minimum by means of a larger step size than the other initial conditions that were trapped in the local minimum.

Table 5.11: Optimization of 2-Dimensional Nonlinear Constrained (On Function) Quadratic Function using SQP

x_1 **Initial Value**	x_2 **Initial Value**	x_1 **Minimum Value**	x_2 **Minimum Value**	**Function Value at Minimum**	**Time Taken by SQP (sec)**
-6	10.05896	-6.27914	0.159464	19.45304	0.01213
-5.5	21.34259	-6.27914	0.159464	19.45304	0.010127
-5	23.97311	3.107493	0.329218	-10.2351	0.012107
-4.5	19.79498	-3.01E-08	-2.72E-23	-20	0.013441
-4	12.10884	-3.10749	-0.32922	-10.2351	0.012996
-3.5	4.297095	-3.10749	-0.32922	-10.2351	0.010818
-3	-1.27008	-3.10749	-0.32922	-10.2351	0.010298
-2.5	-3.74045	-3.10749	-0.32922	-10.2351	0.013095
-2	-3.63719	-2.62E-07	-9.77E-21	-20	0.011522
-1.5	-2.24436	1.28E-08	4.69E-17	-20	0.011635
-1	-0.84147	-1.68E-07	-2.69E-21	-20	0.011802
-0.5	-0.11986	1.92E-08	9.76E-24	-20	0.011225
-1.47E-14	-3.19E-42	-1.47E-14	-3.19E-42	-20	0.005946
0.5	0.119856	-2.44E-08	-1.67E-23	-20	0.011475
1	0.841471	2.73E-07	1.10E-20	-20	0.012435
1.5	2.244364	1.50E-08	-4.76E-17	-20	0.011959
2	3.63719	5.75E-07	9.87E-20	-20	0.012086
2.5	3.740451	3.107493	0.329217	-10.2351	0.013727
3	1.27008	3.107493	0.329218	-10.2351	0.012679
3.5	-4.29709	3.107493	0.329218	-10.2351	0.011334
4	-12.1088	3.107493	0.329218	-10.2351	0.01428
4.5	-19.795	5.90E-08	-1.47E-17	-20	0.014803
5	-23.9731	-3.10749	-0.32922	-10.2351	0.013762
5.5	-21.3426	6.279141	-0.15946	19.45304	0.012046
6	-10.059	6.279141	-0.15946	19.45304	0.019857

The results generated by applying the PSO algorithm to the problem defined in 5.13 are presented in Table 5.12. The results show that the PSO algorithm was able to (at least) get to the vicinity of the global minimum where $f(x) = -20$ with the furthest result being the function value $f(x) = -18.82$. This could possibly be improved by imposing stringent termination conditions and tolerances or increasing the number of particles.

Table 5.12: Optimization of 2-Dimensional Nonlinear Constrained (On Function) Quadratic Function using PSO

Minimum x_1 Value	**Minimum x_2 Value**	**Function Value at Minimum**	**Time Taken by PSO (sec)**
7.65E-01	4.05E-01	-19.25068387	36.60464315
-6.80E-03	6.86E-07	-19.99995377	30.94850084
-5.16E-03	-1.14E-06	-19.99997337	50.53996859
4.85E-01	0.109881772	-19.75238245	29.61096759
-0.006057239	-1.22E-06	-19.99996331	30.77765992
5.61E-06	9.27E-07	-20	34.05040315
-1.50E-03	9.97E-07	-19.99999775	36.9743801
2.19E-01	1.05E-02	-19.95172267	32.40880229
-6.90E-03	-1.33E-06	-19.99995242	31.30027864
1.35E-02	3.46E-06	-19.9998177	30.53929527
5.40E-01	0.149793881	-19.68613504	28.6053666
-0.009356607	1.81E-07	-19.99991245	28.44921459
-0.0044054	9.14E-07	-19.99998059	32.84986359
-8.88E-03	-1.70E-06	-19.9999212	29.12728742
4.62E-01	0.095156686	-19.77747776	28.8533628
4.09E-06	9.65E-07	-20	31.28649526
0.892636008	0.620490869	-18.81819204	27.35407202
7.67E-07	8.77E-07	-20	29.33597898
4.36E-02	8.39E-05	-19.99809813	29.60499274

5.3 NMPC

The previous section showed how the Optimization Roadmap methodology (to graphically obtain insight into the optimization problem) can be applied to constrained and unconstrained static optimization problems. However, NMPC is a dynamic problem in that there is a dynamic model and thus is dependent on time. In this section the methodology will be extended to be applied in an NMPC context (which is the goal of this study) i.e. using the plots to gain insight into the optimization problem.

5.3.1 NMPC Optimization: Optimization Roadmap Methodology

The methodology that was employed for static optimization problems will be extended to the dynamic optimization problems found in NMPC. However, it should be noted that this methodology does not directly deal with the dynamic problem but works with the transformed static optimization problem at every time interval (the transformation is done in the discretization process as described in Section 4.2.1). The last constrained, static optimization example (Example 3 or Section 5.2.2.4) explored the scenario where the cost or objective function to be optimized was constrained and restricted in that the points or values that could be taken on or chosen, needed to satisfy or lie ***on*** the specific constraint function. A simplified NMPC optimization problem can be formulated as:

$$\min_{u} J(\mathbf{x}, u) \tag{5.14}$$

$$\begin{aligned} \text{subject to:} \quad & x_i(k+1) = f_i(x, u), \\ & x \in \mathbb{X}, where\, \mathbb{X} \subset \mathbb{R}^n \\ & u \in \mathbb{U}, where\, \mathbb{U} \subset \mathbb{R} \end{aligned}$$

The simplified NMPC problem defined in 5.14 reveals the constraint (as with constrained static Example 3) that the problem needs to satisfy which include the system state equations. The constraints imposed by the state equations essentially mean that the states x_i are constrained, in that the values that the states can take on are only those that satisfy the state equations. This in turn has a knock on effect onto the objective function ($J(\mathbf{x}, u)$) as the objective function is a function of the states and/or the control input u. Thus, this reveals that the states (x_i) are not independent while the control input (u) is an independent variable.

Extending the idea employed in the constrained static optimization case to NMPC, however, is not a straightforward procedure as there are some important factors that needed to be taken into account such as dealing with the challenge of *dimensionality*.

Below is the procedure for the NMPC optimization roadmap methodology:

1. Begin with initial conditions (one set) for the states.
2. Simulate the system by applying all the input values (within an allowable range) to the system for one time step to obtain the new system state values.
3. Calculate the objective function value (for the current time instant) using the calculated state values.
4. Plot the cost/objective function value versus the independent variable (u).
5. Progress to the next time instant and repeat the procedure using each set of state values, corresponding to the (cost, u) coordinate on the plot, as initial conditions.

The above procedure provides the basic outline of the steps taken to obtain the plots required to provide insight into the system and the optimization problem (Optimization Roadmap). However, it is noticed that when proceeding in Step 5, the number of initial conditions that need to be considered very quickly grows to an unmanageable number. This then renders the method overly cumbersome and impractical, as one would have to deal with many plots.

This *dimensionality* problem can be illustrated by taking a very simple example. If the input (u) is constrained to be $0 \leq u \leq 5$ and an increment of 1 is selected then each u (control input) i.e $u = 0, 1, 2 \ldots 5$ is *applied* to the system to generate the state values for the next time instant corresponding to that input (Step 2). Hence, starting from one initial point, 6 points (which form one plot/curve) are generated by applying the inputs (u's) for the following time instant. Now, all of the inputs (u's) are applied to each of the 6 points in the first time instant which generates $6 * 6$ points (which translates to 6 curves). Following this pattern at every time interval the number of plots increases by 6 times, for this example. In this case, after t time instants 6^t points will be generated so after 5 time intervals $6^5 = 7776$ points and each plot comprise 6 points which would translate into 1296 plots or curves. This clearly illustrates the problem of an "exploding" number of plots which is not a practical solution.

In order to deal with this problem of scale at Step 5, the user is given the option of choosing which points to use as initial conditions from each of the generated plots. Thus, at the presentation of each plot the user has the option of defining the number of points to choose from the plot and then choosing the specific points which will then be used as initial conditions for the next time instant. This provides the user with control over the number of plots that will be generated and hence used for gaining an understanding of the optimization problem.

Modifying the steps in the algorithm above provides the following revised procedure:

1. Begin with initial conditions (one set) for the states.

2. Simulate the system by applying all the input values (within an allowable range) to the system for one time step to obtain the new system state values.

3. Calculate the objective function value (for the current time instant) using the calculated state values.

4. Plot the cost/objective function value versus the independent variable (u).

5. Prompt the user to enter the number of points to use and the selection of points to be used.

6. Progress to the next time instant and repeat (from Step 1) using the set of state values corresponding to each selected point, as initial conditions (for the system simulation).

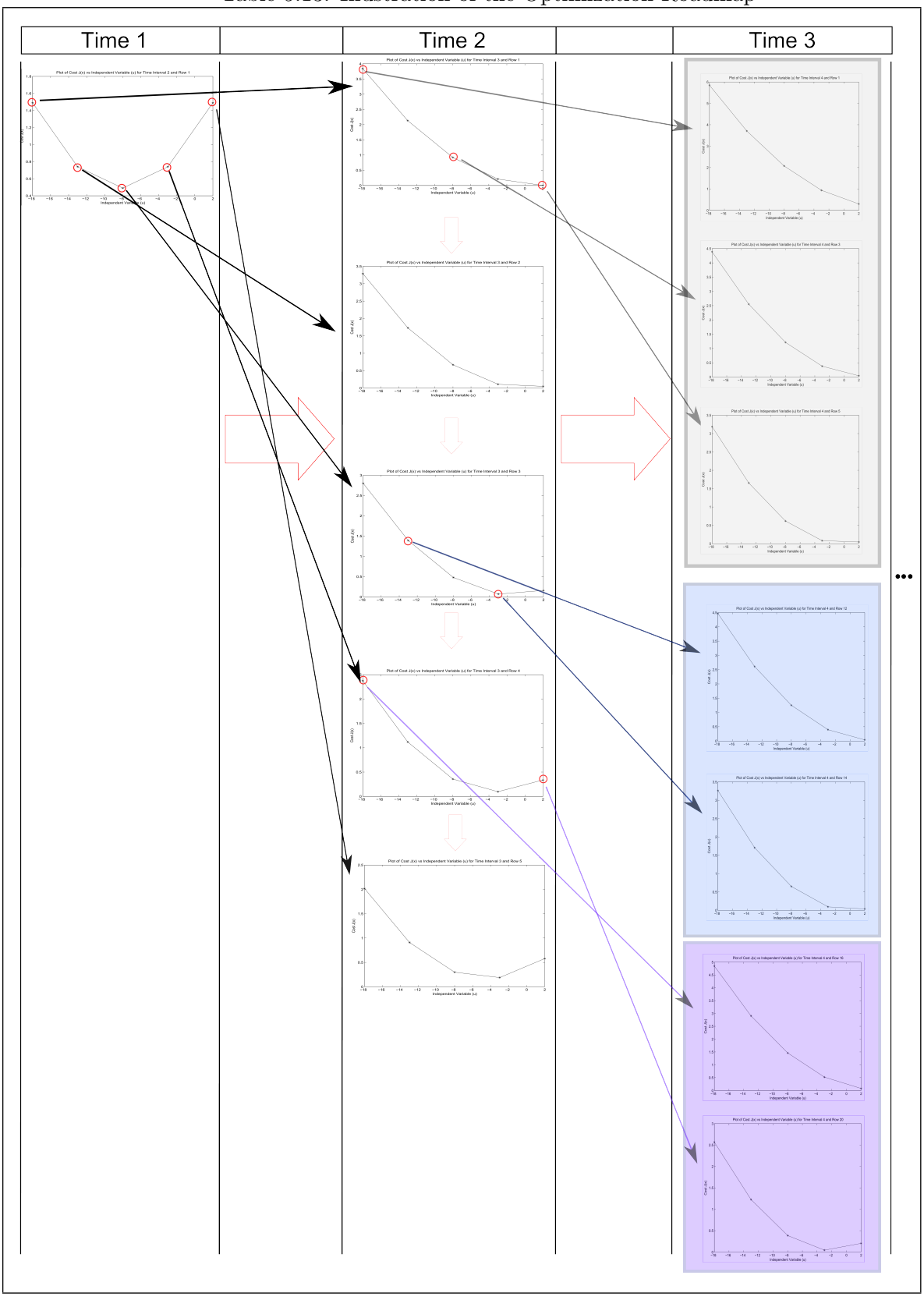
It should noted that the constraints on the control input (u) range are considered (implicitly), in that only feasible values for the control inputs (i.e. within the allowable range) are applied to the system. Secondly, the system dynamics are constraints that need to be adhered to and are taken into account in the Optimization Roadmap by means of simulating the system. Additional constraints (for example on the state values) are taken into account by checking if any constraints are violated and flagging those constraints by a different colour, as done in the static optimization examples. Furthermore, the infeasible points would not be able to be selected for propagation to the next time interval.

Table 5.13 will be used as an illustration for the explanation of the NMPC optimization roadmap methodology. As can be seen at Time 1 the first plot is generated, by applying all the inputs within the allowable input range to the initial conditions chosen for the system, as mentioned in Step 1 and 2 in the procedure above. To propagate to Time 2 in Table 5.13, ALL the points from the plot in Time 1 were chosen as initial conditions. The red circles indicate the points chosen (to be used as initial conditions in generating the plots for the next time interval) and the arrows show the mapping from the selected point to the plot generated by using that point as an initial condition. Thus, since there were five points in the plot at Time 1 there will be five new plots generated for Time 2 which is Step 4 in the procedure above.

Now, for the propagation from Time 2 to Time 3, the user is prompted to choose the number of points to be used from the plot (as they are generated) and then select the points that will be used as the initial conditions for the plots in Time 3. This falls into Step 5 and 6 in the procedure above. The selection of points (in Time 2) denoted by the red circles in the plots show that three points were selected from the first plot, zero from the second, two points from the third and fourth plot and zero points from the fifth which is seven in total. Thus, at Time 3, seven generated plots are expected and are shown in the column for Time 3. The generated plots can be interpreted in the same manner as done for the static optimization examples. The shaded areas in Time 3 illustrate the grouping of the plots in terms of which plot or graph in Time 2 they *originate* from i.e. from

which plot the initial conditions were selected. This process is then repeated until the final time interval.

Table 5.13: Illustration of the Optimization Roadmap



5.3.2 NMPC Example 1: Linear System with a Quadratic Cost Function

In this example the optimization roadmap methodology is applied in an NMPC context. The aim of this example is to illustrate how the methodology is applied but also to demonstrate the value gained by the use of the optimization roadmap. This example will investigate the scenario where the system (which is also the constraint) is linear with two states and the cost or objective function is a quadratic function of the states. Another goal of this example (as well as the other examples) is to investigate the effect of the constraint on the cost or objective function which will be evident from the methodology (the plots).

The cost or objective function is given by:

$$J(x) = x_1^2 + x_2^2 \tag{5.15}$$

while the system equations, which are linear, are given by:

$$\begin{aligned} x_1(k+1) &= 0.3 * x_1(k) + 0.2 * x_2(k) + 0.1 * u(k) \\ x_2(k+1) &= 0.2 * x_1(k) + 0.3 * x_2(k) \end{aligned} \tag{5.16}$$

The system equations provided in 5.16 are, as mentioned previously, actually constraints to the cost function. This is because the states, x_1 and x_2, may only take on values that satisfy the system equations. Hence, they are not independent while the control input, u, may take on any value (within the input range) independently. This is important since the insight methodology uses the plots of the cost function value and the independent variable.

From the above given information, Equations 5.15 and 5.16, which will be the basis for the NMPC problem in this example, it is realised that this problem would fall into the spectrum of linear MPC since the system is linear, there are no nonlinear constraints and the objective function is quadratic.

From this it is expected that the methodology depicts the problem as such i.e. that the cost function is quadratic and hence the optimization should not be a problem for both the SQP and the PSO algorithms. This would be the case as a quadratic function would have only one minimum which is the global minimum and thus the complexity of local minima is not expected to feature in this problem.

The methodology was applied to the NMPC problem with the following parameter values:

1. Initial conditions for x_1 and x_2: $x_1 = 2$ and $x_2 = 1$
2. Range for inputs (u) $[-10 : +5]$
3. Increment between the control input values (independent variable) used in the insight method-

ology is 1.

As alluded to in Section 5.3.1 a challenge faced in the insight approach or methodology is that of *dimensionality*. By dimensionality, in this sense, it is meant that when propagating from one time interval to the next, the number of points used as initial conditions increases by a multiple of the number of input (u) values. In this example the control input (u) range is between -10 and $+5$ in increments of 1, which is 16 data points. Using the aforementioned values for illustration purposes, when propagating from the initial time to the next time interval the number of points to be used as initial conditions changes from 1 initial point to $1 * 16 = 16$ points since each of the 16 inputs (u) are applied to the initial conditions (to generate the plot). The results from applying the inputs to the first initial condition produces new initial conditions. In this manner the following time interval would have $1 * 16 * 16$ or $1 * 16^2 = 256$ initial conditions which, as mentioned before, is already very difficult to manage.

In order to provide some structure to the approach of choosing the points, that will be used as initial conditions in the time interval to follow, an approach to selecting the points that will be propagated is suggested (this approach may be improvised as seen fit). The goal and challenge while selecting the points is that the important information from the graphs is captured or preserved while also reducing the number of points in order to keep the number of graphs manageable and practicable. The approach adopted in order to achieve the aforementioned goals is that after the initial time i.e once applying the inputs to the first initial condition, *all* 16 points (at Time 1) would be chosen as the next set of initial conditions. Following this, as each of the plots are generated and presented, at least 4 points are chosen as the initial conditions for the plot in the next time step (Initial condition points need not be selected from every plot). Once again, the goal is to capture maximum information while keeping the data practically useful, so the four points that will be selected will need to encapsulate the important aspects of the plot. In this way the points that will be selected are:

1. The point corresponding to the minimum input (u) value
2. The point corresponding to the maximum input (u) value
3. The point corresponding to the average input (u) value
4. The point corresponding to the minimum cost value

It should be noted that the points on the curves have the coordinates of cost value and control input value. However, when selecting the point from the curve the corresponding state values are used as the initial conditions for the next time interval.

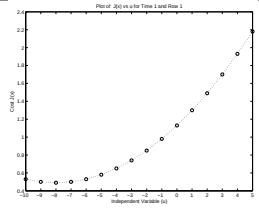
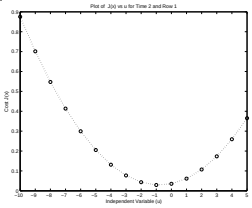
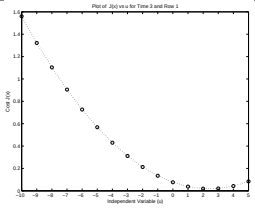
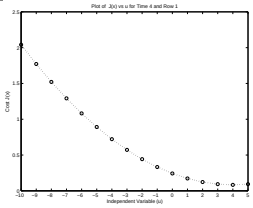
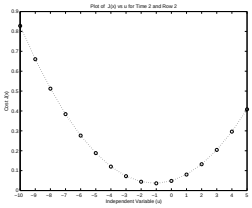
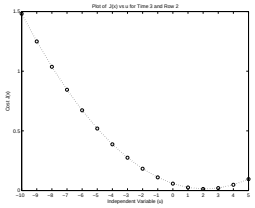
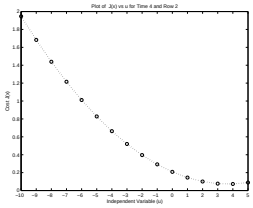
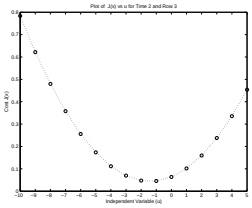
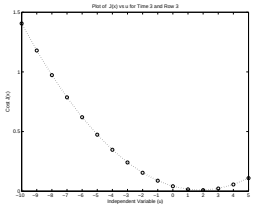
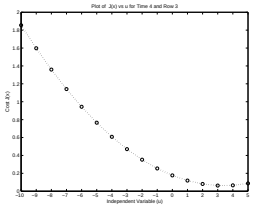
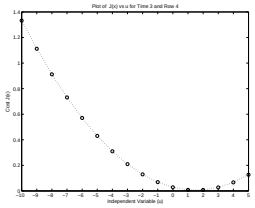
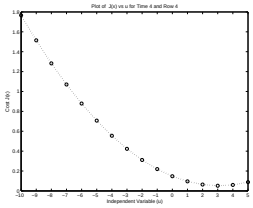
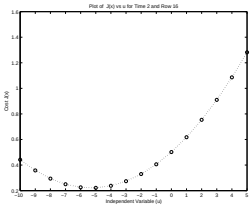
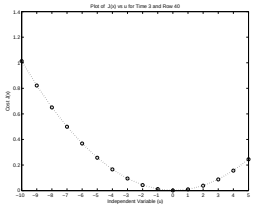
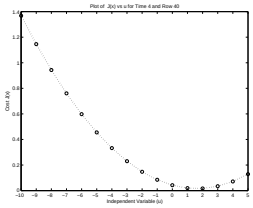
Table 5.14 summarises the optimization roadmap methodology for this example and for the sake of brevity not all of the curves have been included in the table. At Time 1 there is one plot generated since at the beginning of the process there is just one initial condition and by applying all

the control inputs (within the acceptable range) to the system, the plot at Time 1 is generated. The approach mentioned previously describes that *all* the points from the first curve (Time 1) are chosen which in this example means that all 16 points from the curve in the first column are to be used as initial conditions for the next time interval. Therefore, in the second column or Time 2 there are 16 curves since each initial condition point generates a curve.

In order to limit the number of curves that the methodology produces, the user may decide on a manageable or practical number of curves to work with. This is separate from the first selection where all the points from the first curve are selected (as per the approach described earlier). Thus, when the user is presented with the various generated curves the total number of points that are selected as initial conditions (and hence the number of curves generated in the next time interval) from all of the curves, in that time interval, are limited to that user defined number. The user will select points from the curves presented in Time 2 which will generate the curves for the next time interval (Time 3). Once again (at Time 3), the user will be presented with the curves and will choose the number of points and select the initial condition point values (by using the approach outlined above) from the curves which will then propagate to the next time interval. Since the number of curves from Time 3 onwards are bound by the maximum number chosen by the user, it will be noticed that the number of curves from Time 3 onwards in Table 5.14 are the same. The methodology may require some user discretion since it is not necessary to select points from all of the generated curves and not all of the points in the guideline need to be selected from every generated curve.

The plots in Table 5.14 reveal important information about the optimization problem that is to be solved in the NMPC algorithm. Inspecting the curves it is clear that the quadratic nature of the objective function is maintained. This was expected, as mentioned earlier, since the system which constrains the objective function is linear. Although these results from the plots were expected, the value or insight gained is that the optimization problem has a quadratic and convex nature. This also reveals that there are no local minima and in terms of the optimization algorithms, a local optimization algorithm such as SQP should manage to obtain the optimum values without any difficulty.

Table 5.14: Table Depicting the Optimization Roadmap Methodology in Time Intervals for NMPC Example 1

Time 1	Time 2	Time 3	Time 4	...
				...
				...
				...
	⋮			...
		⋮	⋮	...
				...

Tables 5.15 and 5.16 display the results of the application of the NMPC algorithm with the SQP and PSO optimization algorithms, respectively. The first column is the time interval which indicates that this example was executed for 10 time intervals. The second column is the control input u with respect to which the algorithm is optimizing. Column 3 and 4 are the state values; it will

be noticed that the initial conditions $[2, 1]$ are in the first row (the first time interval). The last column provides the computation time for that time interval which will be used for comparison between the two optimization algorithms.

Due to the nature of this problem, the expectation from the problem definition as well as the insight provided by the methodology summarised in Table 5.14, the results from the NMPC algorithm with either the SQP or PSO algorithms are expected to be the same. The results Tables 5.15 and 5.16 show that the control input values obtained are very closely aligned which accedes with the expectation. The column containing the computation time highlights a point that is of particular significance in NMPC. Comparing the computation time of the two algorithms, once again the SQP has exceptionally better performance than the PSO. Although the computation time of the PSO algorithm may be improved by reducing the number of particles, the number of particles used in the PSO algorithm are kept constant for all the examples as are all the other parameters, for comparison purposes.

Table 5.15: NMPC Example 1: NMPC Algorithm Results using SQP with Initial $u = -10$ and Horizon $N = 2$

Time Interval	**Input u(k)**	**State x_1**	**State x_2**	**Computation Time (sec)**
0	-8	2	1	2.04
1	-1.4	0	0.7	0.028
2	-0.42	0	0.21	0.012
3	-0.126	0	0.063	0.012
4	-0.0378	0	0.0189	0.011
5	-0.01134	0	0.00567	0.011
6	-0.003402	0	0.001701	0.011
7	-0.001021	0	0.00051	0.011
8	-0.000306	0	0.000153	0.011
9	-0.000092	0	0.000046	0.011

Table 5.16: NMPC Example 1: NMPC Algorithm Results using PSO with Initial $u = -10$, $N = 2$

Time Interval	**Input u(k)**	**State x_1**	**State x_2**	**Computation Time (sec)**
0	-8	2	1	15.75
1	-1.399798	0	0.7	13.768
2	-0.420061	0.00002	0.21	14.916
3	-0.126008	0	0.063004	14.094
4	-0.037807	0	0.018901	14.206
5	-0.01134	0	0.00567	14.606
6	-0.003397	0	0.001701	14.659
7	-0.001154	0	0.00051	13.973
8	-0.000376	-0.000013	0.000153	14.397
9	-0.000099	-0.000011	0.000043	13.754

The plots in Figures 5.20 and 5.21 provide an additional and important tool. The figures provide

a plot of the cost or objective function value against the time interval which provides a useful comparison of the performance of the optimization algorithms in terms of minimizing the cost. The figures provide a means to compare the results at a time interval level as well as total cost. In this example the results display that in terms of total cost and cost per time interval the PSO and SQP algorithms performed equally.

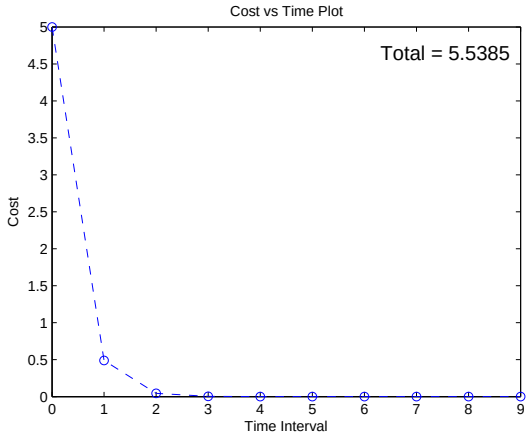


Figure 5.20: Example 1: Cost vs. Time Plot for SQP with Initial $u = -10$, $N = 2$

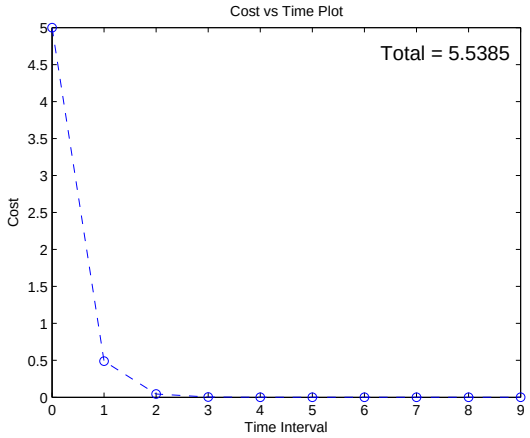


Figure 5.21: Example 1: Cost vs. Time Plot for PSO with Initial $u = -10$ and Horizon $N = 2$

The horizon length (N), which is the prediction and control horizon in this implementation, is an important parameter in NMPC and is chosen on a per application basis. The results obtained in this example were obtained using a horizon length of $N = 2$. Increasing the horizon length may lead to the NMPC algorithm calculating the value for the control input that is not exactly the global minimum, but remains within the *region* of the global minimum, in order to improve the control of the system. However, the increase in horizon length is coupled with the drawback of an increase in computational expense. As an illustration, Table 5.15 which was obtained using a horizon length of $N = 2$ shows that for the first time interval the optimal control input (which was applied) is

$u^* = -8$ while for the results for $N = 5$, the value of ($u^* = -8.44$) was obtained. Appendix B contains the simulation data for the horizon length of $N = 5$. Looking at the corresponding plot in Table 5.14 (Column 1), it is noticed that the values obtained are in the region of the global minimum. Hence, comparing the total cost values obtained for the two horizon lengths ($N = 2$ and $N = 5$), it is noticed that the increase in horizon length **in this example** has a negligible effect on the total cost, with a 0.04% decrease.

5.3.3 NMPC Example 2: Nonlinear System with a Quadratic Cost Function

As was done in Example 1, the optimization roadmap insight methodology will be applied in an NMPC context. This example explores the scenario of a quadratic cost function with a nonlinear system. The aim of this example is to investigate whether the nonlinear system (constraint) affects the quadratic cost function and hence the optimization problem. In other words, this is to highlight whether having a quadratic cost function necessarily safeguards the algorithm from exposure to a non-convex optimisation problem which, as previously highlighted, is one of the biggest challenges in NMPC. A further aim is to investigate whether the insight methodology captures and elucidates the nature of the optimization problem.

The objective function is given by:

$$J(x) = x_1^2 + x_2^2 \tag{5.17}$$

which is constrained by the system equations shown in equation 5.18. Naturally, x_1 and x_2 must satisfy the system equations which, in other words, means that they are not independent. Once again, the independent variable is the control input u. The system equations are highly nonlinear and are given by:

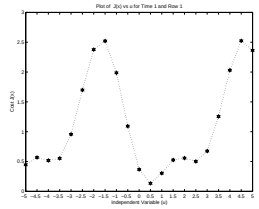
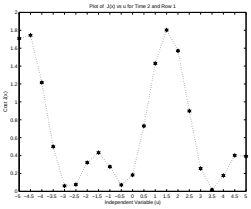
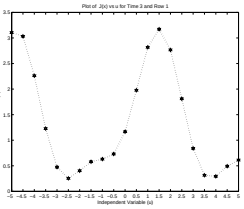
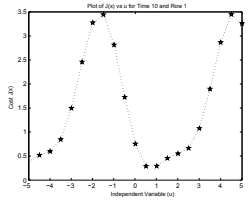
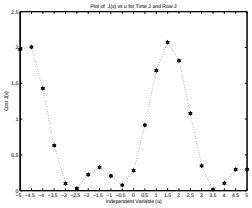
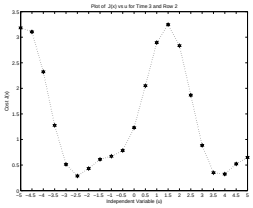
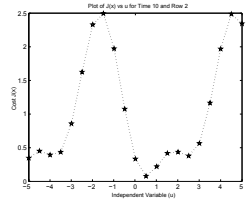
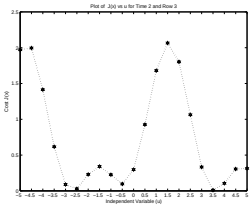
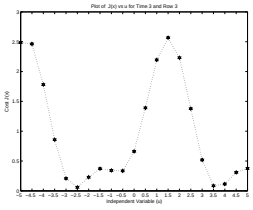
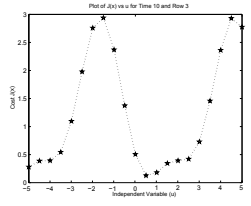
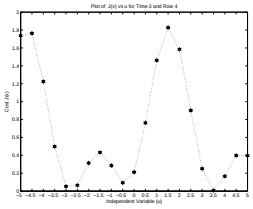
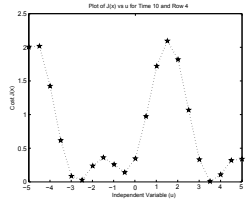
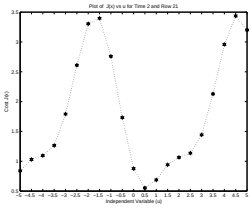
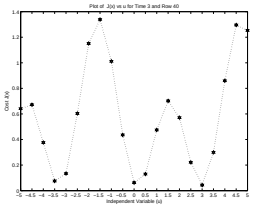
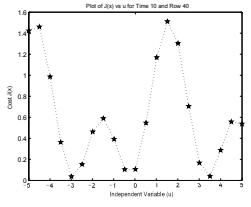
$$\begin{aligned} x_1(k+1) &= 0.3 * (x_2(k))^3 - sin(x_1(k)) * x_1(k))^4 + 0.2 * cos(u(k)) \\ x_2(k+1) &= 0.5 * x_2(k) - x_1(k)^3 + sin(u(k)) \end{aligned} \tag{5.18}$$

One of the additional benefits of the Optimization Roadmap is to enable one to identify problem areas that could cause a local optimizer (such as SQP) problems of getting trapped in local minima. Further value that could be gleaned from the methodology is whether SQP or PSO would be the best algorithm choice in a scenario such as this; as different scenarios have differing requirements. Hence, the different algorithms may lend themselves more readily or be more effective in certain situations over others.

The plots, which have been generated using the methodology, are presented in Table 5.17. These plot were generated by using the following parameter settings:

1. Initial conditions for x_1 and x_2 are $x_1 = 1$ and $x_2 = 1$
2. Range for inputs (u) $[-5 : +5]$
3. Increment between the control input values (independent variable) used in the insight methodology is 0.5.

Table 5.17: Table Depicting the Optimization Roadmap Methodology in Time Intervals for NMPC Example 2

Time 1	Time 2	Time 3	...	Final Time
			...	
			...	
			...	
			...	
			...	
			...	

Examining the plots in Table 5.17, the first most striking characteristic of the plot is that it is not quadratic and has a very nonlinear characteristic. This is important and useful information as

this confirms the expectation that using a quadratic cost or objective function does not avoid the difficulty of a non-convex optimisation problem in NMPC. Hence, it is noticed that the nonlinear constraints (the system equations) significantly affect the structure/shape of the cost function.

The plots from Table 5.17, of the methodology, exhibit that there are regions that could be regarded as problem areas in terms of local minima and trapping regions. In the plot in Time 1 of Table 5.17, it is noticed that in the range of $[-5 : -1.5]$ and $[1 : 5]$ it would be expected that the SQP algorithm would get trapped at the local minima present. Thus, in this case, if it is required that the global minimum should be obtained at every time interval, one would recommend that a global optimisation technique or in this example PSO, since it has a possibility of escaping local minima, be used. As can be seen from the structure of the plots in Table 5.17, PSO (or another global optimisation algorithm) would be more well suited to this problem (in searching for global optima) than SQP, since there are several peaks and minima which the SQP algorithm would not be able to overcome.

Further useful information that could be extracted from the plots are the selection of the initial conditions for the optimization algorithm. Once again, looking at Time 1 in Table 5.17 it would be expected that if the initial guess or starting position is in the region of $[-1.5 : 1]$ then the SQP algorithm will be able to reach the global minimum. As mentioned previously and as will be noticed in the results, the SQP algorithm is fast compared to the PSO and thus would be the algorithm of choice in systems wherein it is required that the optimization computation be expeditious.

Tables 5.18 and 5.19 display the results obtained from the NMPC algorithm using the SQP optimization algorithm. In Table 5.18 the initial condition or starting point for the input was set at $+5$ (the upper bound). Once again looking at the plot in Time 1 of Table 5.17 it is noticed that the interval $+4 \leq u \leq +5$ is an area that may trap a local optimizer. This leads to the expectation that the SQP algorithm would get *trapped* in that *trapping region*. Now, analysis of the optimization results in Table 5.18 exhibits that the SQP optimization algorithm was trapped in the *trapping region* of $u = +5$, not only for the first time interval but for the entire run of the NMPC algorithm. This would clearly produce suboptimal results as will be confirmed with the cost versus time plot.

Table 5.19 displays the results from the NMPC using the SQP algorithm where the initial condition or starting point for the input was at the lower bound $u = -5$. Analysing the plots in Table 5.17 it is noticed that at $u = -5$ there is also a *trapping region*. The SQP algorithm being a local optimization method, is expected to generate results that are local optimizers and (as already seen) get trapped in local minima. Looking at the optimization results in Table 5.19, it is noticed that the SQP algorithm was unable to escape the *trapping region* and ended up at $u = -5$ for all time intervals.

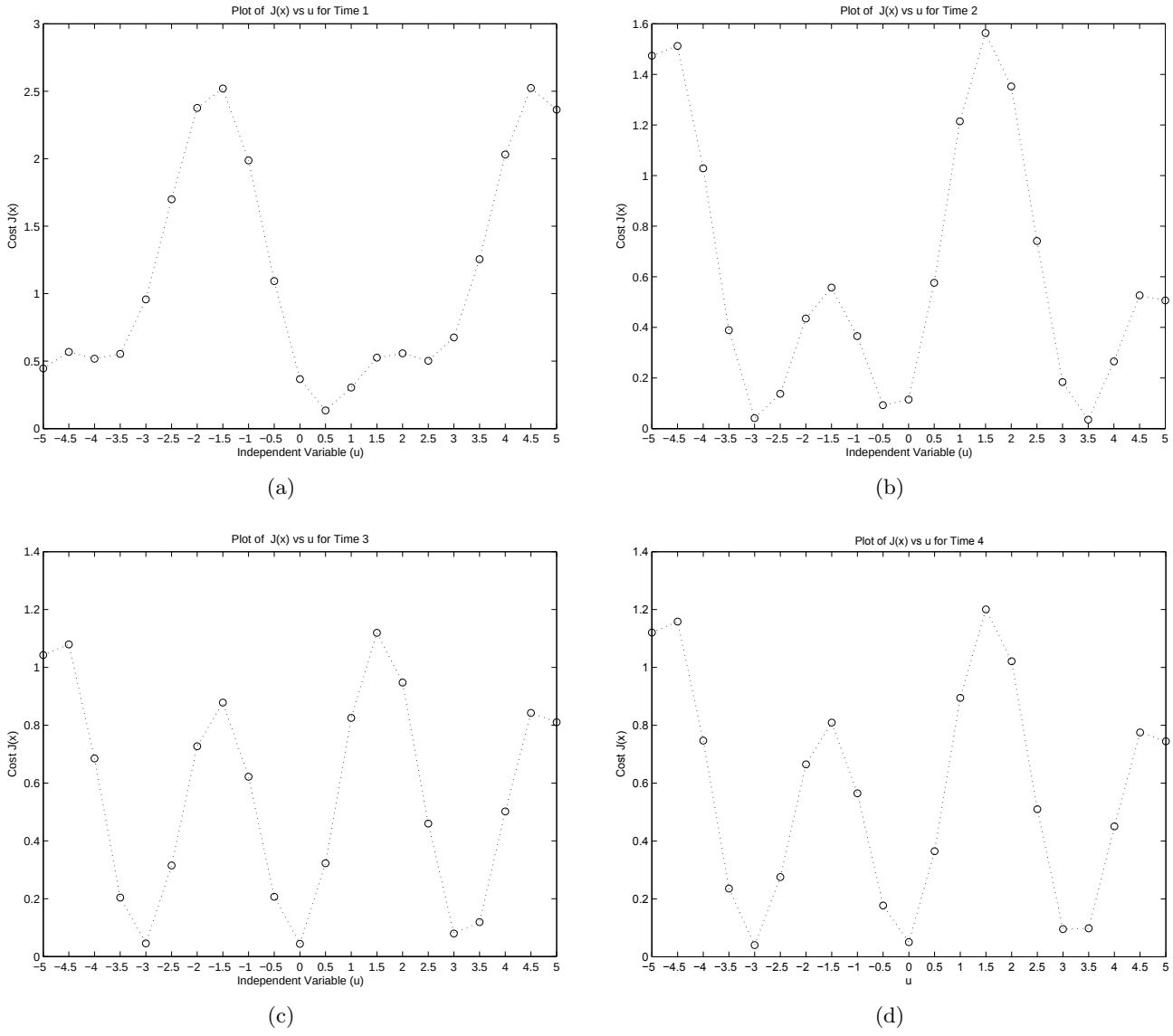


Figure 5.22: NMPC Example 2: Plots showing the Global Minimum Path for 4 Time Intervals

Table 5.18: NMPC Algorithm Results using SQP with Initial $u = 5$, $N = 2$

Time Interval	**Input u(k)**	**State x_1**	**State x_2**	**Computation Time (sec)**
0	5	1	1	0.414
1	5	-0.484739	-1.458924	0.025
2	5	-0.849119	-1.574487	0.007
3	5	-0.72397	-1.133951	0.006
4	5	-0.198731	-1.146443	0.006
5	5	-0.395001	-1.524297	0.007
6	5	-0.996402	-1.659442	0.008
7	5	-0.486668	-0.799399	0.006
8	5	-0.070287	-1.243359	0.007
9	5	-0.519914	-1.580256	0.007

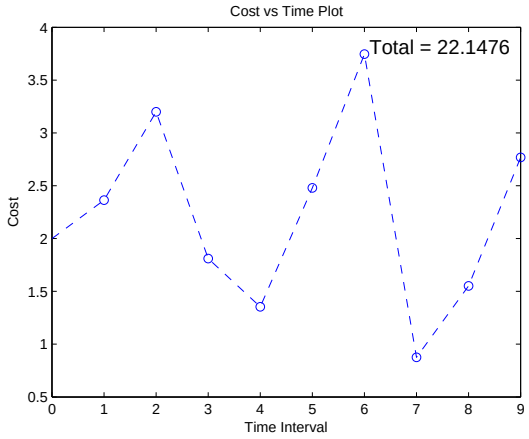


Figure 5.23: Cost vs. Time Plot for SQP u stuck at $+5$, $N = 2$

Table 5.19: NMPC Algorithm Results using SQP with Initial $u = -5$, $N = 2$

Time Interval	**Input u(k)**	**State x_1**	**State x_2**	**Computation Time (sec)**
0	-5	1	1	0.427
1	-5	-0.484739	0.458924	0.025
2	-5	0.111456	1.302286	0.008
3	-5	0.719299	1.608683	0.007
4	-5	1.129275	1.391107	0.006
5	-5	-0.60599	0.214357	0.007
6	-5	0.136497	1.288637	0.006
7	-5	0.698652	1.6007	0.008
8	-5	1.133902	1.418251	0.007
9	-5	-0.585283	0.210155	0.007

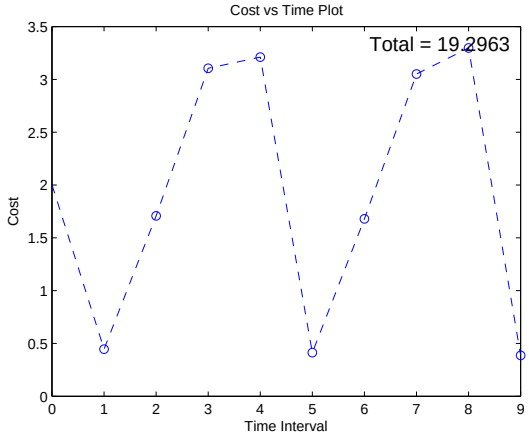


Figure 5.24: Cost vs. Time Plot for SQP with u stuck at -5, $N = 2$

Table 5.20 displays the results generated by NMPC using the PSO algorithm with the initial or starting point being $u = -5$. As already discussed earlier this point is the lower bound of the input range and is also in a *trapping region* within which the SQP algorithm was trapped when using $u = -5$ as a starting point. Analysing the results generated by PSO and Table 5.17, it is noticed that the algorithm has escaped the trapping region and has reached a point which is very close to the global minimum.

Table 5.20: NMPC Algorithm Results PSO with Initial $u = -5$, $N = 2$

Time Interval	**Input u(k)**	**State x_1**	**State x_2**	**Computation Time (sec)**
0	0.48028	1	1	15.823
1	-3.111275	-0.364098	-0.037972	15.645
2	-3.134634	-0.193666	-0.001032	15.756
3	3.149781	-0.199724	-0.000211	16.537
4	3.149715	-0.199678	-0.000327	15.189
5	3.149731	-0.199678	-0.000325	15.022
6	3.14971	-0.199678	-0.000339	16.868
7	3.149716	-0.199678	-0.000325	14.365
8	-3.133443	-0.199678	-0.000324	16.038
9	-3.133482	-0.199678	-0.00035	14.911

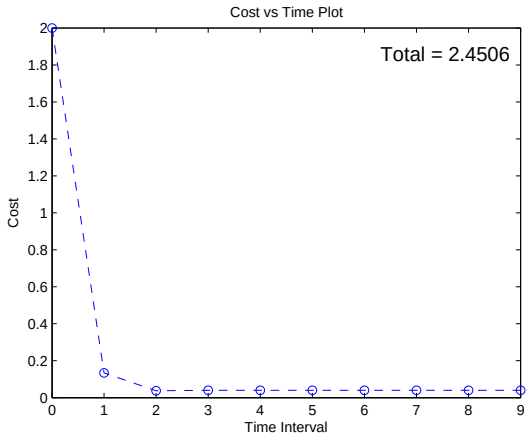


Figure 5.25: Cost vs. Time Plot for PSO with Initial $u = -5$, $N = 2$

Table 5.21 displays the results generated by NMPC using the PSO algorithm with initial condition or starting point for the input being set to $u = 5$. The results show that the PSO optimization algorithm once again managed to escape from the *trapping region* at $u = 5$, and located a solution that is within close proximity to the global minimum as was the case when starting at $u = -5$.

Figure 5.22 shows the *global minimum path* for the first four time intervals. This is the path selecting the global minimum from the plots along the *path* i.e. by selecting a point from the first plot the

next plot along the selected path is fixed from which the global minimum is selected. Comparing the plots with the results generated by the PSO algorithm in Tables 5.20 and 5.21, the plots show that the PSO algorithm has selected the global minima.

Table 5.21: NMPC Algorithm Results PSO with Initial $u = 5$, $N = 2$

Time Interval	**Input u(k)**	**State x_1**	**State x_2**	**Computation Time(sec)**
0	0.480316	1	1	16.97
1	-3.111108	-0.364101	-0.03794	15.202
2	-3.134591	-0.193665	-0.001181	15.644
3	-3.133465	-0.199724	-0.000329	15.322
4	-3.133469	-0.199678	-0.000325	18.747
5	-3.133469	-0.199678	-0.000325	17.24
6	3.149702	-0.199678	-0.000325	15.198
7	-3.133463	-0.199678	-0.00031	15.178
8	-3.133456	-0.199678	-0.000323	17.5
9	-3.133467	-0.199678	-0.000337	16.021

Figures 5.24, 5.23, 5.25 and 5.26 are plots of the cost or objective function value versus the time for the various applications of the NMPC algorithm (with the two optimization algorithms and the differing initial conditions). The objective of this tool was briefly highlighted earlier but, as can be verified by inspection of the plots, it is easy to identify performance problems of the optimization algorithm in the NMPC. Comparing figures 5.23, 5.24 and 5.25 the disparity in the performance (in terms of cost) of the SQP algorithm versus the PSO is clearly discernible. The total cost values for the SQP (when trapped) are significantly higher than those obtained when using the PSO algorithm.

Table 5.22 and Figure 5.26 show the results of the SQP algorithm with initial conditions at $u = 2$. The results display a marked improvement in the total cost being within 12% of the PSO total cost value. For illustration purposes, looking at the plot in Time 1 of Table 5.17 it is evident that one of the reasons for the SQP algorithm total cost being larger than the PSO value was due to it being trapped at the local minimum at ≈ 2.5. However, looking at the time taken to obtain the results, the SQP is appreciably faster than the PSO which offers a significant advantage.

Table 5.22: NMPC Algorithm Results SQP with Initial $u = 2$, $N = 2$

Time Interval	**Input u(k)**	**State x_1**	**State x_2**	**Computation Time (sec)**
1	2.489319	1	1	0.408
2	3.552599	-0.700412	0.106995	0.062
3	3.14035	-0.027859	-0.002428	0.014
4	3.149951	-0.2	0.00005	0.013
5	3.149711	-0.199675	-0.000334	0.013
6	3.149716	-0.199678	-0.000324	0.014
7	3.149716	-0.199678	-0.000324	0.013
8	3.149716	-0.199678	-0.000324	0.013
9	3.149716	-0.199678	-0.000324	0.013
10	3.149716	-0.199678	-0.000324	0.013

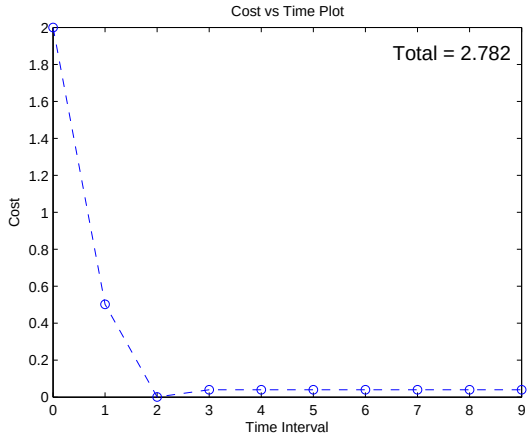


Figure 5.26: Cost vs. Time Plot for SQP with Initial $u = 2$, $N = 2$

Table 5.23: NMPC Algorithm Results using SQP with Initial Point for $u = 1.2$, $N = 2$

Time Interval	**Input u(k)**	**State x_1**	**State x_2**	**Computation Time (sec)**
0	0.480316	1	1	1.266
1	-3.111108	-0.364101	-0.03794	0.167
2	-0.006952	-0.193665	-0.001181	0.014
3	-3.150105	0.200266	-0.000279	0.014
4	-0.008551	-0.200313	0.000341	0.013
5	-3.150145	0.200313	-0.000343	0.015
6	-0.008552	-0.200313	0.000343	0.013
7	-3.150145	0.200313	-0.000343	0.016
8	-0.008552	-0.200313	0.000343	0.013
9	-3.150145	0.200313	-0.000343	0.014

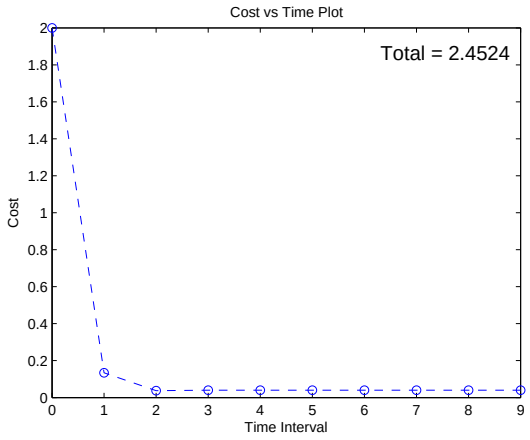


Figure 5.27: Cost vs. Time Plot for SQP with Initial $u = 1.2$, $N = 2$

In order to further improve the performance of the SQP algorithm in this example, the initial conditions for the optimization (SQP) were changed to $u = 1.2$. The results are displayed in Table 5.23 and Figure 5.27 which shows that the total cost decreased from the results obtained when the initial conditions for the SQP algorithm were $u = 2$ and is approximately equal to that obtained by the PSO algorithm (0.07% difference). These results once again highlight that the insight provided by the plots in Table 5.17 facilitates the selection of the initial conditions used in the algorithm, which significantly improves the results.

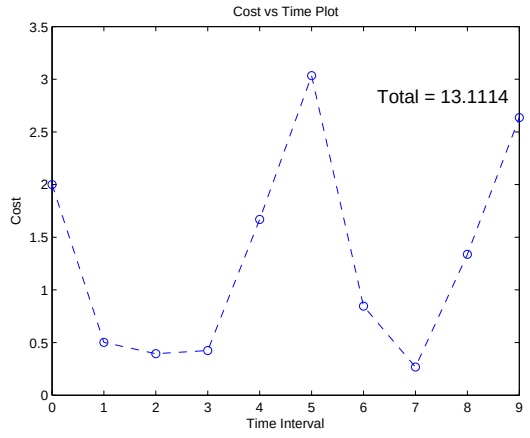


Figure 5.28: Cost vs. Time Plot for SQP with Initial $u = -5$, $N = 5$

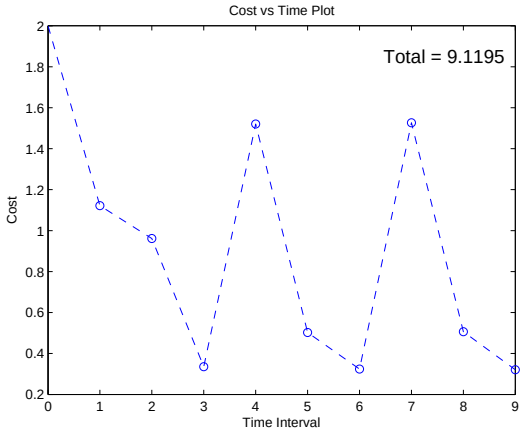


Figure 5.29: Cost vs. Time Plot for SQP with Initial $u = 5$, $N = 5$

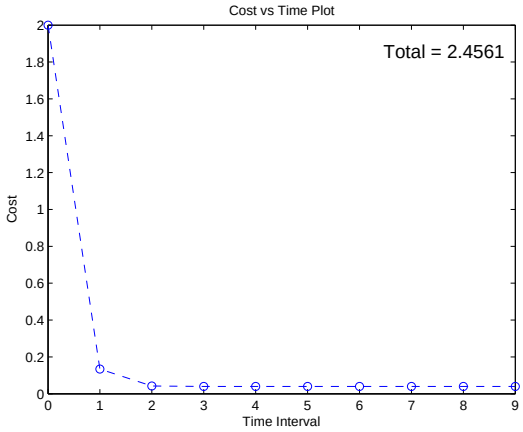


Figure 5.30: Cost vs. Time Plot for SQP with Initial $u = 1.2$, $N = 5$

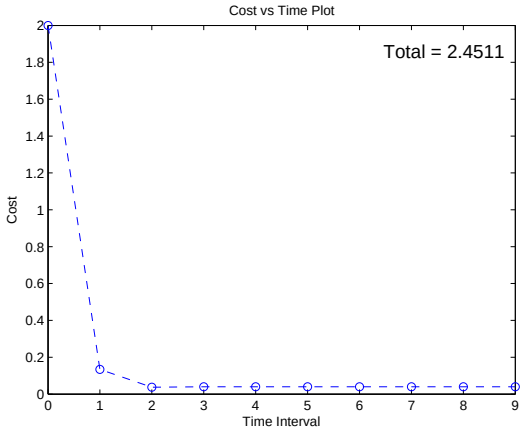


Figure 5.31: Cost vs. Time Plot for PSO with Initial $u = 5$, $N = 5$

In this example the horizon length has a substantial effect on the control and total cost of the system. Figures 5.28, 5.29, 5.30 and 5.31 are the cost vs. time plots obtained for the various initial conditions, with the horizon length of $N = 5$. When performing a comparison of the results in the cost vs. time plots for the corresponding initial conditions for example (Figure 5.23 and Figure 5.29, Figure 5.24 and Figure 5.28), it is noted that the performance of the NMPC algorithm is significantly improved. The cost difference between Figure 5.23, with a horizon length of 2, and Figure 5.29, with a horizon length of 5, shows almost a 60% decrease. However, when inspecting the results from the NMPC using the PSO algorithm it is noticed that there is still a significant disparity in the total cost value (even for horizon length 2).

Using the insight gained from the Optimization Roadmap the initial conditions for the SQP algorithm were adjusted to $u = 1.2$. This produced the results shown in Table 5.24 and the cost vs. time plot in Figure 5.30. Similarly to the scenario with a horizon length of 2 the results (value of total cost) significantly decreased to within 0.02% of the total cost value obtained by PSO with the advantage of the speed of the SQP algorithm.

Table 5.24: NMPC Algorithm Results using SQP with Initial $u = 1.2$, $N = 5$

Time Interval	**Input u(k)**	**State x_1**	**State x_2**	**Computation Time (sec)**
0	0.478853	1	1	1.083
1	-0.030053	-0.363966	-0.039238	0.067
2	-3.151661	0.206139	-0.001453	0.043
3	-0.008854	-0.200359	0.000582	0.039
4	0.008474	0.200313	-0.00052	0.039
5	0.008031	0.199672	0.000176	0.04
6	0.008041	0.199678	0.000159	0.04
7	0.008041	0.199678	0.000159	0.039
8	0.008041	0.199678	0.000159	0.04
9	0.008041	0.199678	0.000159	0.04

5.3.3.1 NMPC Example 2 with Additional Constraint

The investigation to follow includes an additional constraint to the problem defined in Section 5.3.3. As previously mentioned, the problem is already constrained by the input range and the dynamics of the system (the states). The additional constraint that is added to the problem is that:

$$x_1^2 + x_2^2 > 0.25. \tag{5.19}$$

Effectively, this constraint enforces a lower bound on the cost function. This constraint is used to illustrate how the Optimization Roadmap may be used to understand the effect of constraints on the optimization in NMPC.

Similarly to the static examples, the points on the Optimization Roadmap plots that do not adhere to the constraints are shown in red. In addition to this, the user will not be able to select infeasible points from the plots in the Optimization Roadmap methodology to propagate to the next time interval.

Table 5.25 displays an overview of the Optimization Roadmap including the additional constraint. As can be seen, from the Optimization Roadmap plot in Time 1, the global minimum point is now an infeasible point due to the imposed constraint. Thus, it is expected that this constraint will not *allow* the optimization algorithm to reach the global minimum. To illustrate this, the same initial conditions used previously when the SQP optimization algorithm was able to reach the global minimum in Time interval 1 are used. The initial control input was set to $u = 1.2$ and horizon length to $N = 2$ to illustrate the effect. Table 5.26 shows the results from the NMPC algorithm using the SQP optimization algorithm. The control input calculated in the first time interval is not the global minimum obtained previously which was at $u = 0.48$. Thus, the imposed constraint has clearly altered the problem in the sense of where the feasible optimal solutions lie.

Figure 5.32 shows the cost versus time plot for the current investigation. As is evident by comparison to Figure 5.27, the constraint has increased the total cost value. This is a further confirmation of the insight provided by just the one plot from the Optimization Roadmap. By further inspection of the plots within Table 5.25 it is noticed that in some of the plots such as those in Time 4 are not affected by the constraint. In this manner the Optimization Roadmap can be used to gain further insight into the optimization problem and the effect of the constraints.

Table 5.25: Table Depicting the Optimization Roadmap Methodology in Time Intervals for NMPC Example 2 with an Additional Constraint

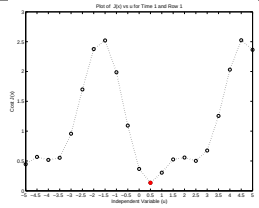
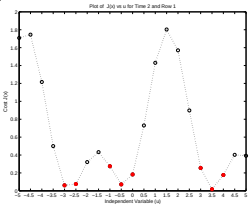
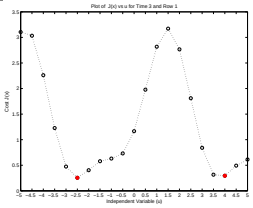
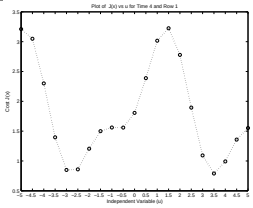
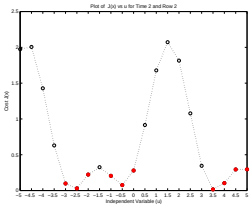
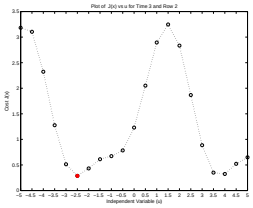
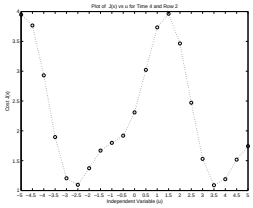
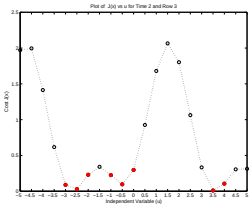
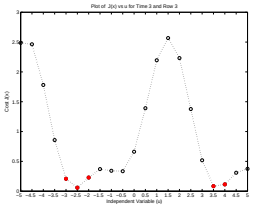
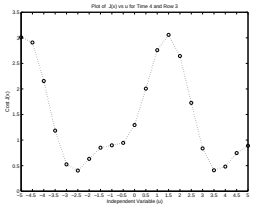
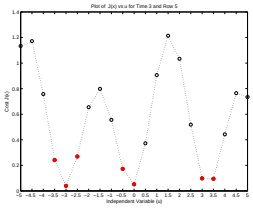
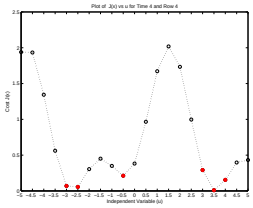
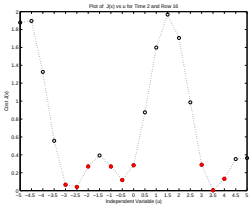
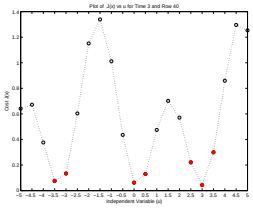
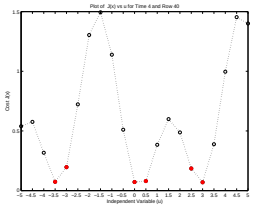
Time 1	Time 2	Time 3	Time 4	...
				...
				...
				...
	⋮			...
		⋮	⋮	...
				...

Table 5.26: NMPC Example 2: NMPC Algorithm Results using SQP Optimization with Additional Constraint with Initial $u = 1.2$, $N = 2$

Time Interval	**Input u(k)**	**State x_1**	**State x_2**	**Computation Time (sec)**
0	0.891082	1	1	4.123
1	0.244249	-0.415757	0.277752	0.059
2	0.233472	0.212559	0.452569	0.013
3	0.237744	0.221951	0.448037	0.013
4	0.237215	0.220822	0.448595	0.013
5	0.23728	0.220961	0.448527	0.013
6	0.237272	0.220944	0.448535	0.013
7	0.237273	0.220946	0.448534	0.013
8	0.237273	0.220945	0.448534	0.013
9	0.237273	0.220946	0.448534	0.013

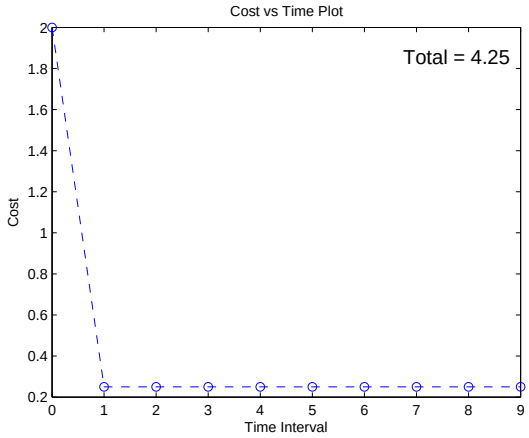


Figure 5.32: Cost vs. Time Plot for SQP with Additional Constraint with Initial $u = 1.2$, $N = 2$

5.3.4 NMPC Example 3: Van der Pol System with a Quadratic Cost Function

In this section the well known (forced) Van der Pol oscillator will be used as the system constraint. The Van der Pol oscillator was selected as the system to be investigated due to various regions of operation which display diverse dynamic characteristics. These regions will be explored in the following examples by means of the insight methodology and the NMPC algorithm.

In the examples investigated in the following sections the cost or objective function used is (as in previous examples) a quadratic function of the two states x_1 and x_2. The cost function is given by:

$$J(x) = x_1^2 + x_2^2. \tag{5.20}$$

The Van der Pol Oscillator system is given by:

$$\dot{x_1} = x_2, \tag{5.21}$$

$$\dot{x_2} = -x_1 + \mu(1 - x_1^2)x_2 + u \tag{5.22}$$

where μ is a scalar parameter and u is the control input. When $\mu = 0$ the Van der Pol system reduces to a simple harmonic oscillator.

5.3.4.1 Van der Pol System with Asymptotically Stable Dynamics

This example aims to explore the region in which the Van der Pol oscillator displays asymptotically stable dynamic characteristics as shown in the phase portrait in Figure 5.33. The parameter values used in this example for the asymptotically stable dynamic characteristics are:

- Initial conditions for x_1 and x_2 are $x_1 = 0.2$ and $x_2 = 0.5$
- $\mu = -2$
- Input range $-0.9 \leqslant u \leqslant 0.9$
- Increment between the control input values (independent variable) used in the insight methodology is 0.2.
- Time intervals for ODE $T = 3s$ (This is different from the integration step size which is a variable step size and is determined by the ODE solver).

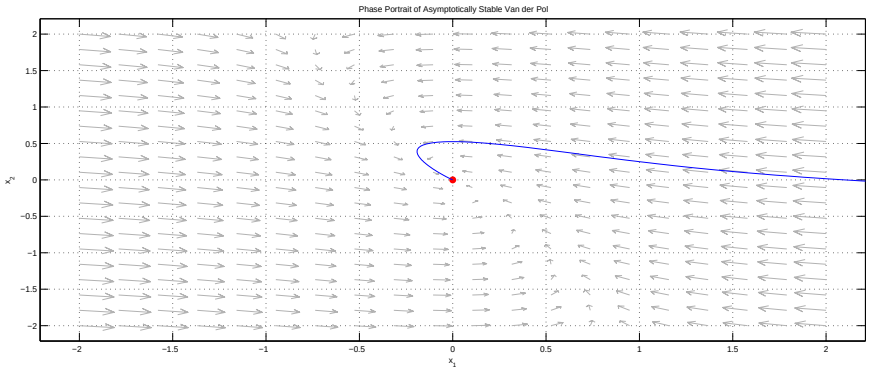


Figure 5.33: Phase Portrait For Asymptotically Stable Van der Pol Oscillator

Table 5.27 displays some of the plots generated from the optimization roadmap methodology. Inspecting the plots in Table 5.27, it is clear that the optimization problem to be solved by the

optimization algorithm, in the NMPC algorithm, is convex. Hence, the local optimization algorithm (SQP) should find the minimum, which is the global minimum due to the convexity of the function. The insight provided by the plots thus prove valuable since knowledge of the type of optimization problem that the optimization algorithm is required to solve enables the user to make informed decisions such as, in this example, the type of optimization algorithm that would be most suited for this problem. Additionally, the insight provided informs the user that the choice of initial conditions will not affect the algorithm's ability to locate the global minimum value. This will be tested and verified.

Inspecting the plot in column one of Table 5.27 it is noticed that the global minimum is approximately located at the point $u = -0.1$. The results obtained by the SQP algorithm for this example are shown in Table 5.28. The optimized value for the control input u calculated by the SQP algorithm validates the value indicated by the plot. To validate that the initial conditions do not change the optimal values calculated by the SQP algorithm, Table 5.30 displays the results with the initial $u = -0.5$ which are the same. Thus, the PSO algorithm does not present any benefit in terms of locating the optima. The results displayed in Table 5.29 were obtained with the PSO algorithm which align exactly with those obtained by the SQP algorithm. The inordinate time taken by the PSO algorithm is attributed to the function evaluations using the ODE solver. This situation highlights the optimization algorithm that is most suited to this scenario which was revealed by the optimization roadmap by means of illustrating the convexity characteristic.

Table 5.27: Table Summarising the Optimization Roadmap Methodology for Asymptotically Stable Van der Pol Example with $\mu = -2$, State Initial Condition $= [0.2, 0.5]$, $T = 3$

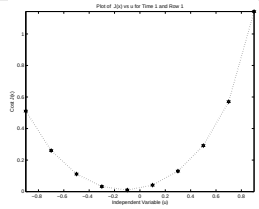
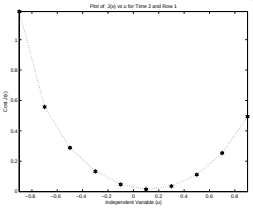
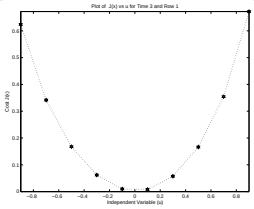
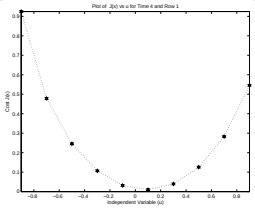
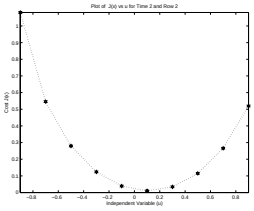
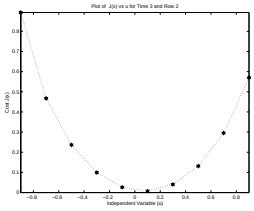
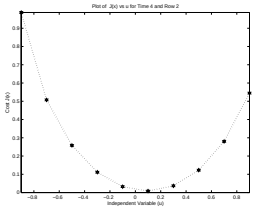
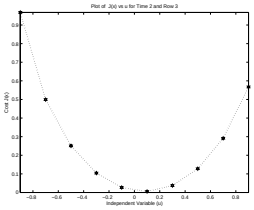
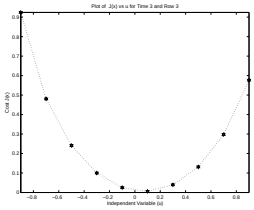
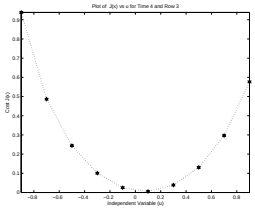
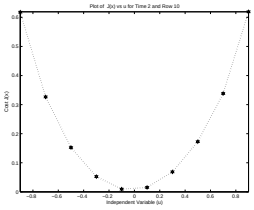
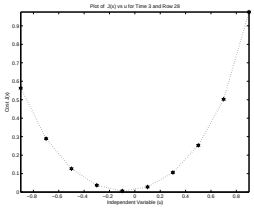
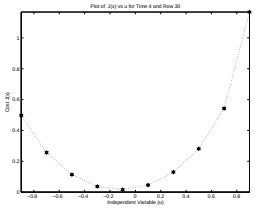
Time 1	Time 2	Time 3	Time 4	...
				...
				...
				...
	⋮	⋮	⋮	...
		⋮	⋮	...
				...

Table 5.28: NMPC with SQP Optimization Algorithm with Asymptotically Stable Van der Pol System with Initial $u = -0.9$, $N = 2$

Time Interval	**Input u(k)**	**State x_1**	**State x_2**	**Computation Time (sec)**
0	-0.11665	0.2	0.5	1.203
1	0.011798	0.016925	-0.096917	0.496
2	-0.00106	-0.001657	0.008885	0.426
3	0.000095	0.000148	-0.000796	0.471
4	-0.000009	-0.000013	0.000071	0.471
5	+0.000001	+0.000001	-0.000006	0.466
6	-0.00000	-0.00000	+0.000001	0.467
7	-0.00000	-0.00000	-0.00000	0.465
8	-0.00000	-0.00000	+0.00000	0.465
9	-0.00000	-0.00000	-0.00000	0.466

Table 5.29: NMPC with PSO Optimization Algorithm with Asymptotically Stable Van der Pol System with Initial $u = -0.9$, $N = 2$

Time Interval	**Input u(k)**	**State x_1**	**State x_2**	**Computation Time (sec)**
0	-0.11665	0.2	0.5	1226.453
1	0.011796	0.016925	-0.096917	1083.076
2	-0.001058	-0.001658	0.008885	1058.251
3	0.000095	0.000149	-0.000795	1187.361
4	-0.000009	-0.000013	0.000071	1378.293
5	-0.000003	0.000001	-0.000006	1226.314
6	0.000001	-0.000004	0	1039.354
7	0	0	0.000001	1009.737
8	0	0	0	1064.219
9	0	0	0	1076.384

Table 5.30: NMPC with SQP Optimization Algorithm with Asymptotically Stable Van der Pol System with Initial $u = -0.5$, $N = 2$

Time Interval	**Input u(k)**	**State x_1**	**State x_2**	**Computation Time (sec)**
0	-0.11665	0.2	0.5	0.809
1	0.011798	0.016925	-0.096917	0.205
2	-0.00106	-0.001656	0.008886	0.19
3	0.000095	0.000149	-0.000796	0.19
4	-0.000008	-0.000013	0.000071	0.188
5	0.000001	0.000001	-0.000006	0.19
6	+0.000000	+0.000000	0.000001	0.189
7	+0.000000	+0.000000	-0.000000	0.196
8	+0.000000	+0.000000	+0.000000	0.19
9	+0.000000	+0.000000	-0.000000	0.189

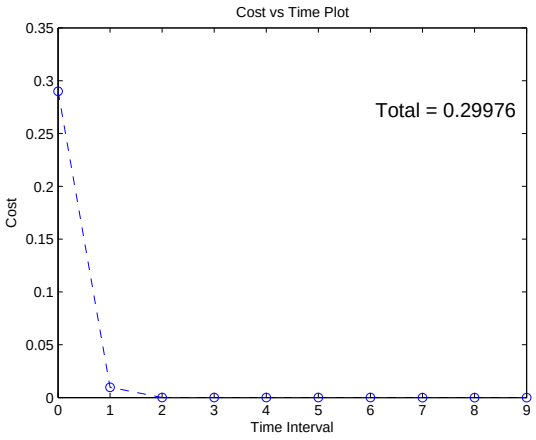


Figure 5.34: Cost vs. Time Plot for Van der Pol SQP with Initial $u = -0.9$, $N = 2$

Similarly to NMPC Example 1, increasing the horizon length had a negligible effect on the optimal control input values obtained as well as the total cost. The NMPC results and the cost vs. time plots for $N = 5$ are available in Appendix B.

5.3.4.2 Van der Pol System with Limit Cycle Characteristics

The aim in this example is to show how the dynamic characteristics of the Van der Pol system change by manipulating the μ constant and initial conditions of the states, x_1 and x_2, thereby affecting the optimization problem, in the NMPC. As mentioned, the cost function will remain as quadratic, as in the previous example, in order to highlight the effect of the system on the NMPC algorithm and more specifically the optimization problem.

This example will explore the scenario of the limit cycle behaviour of the system. Figure 5.35 shows the phase portrait of the forced Van der Pol system (5.21) with $u = 1$ showing the limit cycle of the system and Figure 5.36 displays the simulation of the states over time. The parameter values used in this example for the limit cycle dynamic characteristics are:

- Initial conditions for x_1 and x_2 are $x_1 = 1$ and $x_2 = 1$
- $\mu = 1$
- Input range $-1 \leqslant u \leqslant 1$
- Increment between the control input values (independent variable) used in the insight methodology is 0.2.
- Time intervals for ODE $T = 3s$ [4] (This is different to the integration step size which is a variable step size which is determined by the ODE solver).

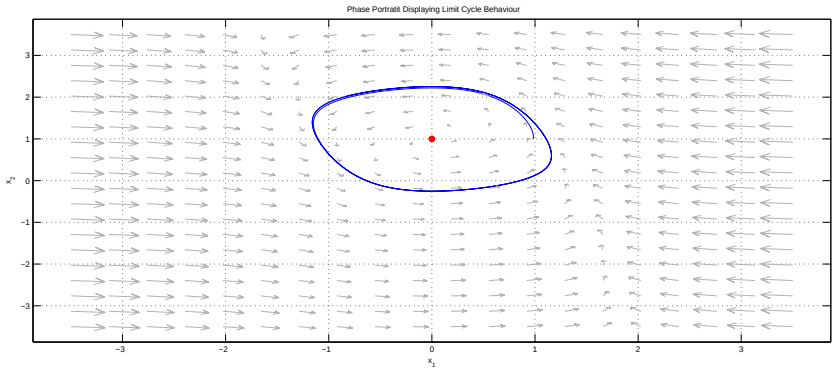


Figure 5.35: Phase Portrait For Van der Pol Oscillator - Limit Cycle

Table 5.31 shows the plots generated from the optimization roadmap. In contrast to the plots generated in the previous example which displayed convex quadratic plots, the plots in Table 5.31

[4]The end time for the ODE solver was chosen as 3s in order to enhance the nonlinear effect of the system on the cost function and hence the optimization problem.

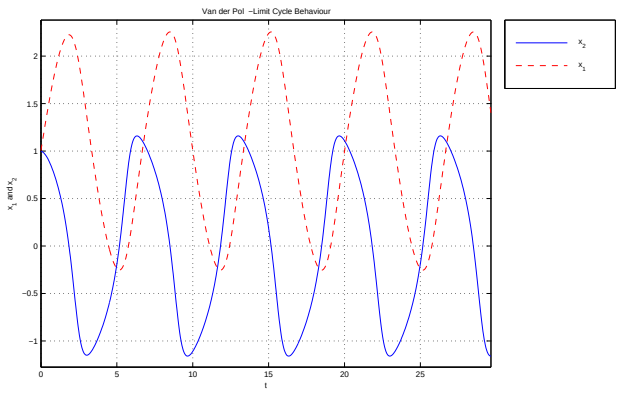


Figure 5.36: Simulation For Van der Pol Oscillator - Limit Cycle

show a non-convex structure. As was the case in Example 2, in Section 5.3.3, the system constraints transform the optimization into a non-convex optimization problem and this is clearly revealed by the optimization roadmap.

Inspecting the plot in column 1 of Table 5.31 reveals that there is a local minimum region which may cause the SQP algorithm to get trapped. Table 5.32 shows the SQP optimization for the Van der Pol system with limit cycle dynamic characteristic with the initial $u = -1$. Once again, inspecting the plot in Time 1 of Table 5.31 and looking at the first row in Table 5.32 reveals that the SQP algorithm was trapped in a local minimum and did not reach the global minimum. Figure 5.37 shows the cost vs. time plot for these results.

Table 5.35 and Figure 5.40 show the NMPC results and the cost vs. time plot for the case where the horizon length $N = 5$ and initial $u = -1$. Looking at the cost vs. time plot it is clear that the total cost value decreased significantly when compared to the case with the horizon length $N = 2$, in Figure 5.37. This is due to the increase in horizon length which improves the performance of the NMPC controller. On the other hand, the increase in horizon length has the drawback of being more computationally expensive. Figure 5.41 shows the cost versus time plot using the initial condition $u = 0.2$ which, as can be seen, increased the total cost compared to when initial $u = -1$ (Figure 5.40).

For comparison purposes, the results of the NMPC algorithm with the PSO optimization algorithm (shown in Table 5.33 and Figure 5.38 for horizon length $N = 2$) are viewed. Comparing these results to those obtained by the SQP algorithm, which were discussed earlier, shows that the PSO algorithm provided a lower total cost. Now, as was done in Example 2, more suitable initial conditions were selected (using the insight gained by the optimization roadmap) for the SQP algorithm to investigate whether the total cost can be improved.

Table 5.34 and Figure 5.39 display the results obtained from the NMPC algorithm using the SQP optimization algorithm with horizon length $N = 2$ and initial $u = 0.5$. The cost vs. time plot shows a definite improvement to the results obtained with the initial conditions $u = -1$. Examining the PSO results indicate that the total cost value now obtained by the SQP is equal to that obtained by the PSO algorithm.

Similarly, with the case where $N = 5$, the results obtained from the SQP algorithm, with initial conditions selected after using the insight provided by the optimization, the roadmap shows a substantial decrease in the total cost. The initial conditions were selected to be $u = 0.7$ with the cost versus time plot shown in Figure 5.42.

Table 5.31: Table Depicting the Optimization Roadmap Methodology in Time Intervals for NMPC Example with Van der Pol System with Limit Cycle and $\mu = 1$ State Initial Condition $[1, 1]$ and $T = 3$

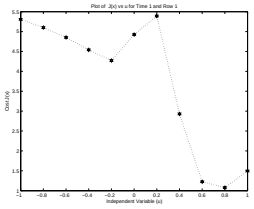
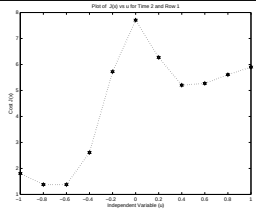
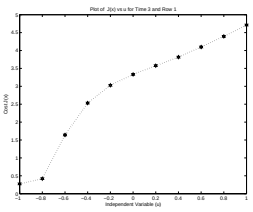
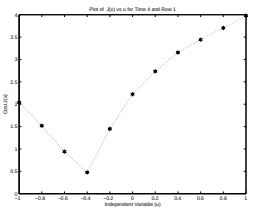
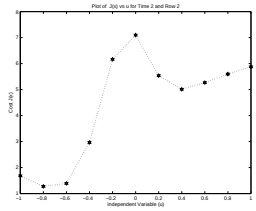
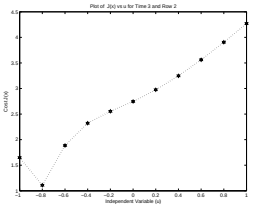
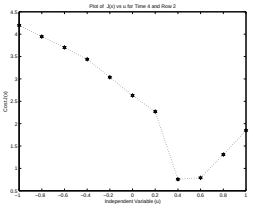
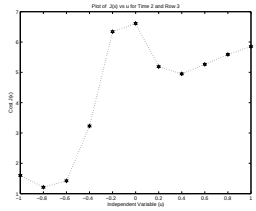
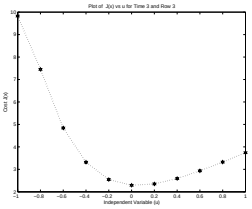
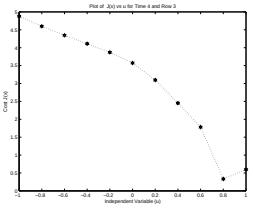
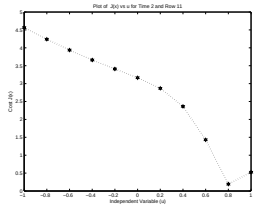
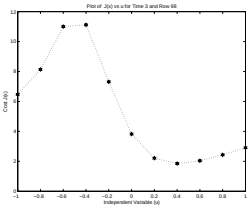
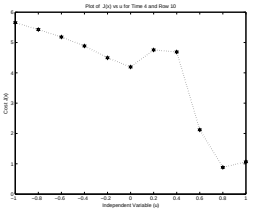
Time 1	Time 2	Time 3	Time 4	...
				...
				...
				...
	⋮	⋮	⋮	...
		⋮	⋮	...
				...

Table 5.32: NMPC SQP Optimization of Quadratic Cost Function with Van der Pol (Limit Cycle behaviour) as constraint and Initial $u = -1$, $N = 2$

Time Interval	**Input u(k)**	**State x_1**	**State x_2**	**Computation Time (sec)**
0	-0.198335	1	1	1.147
1	-0.724207	-1.974836	-0.607725	0.468
2	-0.146565	-0.780191	0.771368	0.541
3	0.272683	1.125306	-1.053251	0.349
4	-0.383449	-1.217142	1.069391	0.338
5	0.320031	1.200703	-1.107493	0.336
6	-0.381668	-1.224561	1.081107	0.35
7	0.326323	1.20458	-1.105643	0.298
8	-0.376678	-1.223324	1.083235	0.349
9	0.330889	1.206695	-1.10339	0.292

Table 5.33: NMPC Algorithm Results using PSO algorithm with Van der Pol System (Limit Cycle behaviour) as the constraint and Initial $u = -1$, $N = 2$

Time Interval	**Input u(k)**	**State x_1**	**State x_2**	**Computation Time (sec)**
0	0.723247	1	1	1084.54
1	0.103089	0.69465	-0.734704	1196.262
2	-0.255583	-1.102847	1.044099	1171.625
3	0.386803	1.21566	-1.064962	1209.75
4	-0.315405	-1.198299	1.10936	1304.374
5	0.386216	1.22572	-1.079357	1304.114
6	-0.322202	-1.202627	1.107723	1371.001
7	0.380431	1.224353	-1.081768	1289.376
8	-0.327514	-1.205144	1.105072	1197.353
9	0.375606	1.223021	-1.083665	1233.908

Table 5.34: NMPC Algorithm Results using SQP optimization with Van der Pol System (Limit Cycle behaviour) as the constraint and Initial $u = 0.5$, $N = 2$

Time Interval	**Input u(k)**	**State x_1**	**State x_2**	**Computation Time (sec)**
0	0.723247	1	1	1.894
1	0.103091	0.69465	-0.734703	0.395
2	-0.255583	-1.102845	1.044101	0.329
3	0.386816	1.215657	-1.064964	0.306
4	-0.3154	-1.198286	1.109374	0.321
5	0.386172	1.225707	-1.079369	0.308
6	-0.322268	-1.202646	1.107703	0.312
7	0.380309	1.224321	-1.081808	0.308
8	-0.32762	-1.205199	1.105013	0.271
9	0.375525	1.223	-1.083696	0.308

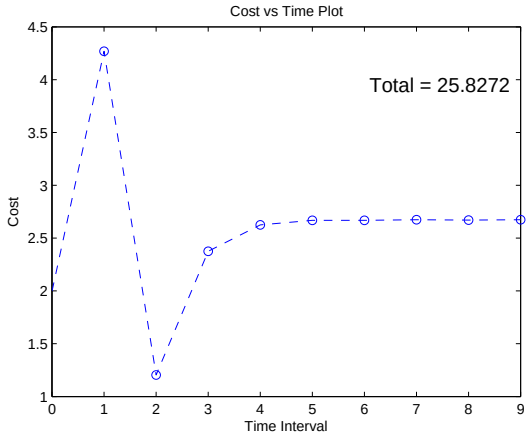


Figure 5.37: Cost vs. Time Plot for Van der Pol SQP with Initial $u = -1$

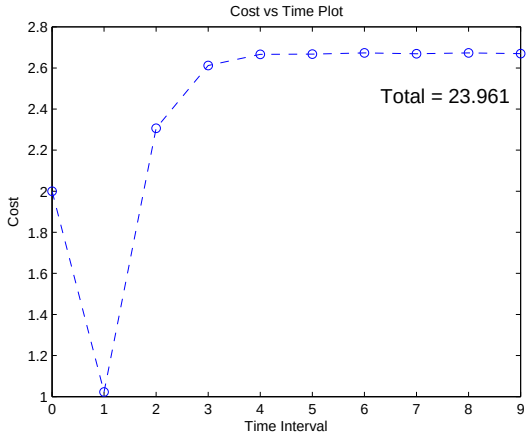


Figure 5.38: Cost vs. Time Plot for Van der Pol PSO

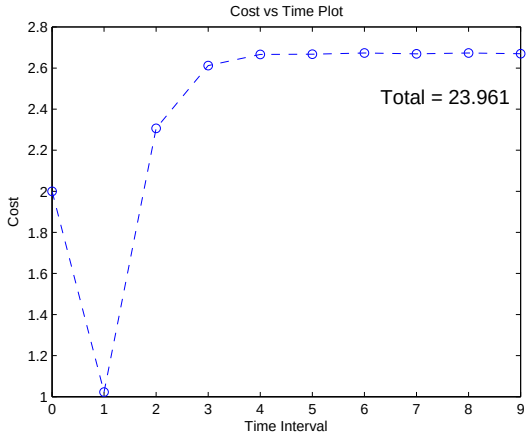


Figure 5.39: Cost vs. Time Plot for Van der Pol SQP with Initial $u = 0.5$

Table 5.35: NMPC Algorithm Results using SQP optimization with Van der Pol System (Limit Cycle behaviour) as the constraint and Initial $u = -1$, $N = 5$

Time Interval	**Input u(k)**	**State x_1**	**State x_2**	**Computation Time (sec)**
0	0.662469	1	1	7.09
1	-0.418344	0.557663	-0.869256	8.897
2	-0.971551	-1.452714	0.648145	7.491
3	0.126205	0.145301	-0.022167	6.869
4	-0.024991	-0.028846	0.004589	6.846
5	0.006247	0.007212	-0.001147	6.279
6	-0.001575	-0.001816	0.000283	6.786
7	0.000374	0.000429	-0.000062	6.876
8	-0.000067	-0.000075	0.000008	6.915
9	-0.000003	-0.000005	0.000002	7.069

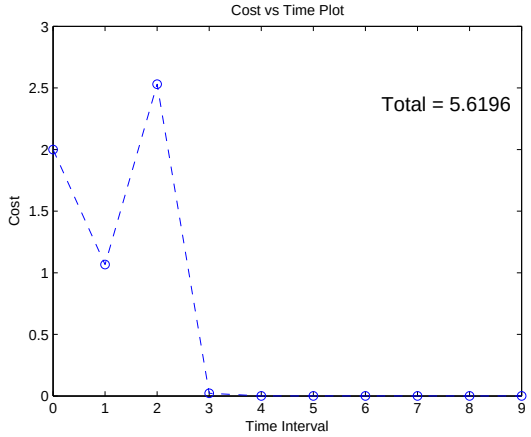


Figure 5.40: Cost vs. Time Plot for Van der Pol using SQP with Initial $u = -1$, $N = 5$

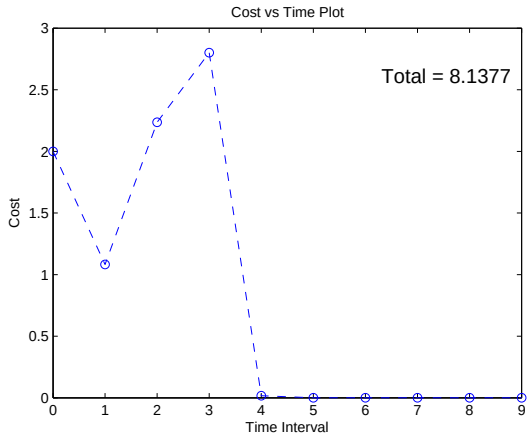


Figure 5.41: Cost vs. Time Plot for Van der Pol using SQP with Initial $u = 0.2$, $N = 5$

Table 5.36: NMPC Algorithm Results using SQP Van der Pol System (Limit Cycle behaviour) as the constraint and Initial $u = 0.7$ and $N = 5$

Time Interval	**Input u(k)**	**State x_1**	**State x_2**	**Computation Time (sec)**
0	0.990476	1	1	14.151
1	0.843744	1.155272	-0.368371	10.879
2	-0.161452	-0.186656	0.029358	8.801
3	0.035492	0.040973	-0.006522	9.377
4	-0.008885	-0.010254	0.001628	9.218
5	0.002212	0.002552	-0.000403	9.612
6	-0.000539	-0.000622	0.000097	9.402
7	0.000132	0.000151	-0.00002	9.753
8	-0.000016	-0.000018	-0.000001	9.255
9	-0.000004	-0.00001	0.000008	9.669

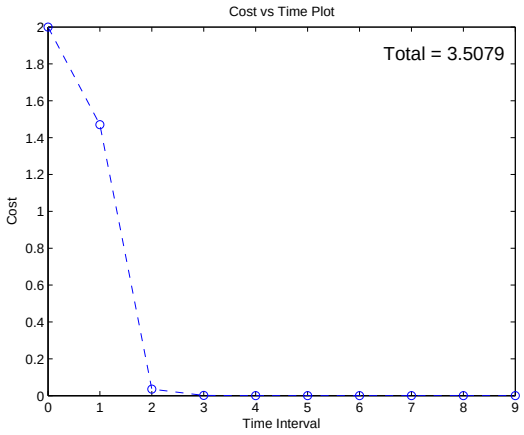


Figure 5.42: Cost vs. Time Plot for Van der Pol using SQP with Initial $u = 0.7$, $N = 5$

5.4 Results Overview

This chapter presented and applied the optimization roadmap to several examples in the static (constrained and unconstrained) optimization domain as well as in a NMPC context. The two optimization algorithms (SQP and PSO) were contrasted against each other in terms of computation time, ability to escape local minima and were also used in validating the insights provided by the optimization roadmap.

The examples in the static optimization domain verified the applicability of the methodology by providing insight into the optimization problem. The application of the methodology to the unconstrained 2-Dimensional Rastrigin function in Section 5.2.1.4 in particular highlighted important insights such as the non-convexity of the function. The application of the methodology to Example 3 of the static constrained optimization problem in Section 5.2.2.4 was a key example in leading on to the application of the methodology in the NMPC context since the constraint is typical of that encountered in NMPC where the states are bound by the dynamics of the system.

The application of the optimization roadmap to the NMPC examples brought about several insights. Firstly, the methodology validated that the linear system with the quadratic objective function produced an easily solved convex optimization problem. Secondly, the optimization roadmap validated and graphically represented the non-convex optimization problem resulting from a nonlinear model (constraint) in spite of the quadratic objective function. The Van der Pol system examples brought forth further insights in that it was shown that the same nonlinear system (with varied parameter values) may have significantly different dynamic characteristics and hence affect the optimization problem in different ways. The optimization roadmap methodology graphically showed this phenomenon where with certain parameter (μ) values and initial conditions the system (although nonlinear) behaved in a *qualitatively linear* manner and resulted in a convex optimization problem to be solved by the optimization problem while with a different μ value and initial conditions the resulting optimization problem was non-convex.

The comparison between the optimization algorithms, in the examples that they were applied to in this study, showed that the SQP is an effective and fast local optimization method while the PSO (for the parameter settings used in the application) is effective in finding the global minima but is comparatively slow. In the Van der Pol oscillator where the system is defined by a differential equation the computation time taken by the PSO algorithm was inordinately long in comparison with the SQP algorithm.

Furthermore, initial condition regions that would put the SQP algorithm in a region such that it would be able to locate the global (or close to global) minima, could be identified from the Optimization Roadmap. The results, in the NMPC examples verify that when starting the SQP algorithm in particular trapping regions, it would get stuck in local minima. On the other hand,

starting the SQP algorithm in a region where it was able to locate global (or close to global) minima, resulted in the same total cost as obtained by PSO while producing the results in a fraction of the time.

Relating back to the MPC-car driving strategy analogy introduced in Chapter 1, the optimization roadmap provides the user with the terrain or road conditions that will be faced by the optimization algorithm. If the optimization algorithm is considered as the analog of the vehicle, then it may be said that the optimization roadmap gives the user insight or knowledge into selecting the most suited vehicle for the terrain. As an example, off-road terrain with inclines and declines may be best suited to a 4x4 off-road vehicle that will be able to deal with the inclines and declines and off-road terrain albeit not the fastest vehicle available. Analogously, if the optimization roadmap depicts the optimization problem as non-convex with many local minima, a global optimization algorithm such as PSO may be best suited. While, on the other hand a sedan (or even a sports car) may be considered ideal for a smooth tarred highway. However, the choice depends on the goal of the user. Since, the optimization roadmap provides the user with regions that may be problematic, the user may use this information and select a route or starting point that may avoid the problematic areas. Hence, using the knowledge of the problematic areas or regions (for example offroad areas) and selecting a route that averts them, a sedan may be employed thereby possibly getting to the destination quicker than if a 4x4 was used.

A shortcoming of the methodology may be identified as the user being required to select the *points of interest* from the plots since this may lead to a user missing key features of the problem. Additionally the increment size may also lead to a similar problem. A caveat to these, is that the goal of the methodology is to gain an understanding and insight which may be an iterative process and require users to invest additional time in order to attain the knowledge via the methodology. However, with the user interacting and iterating through the methodology, valuable knowledge is continuously gained with possible returns coming from reducing costs, reducing the number of trial and error approaches.

Chapter 6

Conclusions

An important challenge identified in the NMPC algorithm is that of the optimization component. This study provides an approach to address this challenge by means of providing visual insight into the optimization problem in NMPC. This assists the user in gaining an understanding of the problem and thus provides a platform for making informed decisions about the approach to solving the problem.

The approach takes advantage of the methodology of NMPC whereby the dynamic optimization problem is converted into a static optimization problem at every interval. As expounded on in Chapter 5, the approach proposed in this work provides a 2-Dimensional plot(s) of the cost function value versus the independent variable (the control input) for every time interval. These plots are limited by the choices made by the user in order to prevent an overload of plots. The resulting plots provided to the user, display the *"topography"* in which the optimization will be operating and thus provides a host of valuable insights. The insights provided to the user include the following:

1. Extent of the effect of the system or non-quadratic objective function on the optimization
2. Regions of non-convexity
3. Initial conditions regions that lead either to the local or global minimum
4. Type of optimization algorithm to use i.e. global or local.

The first point in the above list highlights that the proposed methodology provides a view into how the objective function (for example a quadratic objective function) is influenced by a non-linear system in terms of optimization. Secondly, the plots provided to users visually illustrate regions of non-convexity within each plot, providing the user with an opportunity to leverage this information. The insights into the optimization provide the user with the advantage of identifying regions wherein initial conditions could be set such that a global optimum is reached with a local optimization algorithm. Depending on the results of the plots and the application domain, the proposed Optimization Roadmap methodology will enable users to identify whether a local

optimization algorithm, for example SQP (which is much quicker than PSO as seen in Chapter 5), will suffice in terms of reaching the global optimum or whether a global optimization algorithm is required.

The results presented in Chapter 5 have demonstrated the above-mentioned value that can be derived from the Optimization Roadmap methodology in an NMPC context. It was shown that by using the Optimization Roadmap, users would be able to understand and hence gain confidence into what can be expected from the optimization algorithm as opposed to running the system *blindly*. The Optimization Roadmap methodology provided a view into the convexity or non-convexity of the system which, as shown in the examples, is dependent on the system even when using a quadratic objective function. The Van der Pol example illustrated this since, for certain operating regions, the optimization problem was convex while in others it was non-convex. This insight alone is immensely beneficial since users can use it to approach the problem in a suitable manner, according to the requirements. The insights gained also provide users with the advantage of being able to identify methods for improving the performance of the system, for example in terms of speed or profitability (when linked to global optima).

The goal within this study was not to investigate a new optimization methodology although from the plots (depending on the granularity) the minima may be deduced. Furthermore, the proposed methodology may be seen as a "brute force" method but as previously alluded to, the objective of the proposed methodology is not to ***find*** the minima (which may come out as a consequence) but rather to glean insight into the system and use these insights to make educated or informed decisions. The methodology could be seen as computationally expensive but with the advances in technology and speed of computational hardware this should not be a limiting factor. Moreover, this methodology can be used *a priori* and is not recommended to be real time or online since the goal is to provide users with insights and understanding of the system's behaviour within the optimization step.

The comparison of the two optimization algorithms also provided a degree of affirmation into the pragmatism and usefulness of the proposed Optimization Roadmap methodology in that when applying the various algorithms one was able to (with some degree of confidence) anticipate the result. Secondly, a useful result of the comparison became apparent when looking at the computation time of the SQP and PSO algorithms. The use of one or the other algorithm requires a trade-off either of time or reaching the global optimum. The results clearly demonstrated that while the PSO algorithm was able to reach global optima, in most instances, it took significantly longer to reach the solution. Conversely, SQP reached a solution significantly faster than PSO but would get trapped in local minima (depending on the initial conditions).

This study was limited to investigating single-input systems but with additional *housekeeping* or

tracking of input-value to cost indices can be extended to multiple input systems. It should be noted that the extension would not require additional conceptual development, but would simply require increased computational effort (in terms of tracking the indices). Building on this work, which focused on investigating the methodology to provide insight into the optimization, future work could identify and implement functionality to provide this methodology as a complete tool to users. The additional functionality could include a dynamic granularity feature which allows users to change the granularity of a plot dynamically and hence *zoom in* on particular regions. Furthermore, a point filtering option where users are able to select which points to view would be useful, particularly in the multiple input case and in the case of systems with additional constraints. Future work could also include an *intelligent* component to aid decisions made by users.

Appendix A

This Appendix contains flowcharts of the Optimization Roadmap methodology and the various components of the NMPC software used in this study.

Static Unconstrained Optimization Roadmap Methodology

The following is a flowchart of the Optimization Roadmap methodology for the static unconstrained optimization problem.

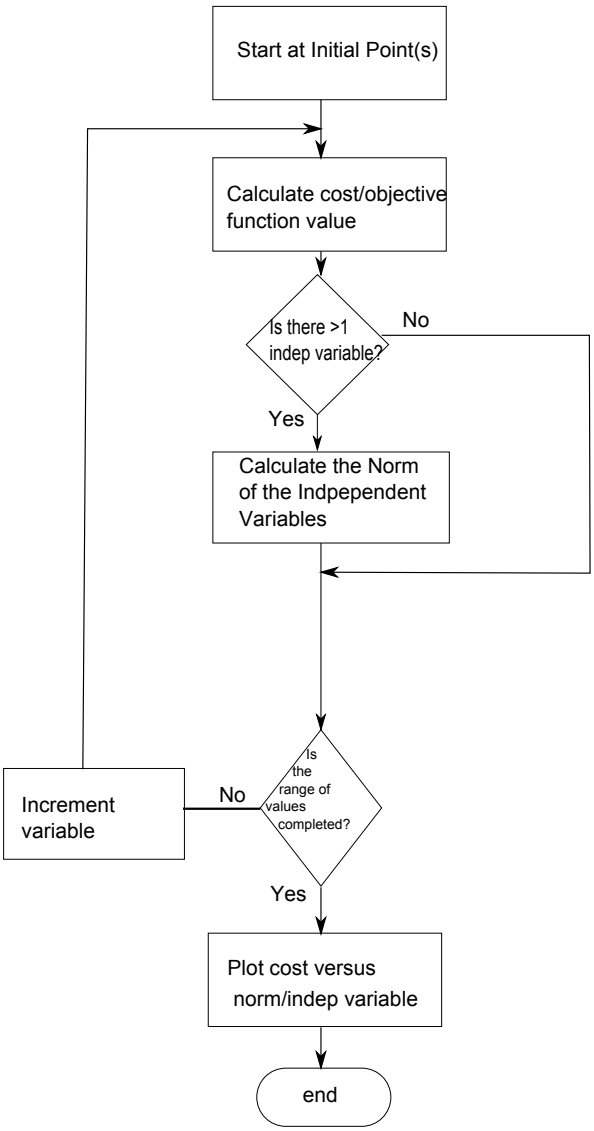


Figure A.1: Flow Chart of Static Unconstrained Optimization Roadmap Strategy

Static Constrained Optimization Roadmap Methodology

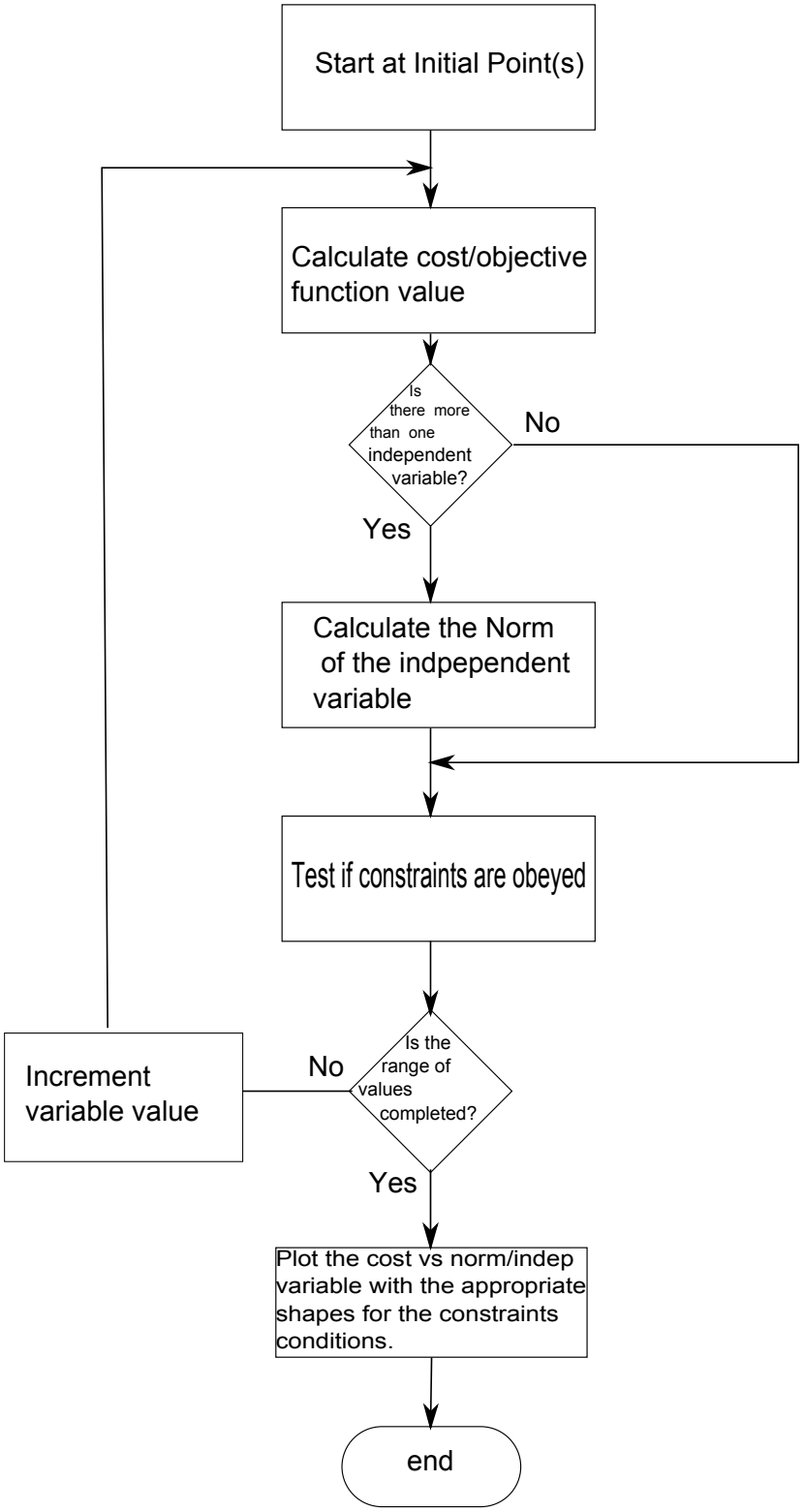


Figure A.2: Flow Chart of Static Constrained Optimization Insight Strategy

NMPC Implementation Discussion

In this section the implementation of the Matlab NMPC algorithm in [1] will be discussed by expounding on the flow charts of the functions and the code. The goal of this discussion is to understand the implementation in order to be able to fully use and modify functions where required.

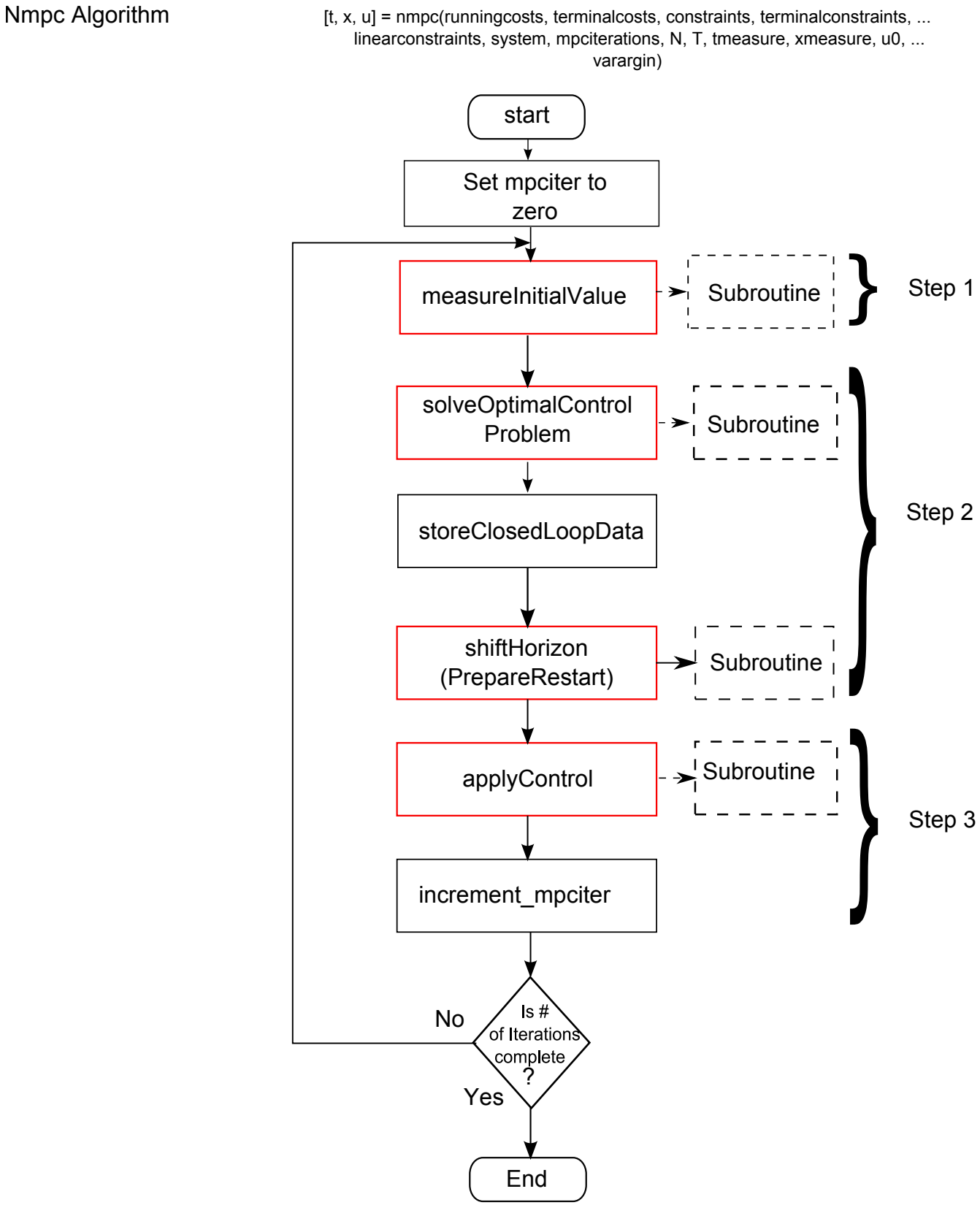


Figure A.3: Flow Chart of NMPC Algorithm [1]

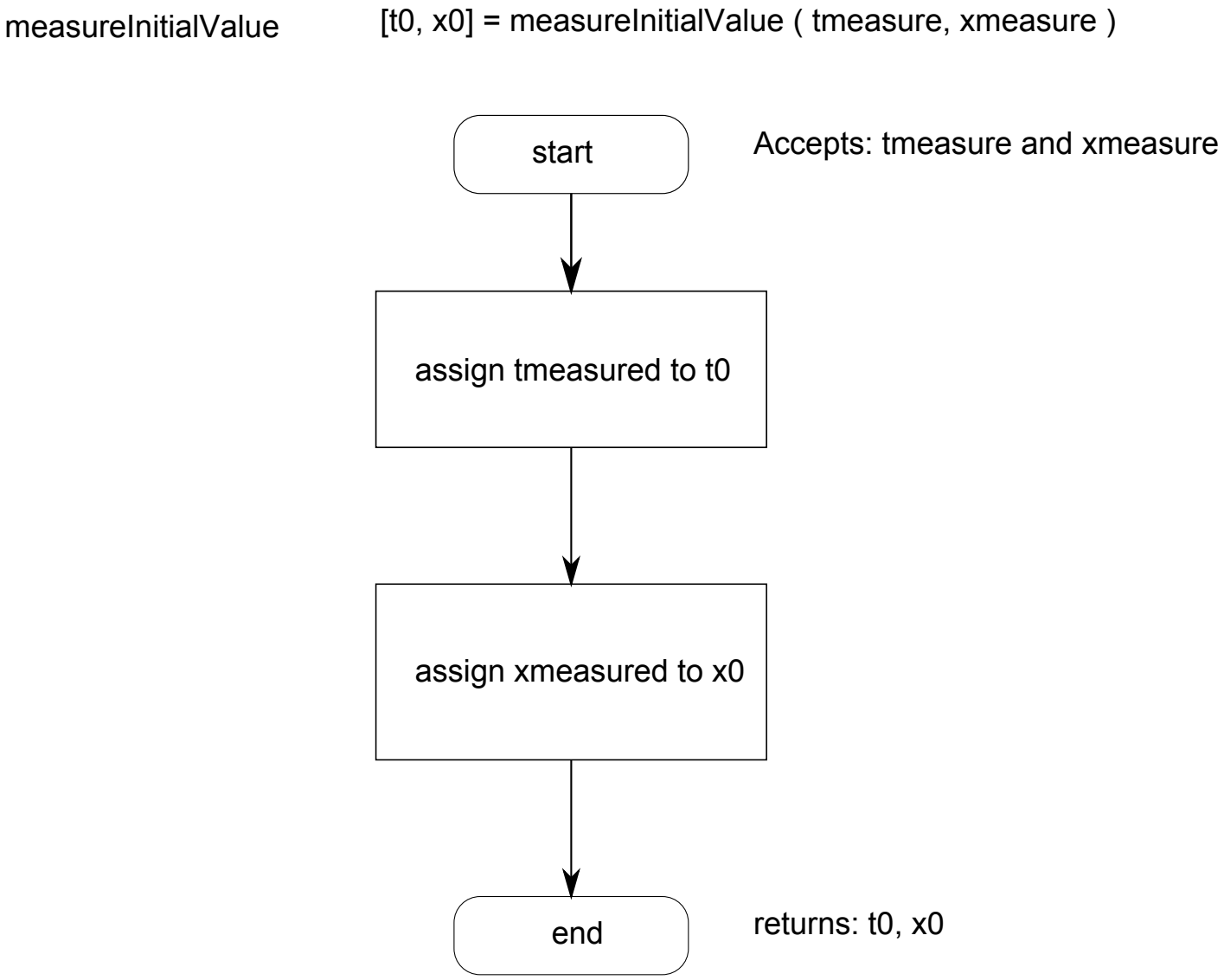


Figure A.4: Flow Chart of measureInitialValue Function [1]

***measureInitialValue*[1]**

measureInitialValue is the first step in the NMPC algorithm and this entails setting the initial values *t0* and *x0*. In the first iteration the values of *tmeasure* and *xmeasure* are obatined by the initial conditions provided by the user (calling function) and these are then set to the initial values *t0* and *x0*.

Following this *t0* and *x0* are given by the values returned by *applyControl* where *x0* is the initial state vector and *t0* is the sampling instant.

measureInitialValue returns [x0,t0]

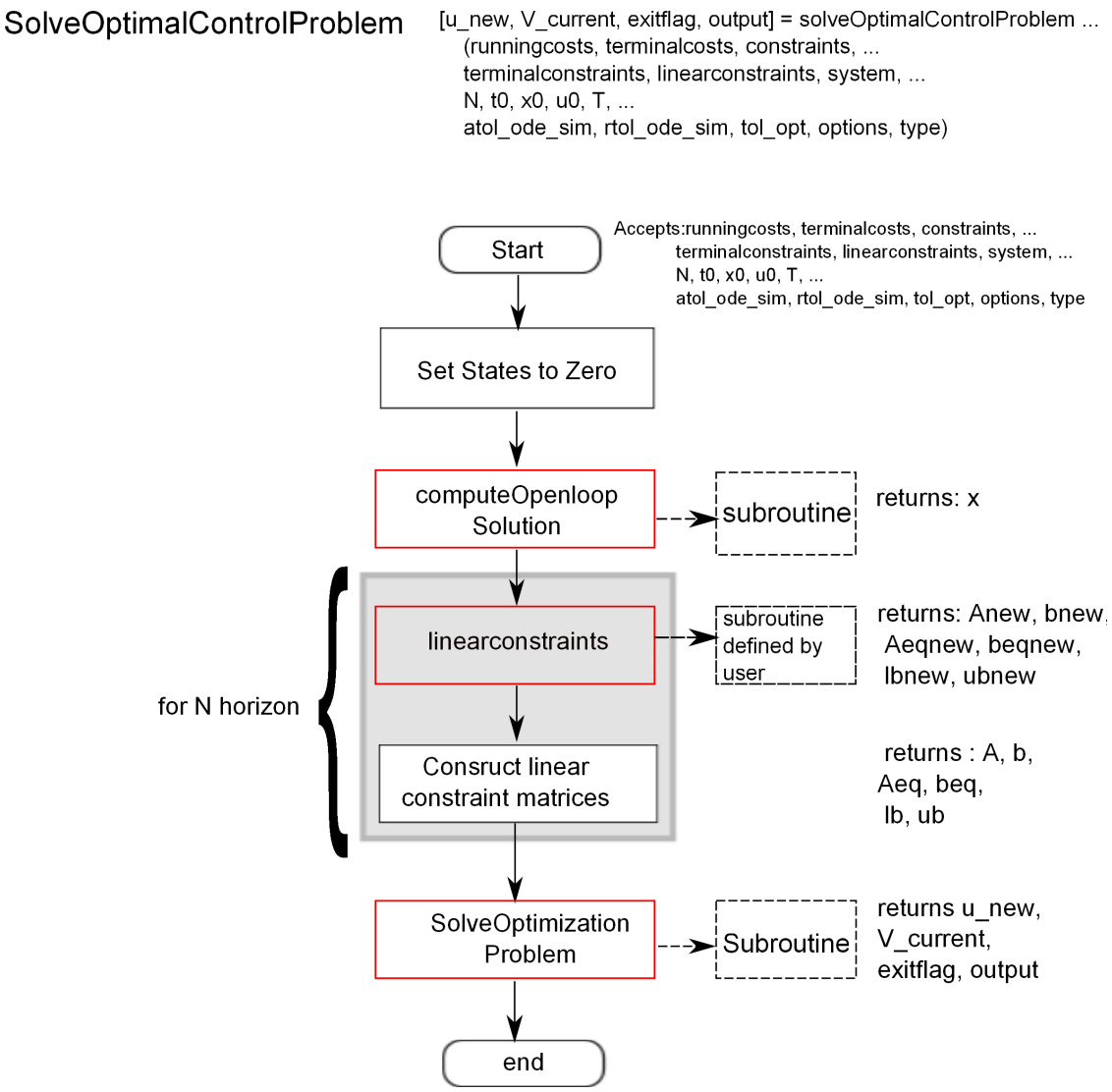


Figure A.5: Flow Chart of solveOptimalControlProblem Function [1]

solveOptimalControlProblem[1]

This function is considered step 2 of the three steps of the NMPC algorithm and accounts for a large amount of the computation required in NMPC. The function name is self explanatory in the sense that this function solves the optimal control problem as given by the algorithm (insert block diagram).

A recursive discretization technique is adopted in the algorithm which in essence divides or reduces the optimal control into lower level problems. These sub-problems being the system dynamics problem and the optimization problem which are then solved separately but with interaction in that values are passed to and from each problem. This recursive discretization thus allows one to solve a static problem rather than a dynamic one.

How does *solveOptimalControlProblem* Work

Step 1 :

- Get Open Loop State values using (ComputeOpenLoop,dynamics,system)
- For the first iteration t0,u0 are obtained from tmeasured and xmeasured
- Thereafter x0,u(·) and obtained from optimization routine i.e first values of x

Step 2 :

- Set the linear constraints of the discretized Optimal Control Problem which are mainly for control and state bounds
- This is done for the horizon

Step 3 :

- State values that are computed by dynamics are sent to optimization routine
- Uses previously calculated Uopt as initial points (u0)
- Calculate new optimal control sequence from the optimization routine
- Send control sequences u(·) and initial state values (x0) to dynamics solver

computeOpenLoopSolution x = computeOpenloopSolution(system, N, T, t0, x0, u, ...
atol_ode_sim, rtol_ode_sim, type)

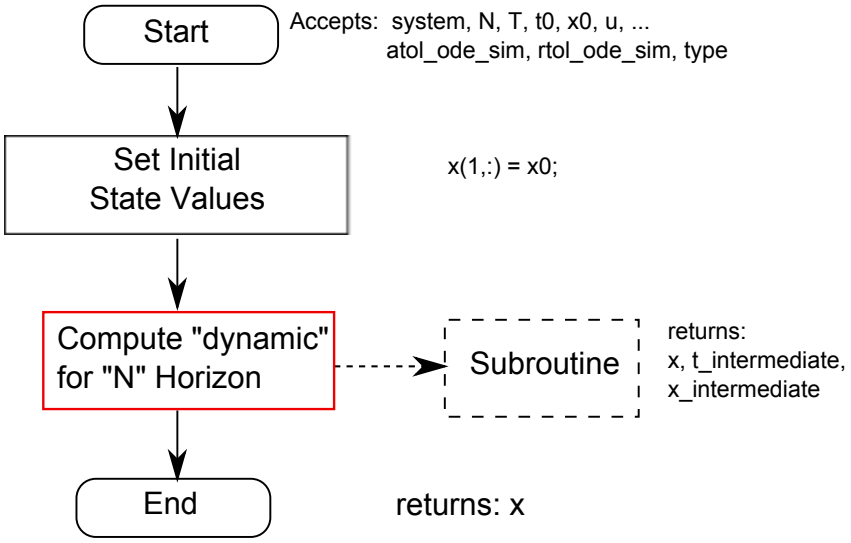


Figure A.6: Flow Chart of computeOpenloopSolution Function [1]

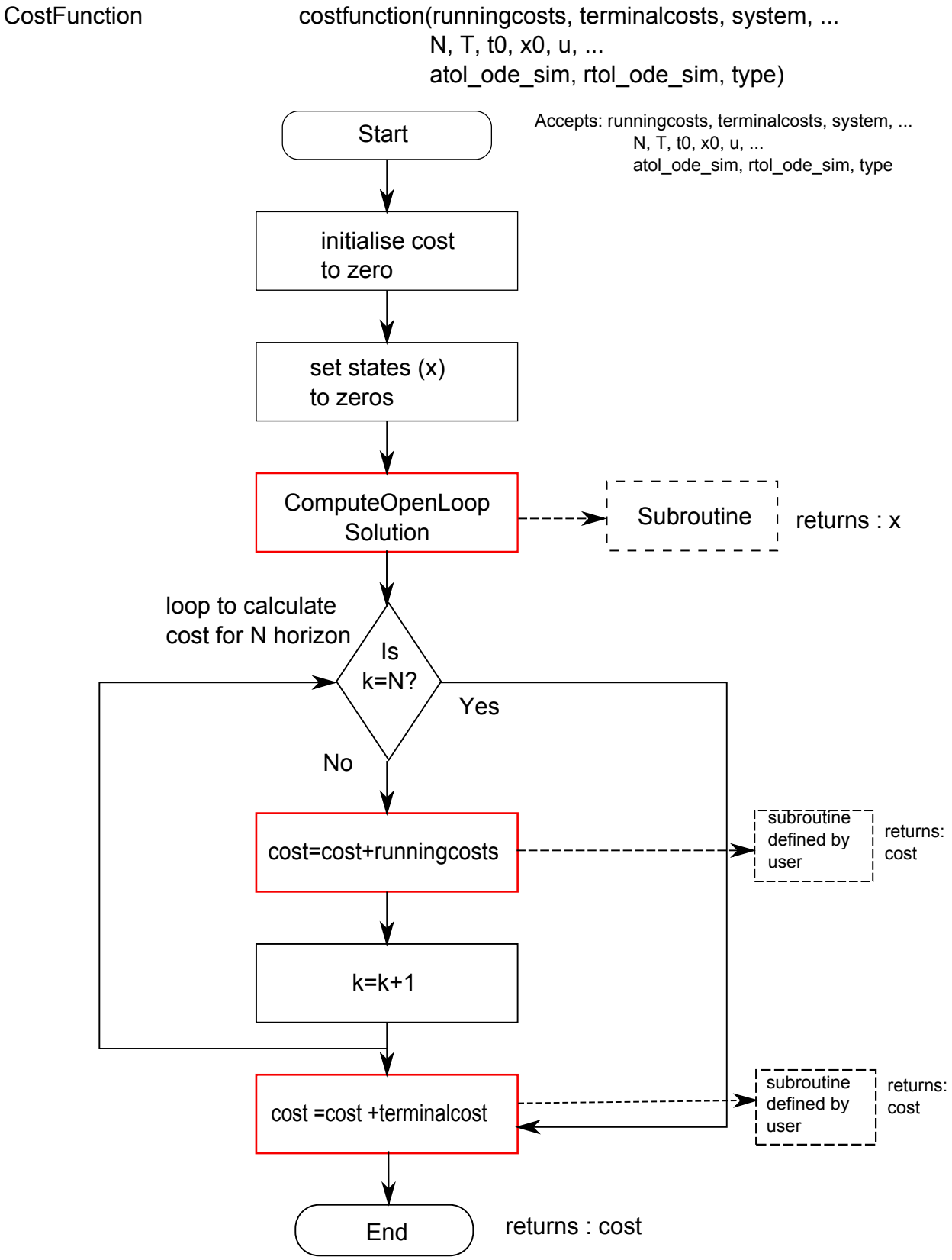


Figure A.7: Flow Chart of costfunction Function [1]

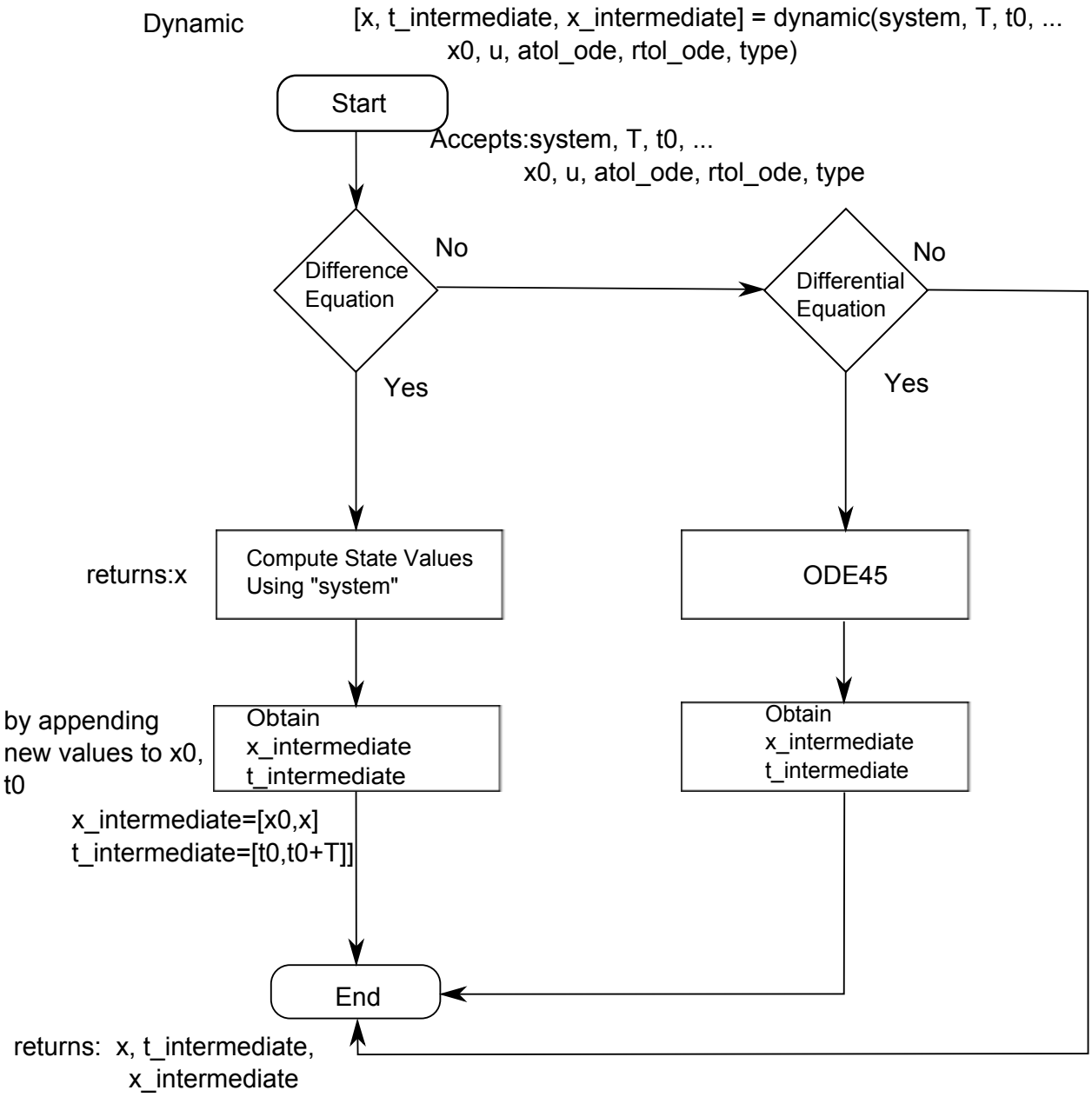


Figure A.8: Flow Chart of dynamic Function [1]

Store Closed Loop Data[1]

This block in the NMPC algorithm is where the closed loop data is stored. This is done by appending *tmeasure*, *xmeasure*, *u_new* onto the existing t, x and u vectors.

The closed loop data is obtained from *applyControl*.

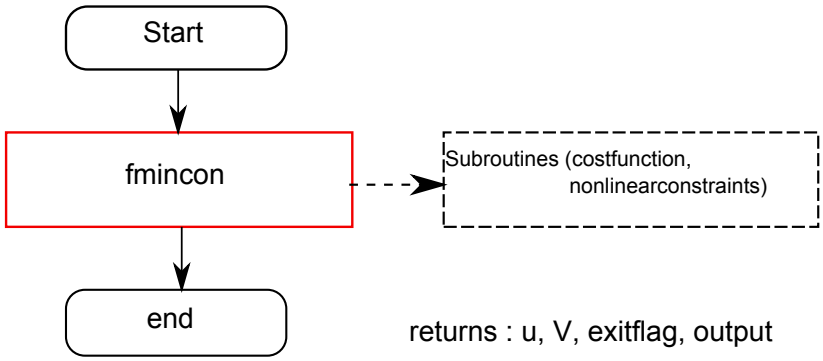
***shiftHorizon*[1]**

Since only the first element of *u_new* is used or applied to system the *shiftHorizon* function shifts the open-loop control which eases the restart. *shiftHorizon* also falls within step 2 of the NMPC algorithm.

shiftHorizon returns *u0* which is the new initial guess of the control.

SolveOptimizationProblem

```
[u, V, exitflag, output] = fmincon(@(u) costfunction(runningcosts, ...
    terminalcosts, system, N, T, t0, x0, ...
    u, atol_ode_sim, rtol_ode_sim, type), u0, A, b, Aeq, beq, lb, ...
    ub, @(u) nonlinearconstraints(constraints, terminalconstraints, ...
    system, N, T, t0, x0, u, ...
    atol_ode_sim, rtol_ode_sim, type), options)
```



finds the minimum of a problem specified by :

min f(x) such that c(x)<=0
ceq(x)=0
A.x<= b
Aeq.x = beq
lb<=x<=ub

x,b,beq,lb,ub are vectors
A,Aeq are matrices

Figure A.9: Flow Chart of SolveOptimizationProblem Function [1]

fmincon[2]

fmincon has the following syntax:
[2] x = fmincon(fun,x0,A,b,Aeq,beq,lb,ub,nonlcon,options)
The *fmincon* function is called in the NMPC routine as below:
[u, V, exitflag, output] = fmincon(@(u) costfunction(runningcosts, terminalcosts, system, N, T, t0, x0, u, atol_ode_sim, rtol_ode_sim, type),...
u0, A, b, Aeq, beq, lb, ub,
@(u) nonlinearconstraints(constraints, terminalconstraints, system, N, T, t0, x0, u, atol_ode_sim, rtol_ode_sim, type), options);

fmincon syntax	How it is called in nmpc
fun	@(u) costfunction(. . .)
x0	u0
A	A
b	b
Aeq	Aeq
lb	lb
ub	ub
nonlcon	@(u) nonlinearconstraints(. . .)
options	options

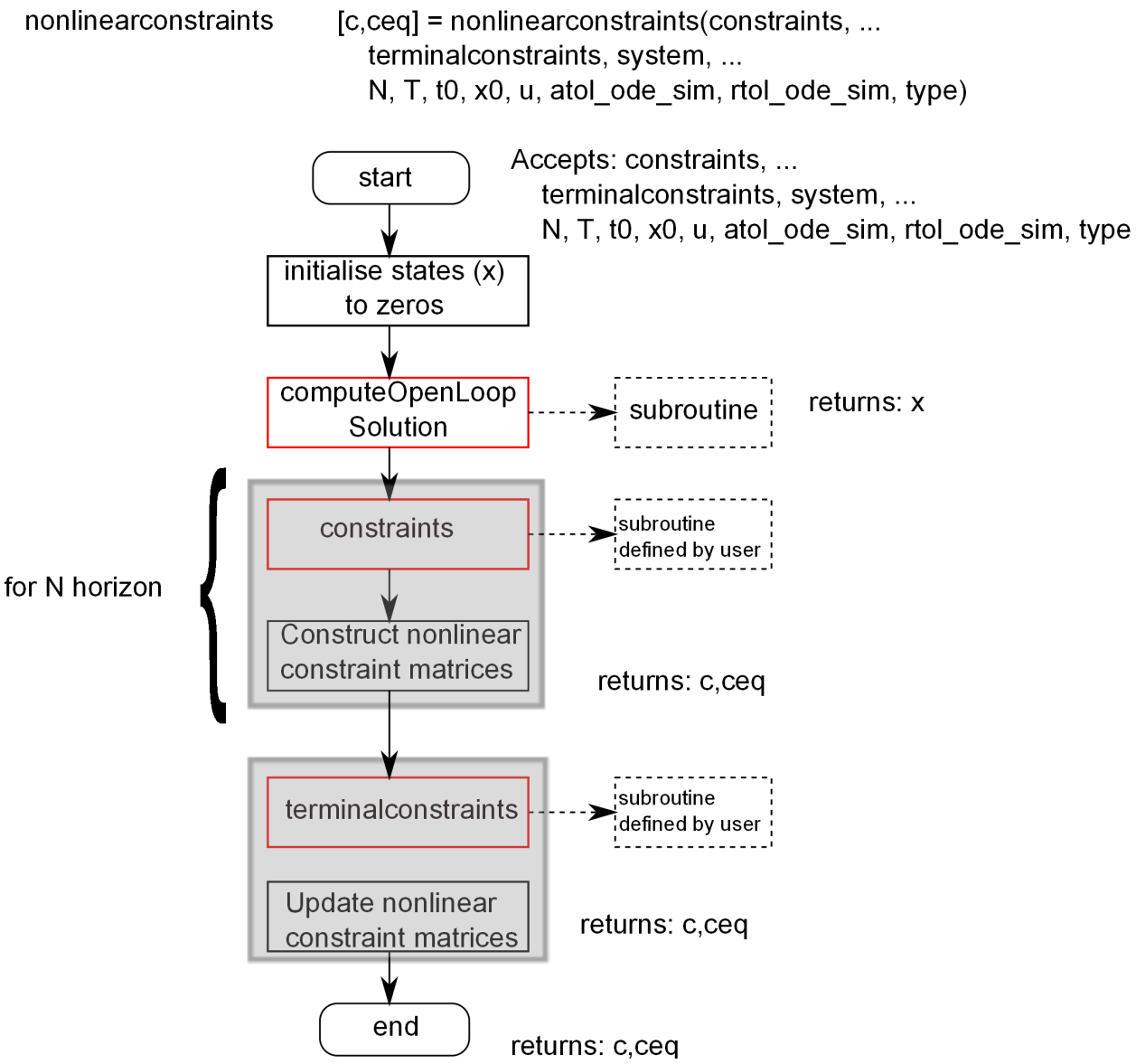


Figure A.10: Flow Chart of nonlinearconstraints Function [1]

applyControl[1]

applyControl is the third step in the NMPC algorithm. This method uses the control input (*u_new*) calculated by *solveOptimalControlProblem* and applies the first element to the system or process for one sampling interval. This is done using the *dynamic* function which in-turn calls the *system* function.

applyControl returns the closed-loop state vector [tmeasure, xmeasure] where *tmeasure* is the sampling instant and *xmeasure* is the closed-loop state vector.

applyControl [tmeasure, xmeasure] = applyControl(system, T, t0, x0, u_new, ...
atol_ode_real, rtol_ode_real, type);

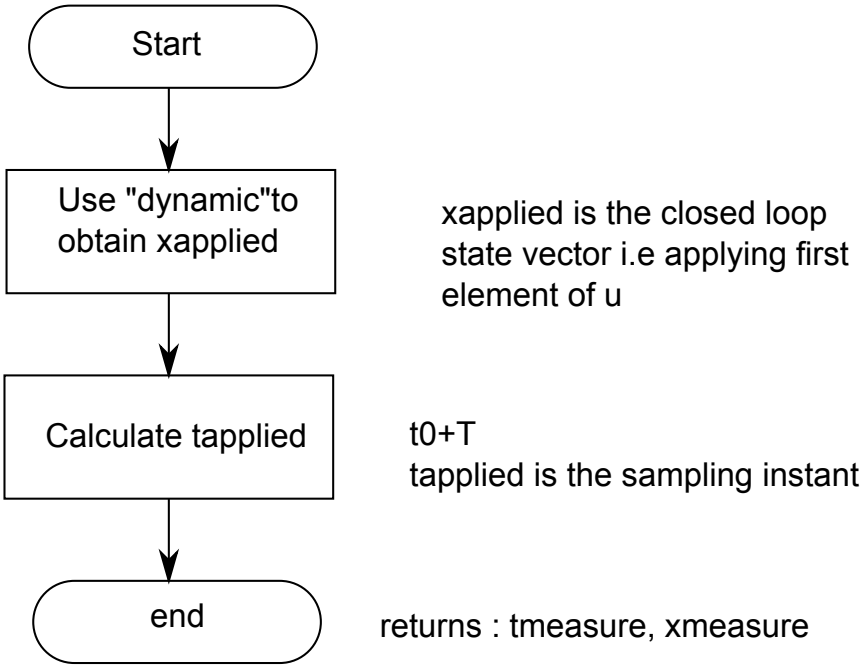


Figure A.11: Flow Chart of applyControl Function [1]

nmpc.m[1]

- Initialise t,x,u to empty matrices
- Set mpciter to zero

- begin while loop(i.e run nmpc for number of mpciterations which is specified by the user) while mpciter< mpciterations

Obtain new Initial Value

[t0,x0] = measureInitialValue(tmeasure,xmeasure)

Start Stopwatch Timer

t_Start=tic;

Solve Optimal Control Problem

[u_new, V_current,exitflag,output]=SolveOptimalControl(. . .)

Set states to Zeros

- Each column is a state
- x is of size (N + 1, length(x0))

Get Open Loop State values x = computeOpenloopSolution(system, N, T, t0, x0, u0, atol_ode_sim, rtol_ode_sim, type);

- Set initial values to states x(1,:)=x0;

Variable	Meaning
mpciterations	♯ of iterations of NMPC algorithm
N	Length of optimization/prediction horizon
T	Sampling interval
tmeasure	Time measurement of initial/ value
xmeasure	State measurement of initial value
u0	Initial guess of open loop control
t_n	Current sampling instant

Appendix B

NMPC Example 1: Results

Table B.1: NMPC Algorithm Results SQP with Initial $u = -10$, $N = 5$

Time Interval	**Input u(k)**	**State** x_1	**State** x_2	**Computation Time (sec)**
0	-8.440095	2	1	1.134
1	-1.394468	-0.044009	0.7	0.1
2	-0.400804	-0.01265	0.201198	0.064
3	-0.115201	-0.003636	0.05783	0.063
4	-0.033111	-0.001045	0.016622	0.062
5	-0.009516	-0.0003	0.004778	0.062
6	-0.002734	-0.000086	0.001373	0.062
7	-0.000785	-0.000025	0.000395	0.062
8	-0.000225	-0.000007	0.000113	0.062
9	-0.000064	-0.000002	0.000033	0.062

Table B.2: NMPC Algorithm Results PSO with Initial $u = -10$, $N = 5$

Time Interval	**Input u(k)**	**State** x_1	**State** x_2	**Computation Time (sec)**
0	-8.440177	2	1	33.697
1	-1.3937	-0.044018	0.7	31.599
2	-0.400578	-0.012575	0.201196	35.907
3	-0.115543	-0.003591	0.057844	25.725
4	-0.0322	-0.001063	0.016635	34.422
5	-0.010212	-0.000212	0.004778	32.696
6	-0.001793	-0.000129	0.001391	36.352
7	-0.000873	0.00006	0.000391	34.544
8	-0.001762	0.000009	0.000129	32.841
9	0.001407	-0.000148	0.000041	31.933

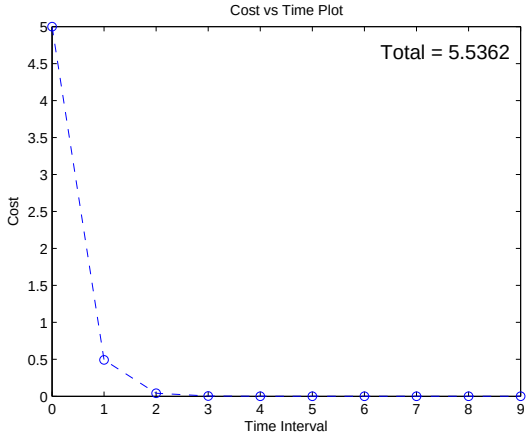


Figure B.1: Example 1: Cost vs. Time Plot for PSO

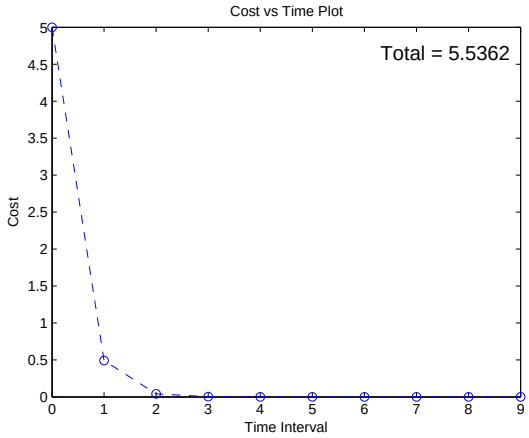


Figure B.2: Example 1: Cost vs. Time Plot for SQP with Initial $u = -10$

NMPC Example 2: Results

Table B.3: NMPC Algorithm Results using SQP Example 2 with Initial $u = 5$, $N = 5$

Time Interval	**Input u(k)**	**State** x_1	**State** x_2	**Computation Time (sec)**
0	3.407936	1	1	0.96
1	-5	-0.734419	-0.763205	0.071
2	5	0.118329	0.973446	0.037
3	5	0.33344	-0.473858	0.026
4	-4.287431	0.020766	-1.232926	0.06
5	5	-0.64471	0.294584	0.014
6	5	0.168228	-0.543658	0.014
7	-4.286146	0.008393	-1.235514	0.065
8	5	-0.648493	0.292768	0.015
9	5	0.171079	-0.539822	0.015

Table B.4: NMPC Algorithm Results using SQP Example 2 with Initial $u = -5$, $N = 5$

Time Interval	**Input u(k)**	**State** x_1	**State** x_2	**Computation Time (sec)**
0	-3.771321	1	1	0.836
1	4.716239	-0.703108	0.088926	0.068
2	-5	0.159004	-0.60794	0.035
3	-5	-0.010776	0.650934	0.063
4	-5	0.139476	1.284393	0.043
5	-2.534528	0.692325	1.598407	0.044
6	-5	0.914218	-0.103097	0.014
7	-5	-0.49691	0.143277	0.014
8	-4.980471	0.08668	1.15326	0.041
9	-5	0.513125	1.540259	0.014

NMPC: Asymptotically Stable Van der Pol Results

Table B.5: NMPC with SQP Optimization Algorithm of Quadratic Cost Function with Van der Pol Asymptotically Stable Dynamics with Initial $u = 0.9$, $N = 5$

Time Interval	**Input u(k)**	**State x_1**	**State x_2**	**Computation Time (sec)**
0	-0.114492	0.2	0.5	2.756
1	0.011209	0.01864	-0.096617	1.696
2	-0.000967	-0.001742	0.008512	1.668
3	0.000083	0.00015	-0.000732	1.665
4	-0.000007	-0.000013	0.000063	1.668
5	0.000001	0.000001	-0.000005	1.67
6	0	0	0	1.664
7	0	0	0	1.667
8	0	0	0	1.666
9	0	0	0	1.668

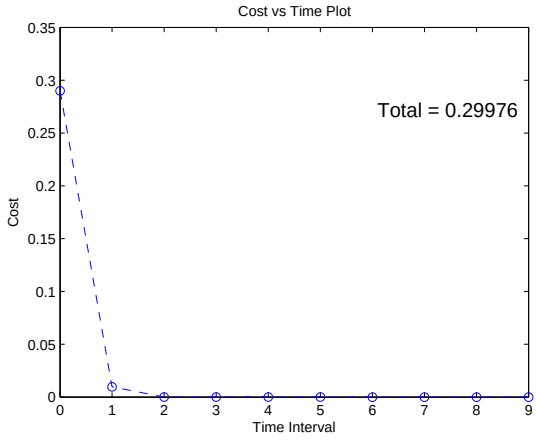


Figure B.3: Cost vs. Time Plot for Van der Pol SQP with Horizon $N = 5$ Initial $u = 0.9$

NMPC: Van der Pol System with Limit Cycle Results

Table B.7 shows the NMPC results using SQP optimization for the Van der Pol system with limit cycle dynamic characteristic with the initial $u = 1$.

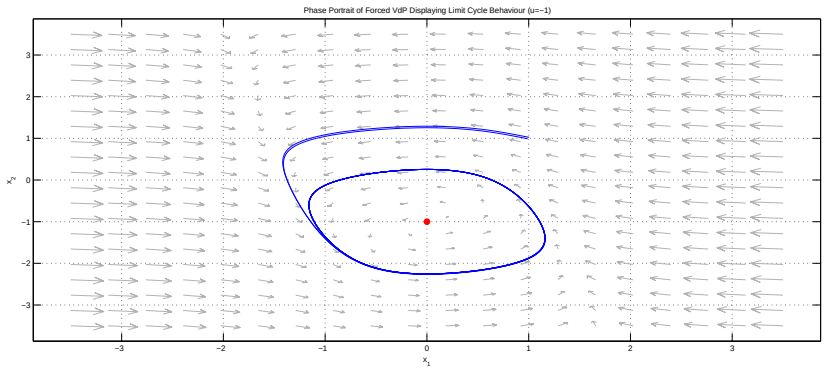


Figure B.4: Phase Plot for Van der Pol depicting Limit Cycle $u = -1$

Table B.6: NMPC Algorithm Results using SQP For Van der Pol with Limit Cycle with Initial $u = 0$, $N = 5$

Time Interval	**Input u(k)**	**State x_1**	**State x_2**	**Computation Time (sec)**
0	0.662469	1	1	16.561
1	-0.418344	0.557663	-0.869256	6.778
2	-0.971551	-1.452714	0.648145	6.267
3	0.126206	0.145302	-0.022166	6.142
4	-0.024986	-0.028841	0.004588	5.957
5	0.006259	0.007218	-0.001144	2.138
6	-0.00152	-0.001763	0.000291	5.33
7	0.000447	0.000513	-0.000077	1.572
8	-0.000091	-0.000104	0.000015	2.071
9	0.000009	0.000012	-0.000004	5.998

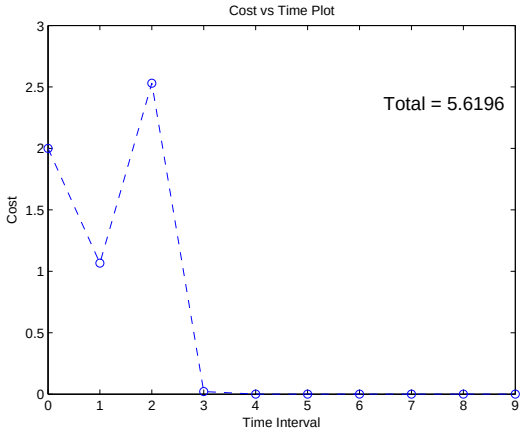
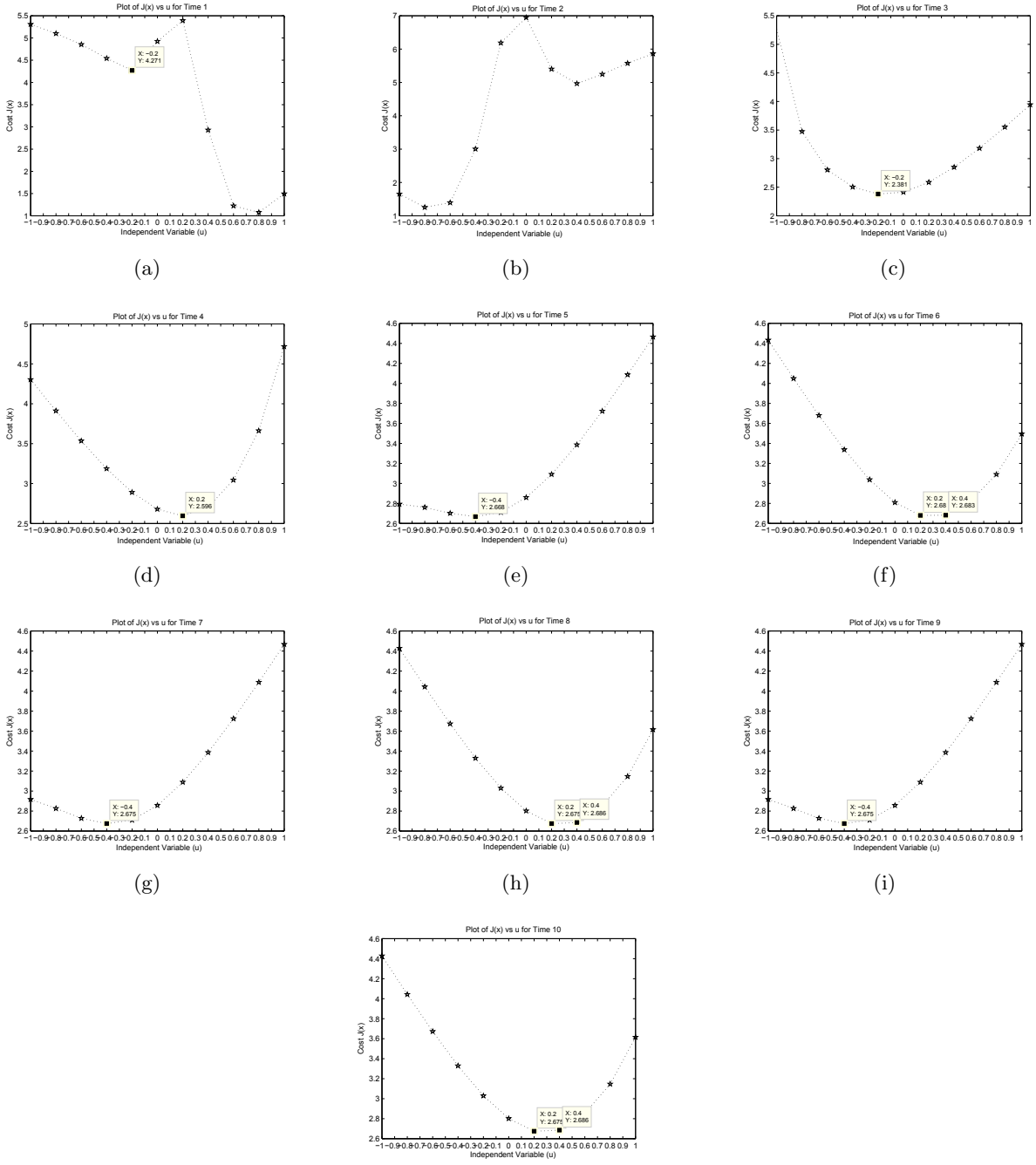


Figure B.5: Cost vs. Time Plot for Van der Pol SQP with Horizon 5 Initial $u = 0$



(a) (b) (c)

(d) (e) (f)

(g) (h) (i)

(j)

Figure B.6: Figure Showing the Path of the SQP algorithm in Limit Cycle Van der Pol with Initial $u = -1$

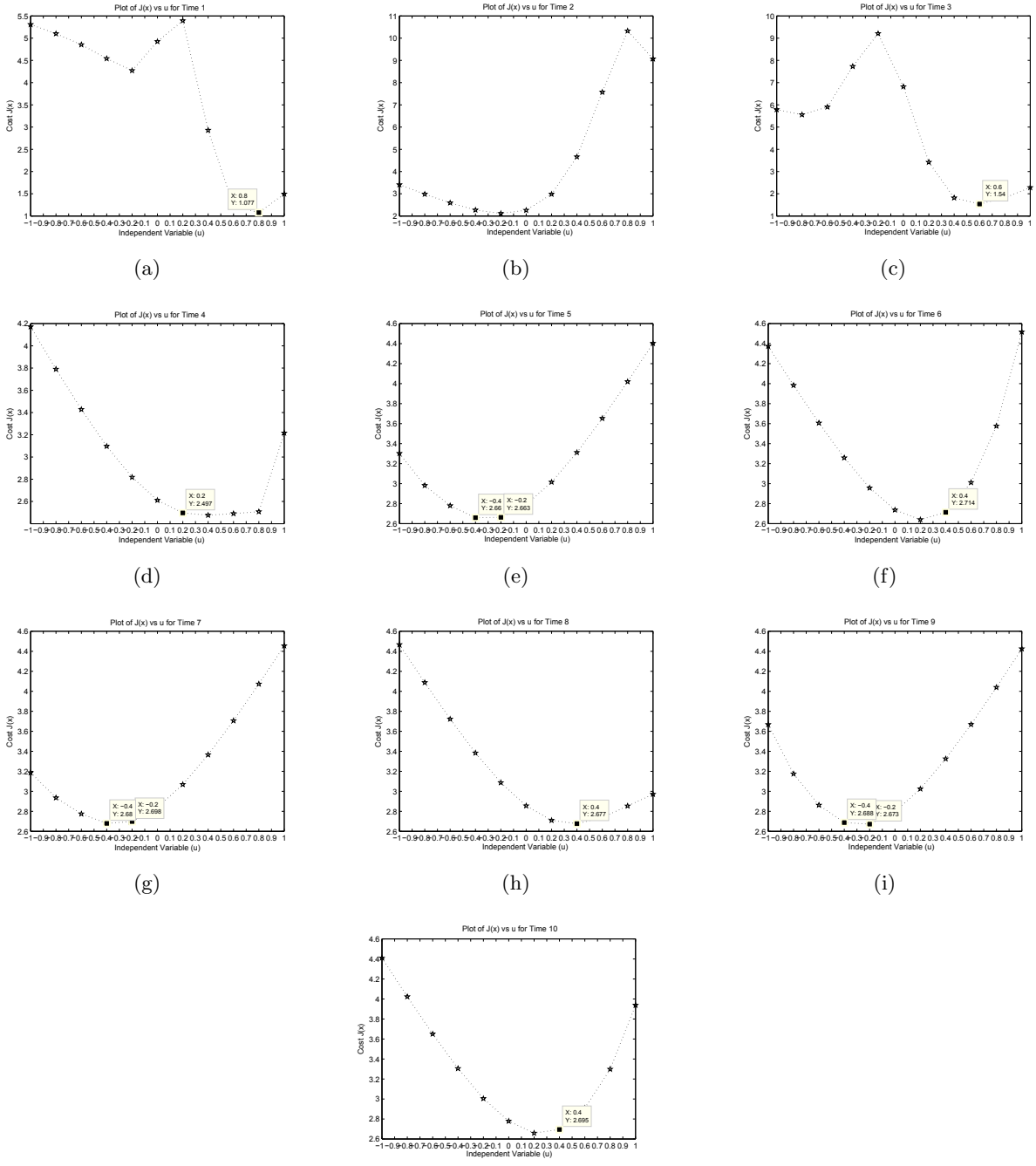


Figure B.7: Figure Showing the Path of the SQP algorithm in Limit Cycle Van der Pol with Initial $u = 1$

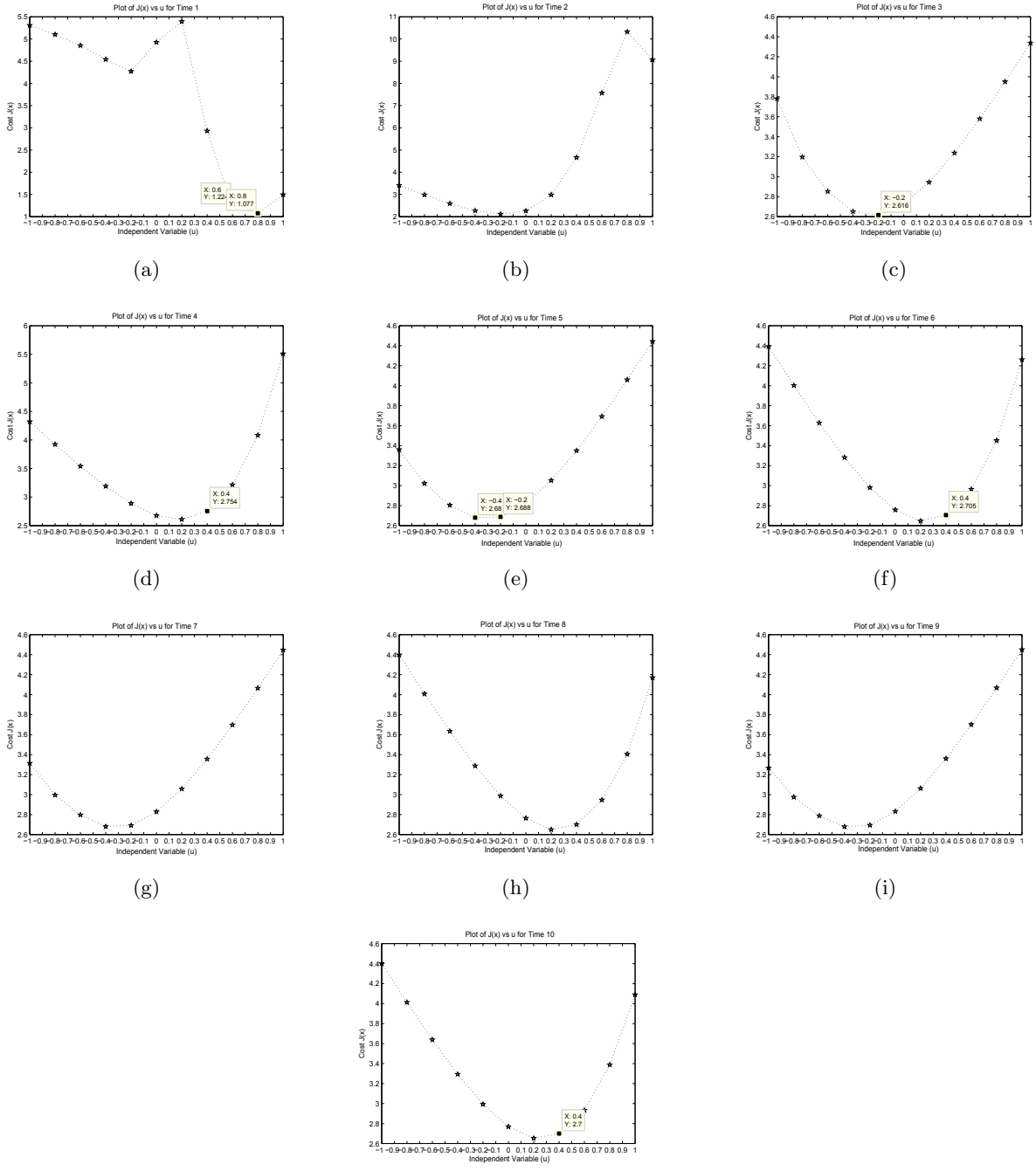


Figure B.8: Figure Showing the Path of the PSO algorithm in Limit Cycle Van der Pol

Table B.7: NMPC SQP Optimization of Quadratic Cost Function for Van der Pol System with Limit Cycle behaviour with Initial $u = 1$, $N = 2$

Time Interval	**Input u(k)**	**State** x_1	**State** x_2	**Computation Time (sec)**
0	0.723247	1	1	2.675
1	0.103091	0.69465	-0.734703	0.692
2	-0.255583	-1.102845	1.044101	0.384
3	0.386816	1.215657	-1.064964	0.334
4	-0.3154	-1.198286	1.109374	0.299
5	0.386172	1.225707	-1.079369	0.337
6	-0.322268	-1.202646	1.107703	0.288
7	0.380309	1.224321	-1.081808	0.343
8	-0.32762	-1.205198	1.105013	0.29
9	0.375525	1.223	-1.083696	0.341

References

[1] L. Grüne and J. Pannek, *Nonlinear Model Predictive Control: Theory and Algorithms*, 1st ed., ser. Communications and Control Engineering. Springer London, 2011.

[2] MATLAB, *version 7.10.0 (R2011b)*. Natick, Massachusetts: The MathWorks Inc., 2011.

[3] R. Findeisen and F. Allgöwer, "An introduction to nonlinear model predictive control," in *Control, 21st Benelux Meeting on Systems and Control, Veldhoven*, 2002, pp. 1–23.

[4] M. M. F. Borrelli, A. Bemporad, *Predictive Control for Linear and Hybrid Systems*. Cambridge, 2011.

[5] E. Camacho and C. Bordons, *Model Predictive Control*, ser. Advanced textbooks in control and signal processing. Springer London, 2007.

[6] S. Qin and T. A. Badgwell, "A survey of industrial model predictive control technology," *Control Engineering Practice*, vol. 11, no. 7, pp. 733–764, 2003.

[7] S. J. Qin and T. A. Badgwell, "An overview of nonlinear model predictive control applications," in *Nonlinear Model Predictive Control*, ser. Progress in Systems and Control Theory, F. Allgöwer and A. Zheng, Eds. Birkhäuser Basel, 2000, vol. 26, pp. 369–392.

[8] X. Yang, Z. Wu, Y. Li, and F. Shen, "Model and optimal control for urban traffic network under incident condition," in *Intelligent Transportation Systems Conference, 2007. ITSC 2007. IEEE*, Seattle, Washington, USA, 2007, pp. 1120–1125.

[9] Y. Fan, M. Da-wei, and C. Fu-hong, "A hierarchical control strategy for antiaircraft multiple launch rocket system," in *Intelligent Human-Machine Systems and Cybernetics (IHMSC), 2011 International Conference on*, vol. 1, Hangzhou, Zhejiang, China, 2011, pp. 31–34.

[10] D. Hanson, W. Perkins, and J. Cruz, "An updated open-loop control strategy for socio-economic systems," in *Decision and Control, 1972 and 11th Symposium on Adaptive Processes. Proceedings of the 1972 IEEE Conference on*, vol. 11, New Orleans, Los Angeles, USA, 1972, pp. 149–150.

[11] L. Wang, *Model Predictive Control System Design and Implementation Using MATLAB®*, ser. Advances in Industrial Control. Springer London, 2009.

[12] N. Blet, D. Megias, J. Serrano, and C. de Prada, "Nonlinear mpc versus mpc using on-line linearisation - a comparative study," in *Proceedings of the 15th IFAC World Congress*, Barcelona, Spain, 2002, pp. 591–591.

[13] C. E. Long, P. K. Polisetty, and E. P. Gatzke, "Deterministic global optimization for nonlinear model predictive control of hybrid dynamic systems," *International Journal of Robust and Nonlinear Control*, vol. 17, no. 13, pp. 1232–1250, 2007.

[14] L. Biegler, "Efficient solution of dynamic optimization and nmpc problems," in *Nonlinear Model Predictive Control*, ser. Progress in Systems and Control Theory, F. Allgöwer and A. Zheng, Eds. Birkhäuser Basel, 2000, vol. 26, pp. 219–243.

[15] D. Mayne, "Nonlinear model predictive control: Challenges and opportunities," in *Nonlinear model predictive control*, ser. Progress in Systems and Control Theory. Springer, 2000, vol. 26, pp. 23–44.

[16] F. Manenti, "Considerations on nonlinear model predictive control techniques," *Computers and Chemical Engineering*, vol. 35, no. 11, pp. 2491–2509, 2011.

[17] V. M. Zavala and L. T. Biegler, "The advanced-step NMPC controller: Optimality, stability and robustness," *Automatica*, vol. 45, no. 1, pp. 86–93, 2009.

[18] A. Kelman, Y. Ma, and F. Borrelli, "Analysis of local optima in predictive control for energy efficient buildings," in *Decision and Control and European Control Conference (CDC-ECC), 2011 50th IEEE Conference on*, Orlando, Florida, 2011, pp. 5125–5130.

[19] C. Long, P. Polisetty, and E. Gatzke, "Nonlinear model predictive control using deterministic global optimization," *Journal of Process Control*, vol. 16, no. 6, pp. 635–643, 2006.

[20] F. Deroo, C. Maier, C. Bohm, and F. Allgöwer, "Offline nmpc for continuous-time systems using sum of squares," in *American Control Conference (ACC), 2011*, San Francisco, California, USA, 2011, pp. 5163–5168.

[21] F. Lydoire and P. Poignet, "Nonlinear model predictive control via interval analysis," in *Decision and Control, 2005 and 2005 European Control Conference. CDC-ECC '05. 44th IEEE Conference on*, Seville, Spain, 2005, pp. 3771–3776.

[22] J. Mercieca and S. G. Fabri, "Particle swarm optimization for nonlinear model predictive control," in *ADVCOMP 2011: The Fifth International Conference on Advanced Engineering Computing and Application in Sciences*, Lisbon, Portugal, 2011, pp. 88–93.

[23] M. Diehl, H. J. Ferreau, and N. Haverbeke, "Efficient numerical methods for nonlinear mpc and moving horizon estimation," in *Nonlinear Model Predictive Control*, ser. Lecture Notes in Control and Information Sciences, L. Magni, D. Raimondo, and F. Allgöwer, Eds. Springer Berlin, 2009, vol. 384, pp. 391–417.

[24] H. Al-Duwaish and W. Naeem, "Nonlinear model predictive control of hammerstein and wiener models using genetic algorithms," in *Control Applications, 2001. (CCA '01). Proceedings of the 2001 IEEE International Conference on*, Mexico City, Mexico, 2001, pp. 465–469.

[25] G. Stein, "Respect the unstable," *IEEE Control Systems Magazine*, pp. 12–25, August 2003.

[26] A. Engelbrecht, *Computational Intelligence: An Introduction.* Chichester, West Sussex PO19 8SQ, England: John Wiley & Sons, 2007.

[27] Q. Zou, J. Ji, S. Zhang, M. Shi, and Y. Luo, "Model predictive control based on particle swarm optimization of greenhouse climate for saving energy consumption," in *World Automation Congress (WAC), 2010*, Kobe, Japan, 2010, pp. 123–128.

[28] H. Stark and Y. Yang, *Vector Space Projections: A Numerical Approach to Signal and Image Processing, Neural Nets, and Optics*, ser. A Wiley-Interscience publication. Wiley: New York, USA, 1998.

[29] S. Boyd and L. Vandenberghe, *Convex Optimization.* New York, NY, USA: Cambridge University Press, 2004.

[30] J. Nocedal and S. Wright, *Numerical Optimization.* Springer New York, 2006.

[31] C. Zălinescu, *Convex Analysis In General Vector Spaces.* World Scientific Publishing Company Incorporated, 2002.

[32] S. Rao, *Engineering Optimization: Theory and Practice.* John Wiley & Sons Hoboken, 2009.

[33] K. Schittkowski, "NLPQL: A fortran subroutine for solving constrained nonlinear programming problems," *Annals of Operations Research*, vol. 5, pp. 485–500, 1985.

[34] E. K. Chong and S. H. Zak, *An Introduction To Optimization, 2nd Edition.* Wiley New York, 2001.

[35] V. Ramos, C. Fernandes, and A. C. Rosa, "Social cognitive maps, swarm collective perception and distributed search on dynamic landscapes," *Brains, Minds and Media - Journal of New Media in Neural an Cognitive Science*, 2005.

[36] C. Hema, M. Paulraj, S. Yaacob, A. Adom, and R. Nagarajan, "Functional link PSO neural network based classification of EEG mental task signals," in *Information Technology, 2008. ITSim 2008. International Symposium on*, vol. 3, Kuala Lumpur, Malaysia, Aug 2008, pp. 1–6.

[37] F. Jiang, M. Frater, and M. Pickering, "Threshold-based image segmentation through an improved particle swarm optimisation," in *Digital Image Computing Techniques and Applications (DICTA), 2012 International Conference on*, Fremantle, Western Australia, Dec 2012, pp. 1–5.

[38] M. R. AlRashidi and M. El-Hawary, "A survey of particle swarm optimization applications in electric power systems," *Evolutionary Computation, IEEE Transactions on*, vol. 13, no. 4, pp. 913–918, Aug 2009.

[39] X. Wang, Y. Wang, H. Zeng, and H. Zhou, "Particle swarm optimization with escape velocity," in *Computational Intelligence and Security, 2006 International Conference on*, vol. 1, Guangzhou, China, Nov 2006, pp. 457–460.

[40] J.-B. Park, Y.-W. Jeong, J.-R. Shin, and K. Lee, "An improved particle swarm optimization for nonconvex economic dispatch problems," *Power Systems, IEEE Transactions on*, vol. 25, no. 1, pp. 156–166, Feb 2010.

[41] J. Maciejowski, *Predictive Control with Constraints*, ser. Pearson Education. Prentice Hall Harlow, England, 2002.

[42] C. Garcia and M. Morari, "Internal model control. a unifying review and some new results," *Industrial & Engineering Chemistry Process Design and Development*, vol. 21, no. 2, pp. 308–323, 1982.

[43] T. J. van den Boom and A. A. Stoorvogel, "Model predictive control," DISC Course, Lecture Notes, 16 January 2010.

[44] D. Mayne, "Control of constrained dynamic systems," *European Journal of Control*, vol. 7, no. 2-3, pp. 87–99, 2001.

[45] S. J. Qin and T. A. Badgwell, "An overview of industrial model predictive control technology," *American Institute of Chemical Engineers Symposium Series*, vol. 93, no. 316, pp. 232–256, 1997.

[46] F. Allgöwer, R. Findeisen, and Z. K. Nagy, "Nonlinear model predictive control: from theory to application," *Journal-Chinese Institute Of Chemical Engineers*, vol. 35, no. 3, pp. 299–316, 2004.

[47] E. F. Camacho and C. Bordons, "Nonlinear model predictive control: An introductory review," in *Assessment and Future Directions of Nonlinear Model Predictive Control*, ser. Lecture Notes in Control and Information Sciences, R. Findeisen, F. Allgöwer, and L. T. Biegler, Eds. Springer Berlin, 2007, vol. 358, pp. 1–16.

[48] M. Werling and D. Liccardo, "Automatic collision avoidance using model-predictive online optimization," in *Decision and Control (CDC), 2012 IEEE 51st Annual Conference on*, Maui, Hawaii, Dec 2012, pp. 6309–6314.

[49] A. Rahideh and M. Shaheed, "Real time nonlinear model predictive control for fast systems," in *Power Electronics Electrical Drives Automation and Motion (SPEEDAM), 2010 International Symposium on*, Pisa, Italy, June 2010, pp. 1732–1737.

[50] R. Lopez-Negrete, F. J. D'Amato, L. T. Biegler, and A. Kumar, "Fast nonlinear model predictive control: Formulation and industrial process applications," *Computers and Chemical Engineering*, vol. 51, pp. 55–64, 2013.

[51] J. Gruber, T. Alamo, D. Ramirez, C. Bordons, and E. Camacho, "A convex approach for nmpc based on second order volterra series models," in *Decision and Control (CDC), 2010 49th IEEE Conference on*, Atlanta , Georgia, USA, 2010, pp. 1336–1341.

[52] M. Čižniar, M. Fikar, and M. Latifi, "Design of constrained nonlinear model predictive control based on global optimisation," in *18th European Symposium on Computer Aided Process Engineering*, ser. Computer Aided Chemical Engineering, B. Braunschweig and X. Joulia, Eds. Elsevier, 2008, vol. 25, pp. 563–568.

[53] K. Dalamagkidis, K. Valavanis, and L. Piegl, "Nonlinear model predictive control with neural network optimization for autonomous autorotation of small unmanned helicopters," *Control Systems Technology, IEEE Transactions on*, vol. 19, no. 4, pp. 818–831, 2011.

[54] Y. Xia and J. Wang, "A recurrent neural network for nonlinear convex optimization subject to nonlinear inequality constraints," *Circuits and Systems I: Regular Papers, IEEE Transactions on*, vol. 51, no. 7, pp. 1385–1394, July 2004.

[55] S. Gros, R. Quirynen, and M. Diehl, "Aircraft control based on fast non-linear mpc & multiple-shooting," in *Decision and Control (CDC), 2012 IEEE 51st Annual Conference on*, Maui, Hawaii, 2012, pp. 1142–1147.

[56] H. Ferreau, G. Lorini, and M. Diehl, "Fast nonlinear model predictive control of gasoline engines," in *Computer Aided Control System Design, 2006 IEEE International Conference on Control Applications, 2006 IEEE International Symposium on Intelligent Control, 2006 IEEE*, Munich, Germany, 2006, pp. 2754–2759.

[57] A. Grancharova, T. A. Johansen, and P. Tϕndel, "Computational aspects of approximate explicit nonlinear model predictive control," in *Assessment and Future Directions of Nonlinear Model Predictive Control*, ser. Lecture Notes in Control and Information Sciences, R. Findeisen, F. Allgöwer, and L. Biegler, Eds. Springer Berlin, 2007, vol. 358, pp. 181–192.

[58] C. Maier, C. Bohm, F. Deroo, and F. Allgöwer, "Predictive control for polynomial systems subject to constraints using sum of squares," in *Decision and Control (CDC), 2010 49th IEEE Conference on*, Atlanta, Georgia, USA, 2010, pp. 3433–3438.

[59] G. Franzé, A. Casavola, D. Famularo, and E. Garone, "An off-line mpc strategy for nonlinear systems based on sos programming," in *Nonlinear Model Predictive Control*, ser. Lecture Notes in Control and Information Sciences, L. Magni, D. Raimondo, and F. Allgöwer, Eds. Springer Berlin, 2009, vol. 384, pp. 491–499.

[60] W. Chen, X. Li, and M. Chen, "Suboptimal nonlinear model predictive control based on genetic algorithm," in *Intelligent Information Technology Application Workshops, 2009. IITAW '09. Third International Symposium on*, 2009, pp. 119–124.

[61] J. Mercieca and S. Fabri, "A metaheuristic particle swarm optimization approach to nonlinear model predictive control," *International Journal On Advances in Intelligent Systems*, vol. 5, no. 3 and 4, pp. 357–369, 2012.

[62] G. Sandou and S. Olaru, "Particle swarm optimization based NMPC: An application to district heating networks," in *Nonlinear Model Predictive Control*, ser. Lecture Notes in Control and Information Sciences, L. Magni, D. M. Raimondo, and F. Allgöwer, Eds. Springer Berlin, 2009, vol. 384, pp. 551–559.

[63] X. Xia, "Nonlinear predictive control based on particle swarm optimization applied to pH neutralization reaction," in *System Science and Engineering (ICSSE), 2012 International Conference on*, Dalian, Liaoning, China, 2012, pp. 445–448.

[64] J. Fan and M. Han, "Nonlinear model predictive control of ball-plate system based on Gaussian particle swarm optimization," in *IEEE Congress on Evolutionary Computation*, Brisbane, Australia, 2012, pp. 1–6.

[65] J.-M. Xiao and X.-H. Wang, "Nonlinear neural network predictive control for power unit using particle swarm optimization," in *Proceedings of the Fifth International Conference on Machine Learning and Cybernetics*, Dalian, China, 2006, pp. 2851–2856.

[66] F. Martinsen and L. T. Biegler, "A new optimization algorithm with application to nonlinear MPC," *Journal of Process Control*, vol. 14, no. 8, pp. 853–865, 2004.

[67] F. Martinsen, L. T. Biegler, and B. A. Foss, "Application of optimization algorithms to nonlinear MPC," in *Proceedings of 15th IFAC World Congress, Barcelona, Spain*, 2002.

[68] R. A. Bartlett, A. Wachter, and L. Biegler, "Active set vs. interior point strategies for model predictive control," in *American Control Conference, 2000. Proceedings of the 2000*, vol. 6, Chicago, Illinois, 2000, pp. 4229–4233.

[69] M. J. Tenny, S. J. Wright, and J. B. Rawlings, "Nonlinear model predictive control via feasibility-perturbed sequential quadratic programming," *Computational Optimization and Applications*, vol. 28, no. 1, pp. 87–121, 2004.

[70] R. Tousain and O. Bosgra, "Efficient dynamic optimization for nonlinear model predictive control-application to a high-density poly-ethylene grade change problem," in *Decision and Control, 2000. Proceedings of the 39th IEEE Conference on*, vol. 1, Sydney, Australia, 2000, pp. 760–765vol.1.

[71] Z. Min and S. Pingping, "Nonlinear model predictive control algorithm based on filter-trust-region method," in *Control Conference (CCC), 2012 31st Chinese*, Hefei, China, 2012, pp. 4069–4074.

[72] P. Potočnik and I. Grabec, "Nonlinear model predictive control of a cutting process," *Neurocomputing*, vol. 43, no. 14, pp. 107–126, 2002.

[73] M. H. Chauhdry and P. B. Luh, "Nested partitions for global optimization in nonlinear model predictive control," *Control Engineering Practice*, vol. 20, no. 9, pp. 869–881, 2012.

[74] A. Helbig, W. Marquardt, and F. Allgöwer, "Nonlinearity measures: definition, computation and applications," *Journal of Process Control*, vol. 10, no. 23, pp. 113–123, 2000.

[75] K. Emancipator and M. H. Kroll, "A quantitative measure of nonlinearity." *Clinical Chemistry*, vol. 39, no. 5, pp. 766–772, 1993.

[76] A. Stack and F. Doyle III, "A measure for control relevant nonlinearity," in *American Control Conference, Proceedings of the 1995*, vol. 3, Seattle, Washington, 1995, pp. 2200–2204.

[77] A. Stack and F. Doyle, "Application of a control-law nonlinearity measure to the chemical reactor analysis," *AIChE Journal*, vol. 43, no. 2, pp. 425–439, 1997.

[78] A. Stack and F. Doyle III, "The optimal control structure: an approach to measuring control-law nonlinearity," *Computers amd Chemical Engineering*, vol. 21, no. 9, pp. 1009–1019, 1997.

[79] T. Schweickhardt, F. Allgöwer, and F. Doyle III, "Nonlinearity quantification for the optimal state feedback controller," in *European Control Conference (ECC), 2003*, Cambridge, UK, September 2003, pp. 4611–4617.

[80] G. T. Tan, M. Huzmezan, and K. E. Kwok, "On measuring closed-loop non-linearity: A topological approach," *The Canadian Journal of Chemical Engineering*, vol. 85, no. 4, pp. 490–505, 2007.

[81] G. T. Tan, M. Huzmezan, and K. Kwok, "On measuring closed-loop nonlinearity - a Vinnicombe metric approach," in *Decision and Control, 2003. Proceedings. 42nd IEEE Conference on*, vol. 6, Maui, Hawaii, 2003, pp. 6163–6168Vol.6.

[82] M. Nikolaou, "When is nonlinear dynamic modeling necessary?" in *American Control Conference, 1993*, San Francisco, California, 1993, pp. 910–914.

[83] K. R. Harris, M. Colantonio, and A. Palazolu, "On the computation of a nonlinearity measure using functional expansions," *Chemical Engineering Science*, vol. 55, no. 13, pp. 2393–2400, 2000.

[84] G. S. Androulakis and M. N. Vrahatis, "OPTAC: a portable software package for analyzing and comparing optimization methods by visualization," *Journal of Computational and Applied Mathematics*, vol. 72, no. 1, pp. 41–62, July 1996.

[85] A. Messac and X. Chen, "Visualizing the optimization process in real-time using physical programming," *Engineering Optimization*, vol. 32, no. 6, pp. 721–747, 2000.

[86] Y.-H. Kim, K. H. Lee, and Y. Yoon, "Visualizing the search process of particle swarm optimization," in *Proceedings of the 11th Annual conference on Genetic and evolutionary computation*. ACM, 2009, pp. 49–56.

[87] T. Binder, L. Blank, H. Bock, R. Bulirsch, W. Dahmen, M. Diehl, T. Kronseder, W. Marquardt, J. Schlöder, and O. Stryk, "Introduction to model based optimization of chemical processes on moving horizons," in *Online Optimization of Large Scale Systems: State of the Art*, M. Grötschel, S. Krumke, and J. Rambau, Eds. Springer Berlin, 2001, pp. 295–340.

[88] O. von Stryk and R. Bulirsch, "Direct and indirect methods for trajectory optimization." *Annals of Operations Research*, vol. 37, no. 1-4, pp. 357–373, 1992.

[89] L. T. Biegler, "An overview of simultaneous strategies for dynamic optimization," *Chemical Engineering and Processing: Process Intensification*, vol. 46, no. 11, pp. 1043–1053, 2007, special Issue on Process Optimization and Control in Chemical Engineering and Processing.

[90] S. Ebbesen, P. Kiwitz, and L. Guzzella, "A generic particle swarm optimization matlab function," in *American Control Conference (ACC), 2012*, Montrèal, Canada, June 2012, pp. 1519–1524.

[91] D. Bratton and J. Kennedy, "Defining a standard for particle swarm optimization," in *IEEE Swarm Intelligence Symposium*, 2007, pp. 120–127.

[92] L. A. Rastrigin, "Extremal control systems," in *Theoretical Foundations of Engineering Cybernetics Series*, Nuaka, Moscow, Russia, 1974, in Russian.