GLOBAL OPTIMIZATION OF DYNAMIC PROCESS SYSTEMS USING COMPLETE SEARCH METHODS

GLOBAL OPTIMIZATION OF DYNAMIC PROCESS SYSTEMS USING COMPLETE SEARCH METHODS

BY

ALI MOHAMMAD SAHLODIN, B.Sc., M.Sc.

A THESIS

SUBMITTED TO THE SCHOOL OF GRADUATE STUDIES AT

MCMASTER UNIVERSITY

IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE DEGREE

OF

DOCTOR OF PHILOSOPHY

DOCTOR OF PHILOSOPHY (2012) MCMASTER UNIVERSITY
(Chemical Engineering) Hamilton, Ontario, Canada

TITLE: GLOBAL OPTIMIZATION OF DYNAMIC PROCESS SYSTEMS USING COMPLETE SEARCH METHODS

AUTHOR: ALI MOHAMMAD SAHLODIN
B.Sc., M.Sc.

SUPERVISORS: Dr. Benoît Chachuat, Dr. Prashant Mhaskar

NUMBER OF PAGES: xv, 168

To my dear wife, Fatemeh, and my beloved parents for their support and love.

Abstract

This thesis considers the problem of finding a global optimum for a dynamic optimization problem with a certification of global optimality. Dynamic optimization problems are often complex due to sever nonlinearity and nonconvexity caused by the participating differential equations. Consequently, globally solving a dynamic optimization problem (a.k.a Global Dynamic Optimization (GDO)) is often difficult, especially when a mathematical proof of global optimality is required. Such a proof can be established by using Spatial Branch-and-Bound (SBB) procedures, whose successful application requires the ability to construct tight bounds for the dynamic optimization model, including the embedded differential equations. This requirement has been a major bottleneck in efficient solution of dynamic optimization problems to global optimality using SBB. The present research aims to address this bottleneck by developing effective bounding techniques for dynamic models, specifically models with ordinary differential equations (ODEs). The focus is on convex relaxations for SBB that are known to be advantageous over interval bounds due to their (generally) tighter enclosures and faster convergence.

In the first part of the thesis, a novel algorithm for constructing convex and concave relaxations of solutions of nonlinear parametric ODEs is developed. This technique builds upon a validated interval method for ODEs, and uses the McCormick relaxation technique for computing the relaxations of state variables. In addition to better convergence properties, the relaxations so obtained are guaranteed to be no looser than their underlying interval bounds, and are typically tighter in practice. Moreover, they are rigorous in a sense that the validated ODE method accounts for truncation errors during ODE integration. Nonetheless, the tightness of the relaxations is affected by the dependency problem of interval arithmetic that is not addressed systematically in the underlying interval ODE method. The dependency problem is a major source of overestimation in interval computations. However, it can be mitigated by Taylor models, which replace interval arithmetic with symbolic computations where possible. The use of Taylor models in validated ODE methods usually yields tighter interval bounds on state variables. Moreover, Taylor models have

better convergence properties than interval arithmetic.

The second part of the thesis considers improving the relaxation algorithm by using a Taylor model ODE method instead of the interval ODE method. This way, the advantages of both Taylor models and convex/-concave relaxations in producing tighter enclosures are combined. Here again, the relaxations are guaranteed to be no looser than the interval bounds obtained from Taylor models, and are usually tighter in practice. Therefore, the algorithm considerably improves upon not only its previous form which is based on interval arithmetic, but also the Taylor model ODE method. Despite their tightness, however, the relaxations obtained by the proposed algorithm are nonlinear and (potentially) nonsmooth. This would hamper their reliable and efficient solution by conventional nonlinear programming techniques.

In the final part of the thesis, the above-mentioned drawbacks are avoided by incorporating polyhedral relaxations into the Taylor model-based relaxation algorithm. By doing so, linear, yet relatively tight, relaxations can be constructed for the dynamic optimization model. The resulting relaxation algorithm along with a SBB procedure is implemented in the MC++ software package in order to build a complete GDO algorithm. For comparisons with previous work, a state-of-the-art GDO algorithm that uses Taylor models only (without relaxations) is also implemented in a similar manner. Case studies on benchmark problems demonstrate the effectiveness of the developments in this thesis in reducing the computational complexity of GDO. In particular, it is shown that the use of relaxations in the proposed GDO algorithm can speed up the convergence by orders of magnitude compared to the previous work, where no relaxations are used.

Acknowledgments

My deepest gratitude goes toward Dr. Benoît Chachuat, whose expertise and original visions together with his excellent supervision played a key role in the success of this research. All the contributions in this research owe a great deal to both his ideas and direct assistance. I would also like to thank my co-supervisor, Dr. Prashant Mhaskar for his kind support and cooperation during the second half of my PhD studies. I am grateful for the constructive discussions that I had with Dr. Christopher L. E. Swartz and Dr. Ned Nedialkov, during my committee meetings. Their invaluable comments and suggestions have definitely improved the quality of this research.

I should also thank McMaster Advanced Control Consortium (MACC) and Department of Chemical Engineering for providing a positive atmosphere and making my stay here a unique experience. I am especially grateful to the administrative staff, in particular, Kathy Goodram, Lynn Falkiner, Nanci Cole, and Melissa Vasil for their help and responsiveness. Also, I would like to acknowledge MACC and School of Graduate Studies for their financial support, which made this research possible.

I am truly grateful to my beloved wife and dear friend, Fatemeh, for her wonderful company, patience, and encouragement which means the world to me. My special appreciation goes out to my caring parents for their sacrifice and endless support to which all my success owes.

Finally, I cannot thank enough merciful God for giving me the opportunity to live and blessing me with all I need to grow both materially and spiritually.

Ali Mohammad Sahlodin
December 16, 2012

Contents

1 Introduction **1**

1.1 State-of-the-Art in Global Optimization of Dynamic Systems 3

1.2 Thesis Outline 6

2 Background **8**

2.1 Problem Formulation 8

2.2 Direct Solution Methods for Dynamic Optimization 11

2.2.1 Simultaneous Approach 12

2.2.2 Sequential Approach 12

2.3 Bounding and Relaxation of Factorable Functions and Programs . . . 13

2.3.1 Interval Analysis 14

2.3.2 Taylor Models 16

2.3.3 Convex and Concave Relaxations 19

2.3.4 Convergence Properties of Bounding and Relaxation Methods 25

2.3.5 Polyhedral Relaxation of Factorable Programs 30

2.4 Bounding Solutions of Parametric ODEs 37

2.4.1 Validated Solution of ODEs using Interval Analysis 38

2.4.2 Validated Solution of ODEs using Taylor Models 44

2.5 Continuous Local and Global Optimization 48

2.5.1 Global and Local Solution 48
2.5.2 Convex Optimization Problem 51
2.5.3 Branch-and-Bound Search for Global Optimization 51

3 Convex/Concave Relaxation of Dynamic Models using Interval-based ODE Methods 58

3.1 Interval Analysis based State Relaxation Algorithm 58
3.1.1 Phase I: A Priori State Convex/Concave Relaxations 59
3.1.2 Phase II: Tight State Convex/Concave Relaxations 66
3.2 Numerical Case Studies 70
3.2.1 A Simple Scalar ODE 71
3.2.2 Lotka-Volterra System 75
3.3 Conclusions 78

4 Convex/Concave Relaxation of Dynamic Models using Taylor Model-based ODE Methods 80

4.1 Taylor Model-based Relaxations 81
4.1.1 Proposed McCormick-Taylor Models 82
4.1.2 McCormick-Taylor Models for Ordinary Differential Equations 85
4.2 Numerical Case Studies 90
4.2.1 Enclosure Quality and Computational Aspects 91
4.2.2 Convergence Order of Taylor Model-based Bounds/Relaxations for ODEs 100
4.3 Conclusions 104

5 Global Dynamic Optimization using Polyhedral Relaxations 107

5.1 GDO Algorithm using Taylor Model-derived Interval Bounds 109
5.1.1 Constraint Handling and Taylor Model-based Domain Reduction 109

5.2 GDO Algorithm using Polyhedral Relaxations from Taylor Models and McCormick-Taylor Models . . . 111
5.2.1 LP-based Domain Reduction . . . 113
5.3 Branch-and-Bound Procedure . . . 115
5.4 Software Implementation of GDO Solver . . . 118
5.4.1 SBB . . . 120
5.4.2 Lower-bounding Solver . . . 121
5.4.3 Upper-bounding Solver . . . 123
5.4.4 Third-party Software . . . 124
5.5 Numerical Case Studies . . . 125
5.5.1 Singular Control Problem . . . 125
5.5.2 Van der Pol Problem . . . 129
5.5.3 Denbigh Problem . . . 131
5.5.4 Flow Control Problem . . . 134
5.5.5 Reversible Reactions Problem . . . 136
5.5.6 Scaling Problem . . . 139
5.6 Conclusions . . . 142

6 Conclusions and Future Work **144**
6.1 Future Work . . . 147

References **149**

A Illustrative Example of MC++ GDO Solver **160**
A.1 Model Setup . . . 160
A.2 Solver Call and Settings . . . 163
A.3 Screen Output . . . 165

List of Tables

2.1 Recursive computation of relaxations and subgradients 24

3.1 Comparison between interval and convex/concave bounds in Problem (3.17), for Taylor series expansion orders $k = 5, 7, 10, 20$ and a constant stepsize of $h = 0.01$ (100 steps). 72

3.2 Comparison between the proposed discretize-then-relax approach and the solvers VNODE-LP [63] and VSPODE [49] in Problem (3.17), for various (absolute and relative) tolerances of stepsize adaptation and a Taylor series expansion order of $k = 10$. 73

3.3 Refined interval bounds for x at $t = 1$ with $N = 0, 1, 2, 5$ and 10 reference points in the scalar ODE problem (3.17). 75

4.1 Comparison of Taylor model-based methods for interval bounds and McCormick relaxations in Problem (3.18,3.19); TM and MCTM stand for the Taylor model and McCormick-Taylor model methods, respectively. 97

4.2 Effect of Taylor series expansion order on the performance of Taylor model-based methods for interval bounds and McCormick relaxations in Problem (3.18,3.19). 97

4.3 Effect of number of variables n_p on performance of Taylor/McCormick-Taylor models of order $q = 6$ (with and without subgradient calculations) in Problems (4.12-4.17). 100

5.1 Global solution and node counts for Problem (5.6) without domain reduction techniques. 127

5.2 CPU time results (seconds) for Problem (5.6) without domain reduction techniques. 127

5.3 Global solution and node counts for Problem (5.6) with applicable domain reduction techniques. 128

5.4 CPU time results (seconds) for Problem (5.6) with applicable domain reduction techniques. 129

5.5 Global solution and node counts for Problem (5.7) with applicable domain reduction techniques. 130

5.6 CPU time results (seconds) for Problem (5.7) with applicable domain reduction techniques. 130

5.7 Global solution and node counts for Problem (5.8) with applicable domain reduction techniques. 132

5.8 CPU time results (seconds) for Problem (5.8) with applicable domain reduction techniques. 132

5.9 Global solution and node counts for Problem (5.9) with applicable domain reduction techniques. 135

5.10 CPU time results (seconds) for Problem (5.9) with applicable domain reduction techniques. 136

5.11 Concentration data used in the parameter estimation problem (5.10). 137

5.12 Nodes and CPU times for global solution of Problem (5.10) with default solver settings. 138

5.13 Nodes and CPU times for global solution of Problem (5.10) with Max_Repeat=0. 138

List of Figures

2.1 Control vector parameterization 12

2.2 The wrapping effect of interval analysis 16

2.3 Illustration of Taylor models. 17

2.4 Convex (left plot) and non-convex (right plot) sets 19

2.5 Convex (left plot) and non-convex (right plot) functions. 20

2.6 Illustration of convex/concave relaxations and bounds. 21

2.7 Interval bounds, McCormick relaxations, and affine relaxations 25

2.8 Image of the function $z(p)$ and the interval inclusion associated with an arbitrary enclosure of $z(p)$. 28

2.9 Convex and concave envelopes for a convex (left plot), concave (center plot), and convexo-concave (right plot) univariate function. 33

2.10 Outer-approximation of a convex function using polyhedral cuts; left plot: bisection rule, right plot: maximum error rule. 34

2.11 Two-phase interval series method. 40

2.12 Minimization of a function with multiple local optima 50

2.13 Spatial branch-and-bound procedure. 53

2.14 The branch-and-bound tree. 54

3.1 Algorithm for convex/concave relaxations of parametric ODEs 60

3.2 A priori and tightened interval bounds for the state variable x in the scalar ODE problem (3.17). 71

3.3 Interval bounds, convex/concave relaxations, and affine relaxations (for $p^{\text{ref}} = 0$) for the state variable x at $t = 1$ in the scalar ODE problem (3.17). 73

3.4 Subgradients of the convex and concave relaxations for the state variable x at $t = 1$ in the scalar ODE problem (3.17). 74

3.5 Refined interval bounds and convex/concave relaxations, with 10 reference points, for the state variable x at $t = 1$ in the scalar ODE problem (3.17); the dotted lines and relaxations correspond to the case without the tightening procedure. 76

3.6 Interval bounds (left plot) and convex/concave bounds at $p = 3$ (right plot) for the state variables x_1 and x_2 in the Lotka-Volterra problem (3.18),(3.19). 77

3.7 Interval and convex/concave bounds for the state variables x_1 (left plot) and x_2 (right plot) at $t = 2$ in the Lotka-Volterra problem (3.18),(3.19). 78

3.8 Convex/concave bounds for the state variables x_1 (left plot) and x_2 (right plot) at $t = 2$ in the modified Lotka-Volterra problem (3.20),(3.21). 79

4.1 McCormick-Taylor model and its McCormick relaxations 85

4.2 Interval bounds after Phase I and Phase II in Problem (4.11). 92

4.3 Taylor and McCormick-Taylor models at $t = 2$ in Problem (4.11) . . . 92

4.4 Interval and convex/concave bounds at $t = 2$ in Problem (4.11) . . . 93

4.5 Subgradients of the convex and concave relaxations derived from the McCormick-Taylor model of x at $t = 2$ in Problem (4.11). 94

4.6 Interval bounds (left plot) and convex/concave bounds at $p = 3$ (right plot) for x_1 and x_2 in Problem (3.18,3.19). MCTM stands for McCormick-Taylor model. 96

4.7 Interval and convex/concave bounds for x_1 (left plot) and x_2 (right plot) at $t = 8$ in the problem (3.18,3.19). 98

4.8 Convex and concave bounds of x_1 (left plot) and x_2 (right plot) at $t = 1$ in Problem (4.12,4.13). 99

4.9 Pointwise and Hausdorff convergence of Taylor model $\mathcal{T}_{x_2(t_f)}(p)$ with interval bounding of polynomial from the solution of the Lotka-Volterra system. The index subscript of x_2 is omitted for notational simplicity. 103

4.10 The range of the polynomial $\mathcal{P}_{x_2}(p)$ versus the width of the variable set. . 104

4.11 Pointwise and Hausdorff convergence of McCormick-Taylor model $\mathcal{MT}_{x_2(t_f)}(p)$ with relaxation of polynomial from the solution of the Lotka-Volterra system. The index subscript of x_2 is omitted for notational simplicity. 105

4.12 Example of monotonic convex/concave relaxations that also reach the function at the variable bounds. 106

5.1 Naive linearization approach 108

5.2 Illustration of the GDO solver. 120

5.3 Effect of order of Taylor/McCormick-Taylor models in global optimization of Problem (5.8) using the three algorithms ($n_p = 3$). 133

5.4 Scaling of nodes and CPU time in global optimization of Problem (5.11) with increasing the number of both states and optimization variables. 140

5.5 Scaling of time for solving the lower-bounding problem in Problem (5.11) using the Taylor model methods; left plot: $q = 4$; right plot: $q = 1$. 142

Listings

A.1 Setup of Problem (5.7) in MC++ 161
A.2 Solver call and settings for global solution of (5.7) using MC++ . . . 163
A.3 MC++ output for Problem (5.7) 168

List of Acronyms

CPT Constraint Propagation on Taylor Models

DAE Differential-Algebraic Equation

GDO Global Dynamic Optimization

HOE High-order Enclosure

IA Interval Analysis

IVP Initial Value Problem

LBS Lower-bounding Solver

LEPUS Local Excess Per Unit Step

LP Linear Programming

MCTM McCormick-Taylor Model

PRMCTM Polyhedral Relaxation of McCormick-Taylor Models

NLP Nonlinear Programming

ODE Ordinary Differential Equation

PRTM Polyhedral Relaxation of Taylor Models

SBB Spatial Branch-and-Bound

TM Taylor Model

UBS Upper-bounding Solver

Chapter1

Introduction

The purpose of a chemical plant is to process raw material and produce some end-products. Chemical processes are designed to operate in continuous, batch, or semi-batch manners. A typical continuous process is identified by a continuous input of raw materials and a continuous output of products. On the other hand, a batch process has no flow into and out of the process unit during its operation. Finally, a semi-batch process is the one which has only a continuous input or a continuous output flow.

In another classification, chemical processes are operated in either steady-state or dynamic fashion. A steady-state operation means that none of the process variables change with time. On the contrary, one or more of the process variables change with time in a dynamic operation. All batch and semi-batch processes are inherently dynamic because they have a start and stop time, during which the raw materials and products evolve with time. Regarding continuous processes, those with cyclic operation, such as pressure-swing adsorption, are also inherently dynamic. Other continuous processes, that are typically run in steady-state, experience dynamic operations during start-ups, shutdowns, and transition periods.

The optimal performance of dynamic (interchangeably transient or unsteady-state) processes subject to process constraints, such as product specifications and safety regulations, is the subject of the *dynamic optimization* area. Practical chemical engineering examples include optimal grade transition in polymerization processes

[23], where an optimal control policy is sought during the transition in order to reduce the amount of off-specification products. Another example is maximizing the production of a certain pharmaceutical product in a batch operation, where dynamic optimization can be employed to obtain optimal design and operating conditions such as the reactor volume, the reaction time, and control trajectories (e.g. flowrate and temperature). Besides transient processes, dynamic optimization can be applied to those steady-state processes in which process variables evolve with respect to a physical dimension. An example of this type is flow through a tubular reactor where concentrations change along the reactor length.

Dynamic optimization presents much more complexity compared to steady-state optimization. This is due to the time-varying behavior of the dynamic process, which must be expressed in the optimization model by (often nonlinear) differential equations. The complexities introduced by this time-varying behavior are threefold. First, a dynamic optimization problem can have time-varying decision variables (e.g. temperature profile in a batch reactor), which lead to an infinite dimensional optimization problem. Second, nonlinear differential equations used to express the time-varying behavior generally have no explicit solution, and their numerical solution can be challenging from computational and theoretical perspectives. Third, the nonlinear differential equations usually cause severe nonlinearity to the optimization model, which in turn can give rise to a highly nonconvex optimization model. As a result of nonconvexity, the dynamic optimization model may exhibit many local solutions; that is, solutions that are optimal only in a feasible neighborhood around them. Multiple local solutions can exist for even small-scale dynamic optimization problems, e.g. optimal temperature or feed rate control in batch reactors [53, 10] and optimal profile of catalyst blend in a tubular reactor [54]. The presence of local solutions impedes finding a global solution, which is optimal over the entire feasible region, by conventional local search techniques. Failure to locate a global solution leads to increased operational and/or capital cost of the plant, which is not desired given the resource limitations and market competitions.

In addition to the optimal performance of dynamic processes, dynamic optimization can be used in model identification using parameter estimation from time-series data. In this application, the objective is to validate a given dynamic model structure based on its fitness to the experimental data. To this end, a dynamic optimization problem can be formulated, where the objective is defined as a measure of error of the calculated model results, based on its estimated parameters, from the data. If the optimal error exceeds a certain threshold, the current model is discarded, and a new model is attempted. Here, it is important to judge the model fitness based on

the best possible fit, which is obtained only from a global solution of the dynamic optimization problem. Failure to find a global solution can lead to misleading results; a good model could be discarded if its poor fit is only due to a local solution.

There are other applications where finding the global solution to a dynamic optimization problem is not only desired, but also a necessity. For example, in bilevel dynamic optimization problems [60], the outer dynamic optimization problem is constrained by the inner one. Therefore, the inner problem must be solved to global optimality in order to ensure the feasibility of the outer one.

Provided the wide application of dynamic optimization and the need for their global solution, a great motivation lies in developing optimization techniques that can *guarantee* locating the global solution to such problems. This is addressed in the context of Global Dynamic Optimization (GDO). Literature on GDO along with motivations for this research is reviewed in the next section.

1.1 State-of-the-Art in Global Optimization of Dynamic Systems

The Spatial Branch-and-Bound (SBB) principle [69] provides a popular framework for finding the global optimum value of a problem with a guarantee of global optimality. In SBB, the idea is to construct lower and upper bounds on the global solution value and refine those bounds by successively branching on the variable space. The procedure is continued until the bounds are tight enough to enclose the global solution value within a given accuracy. For an efficient SBB, it is important to use bounds that tightly enclose the global optimum value and converge to it quickly as the variable space is branched. Over the last few decades, substantial progress has been made in applying SBB to steady-state nonlinear programming (NLP) problems, including development of effective bounding schemes and efficient branch-and-bound procedures. As a result, many large-scale NLP problems can now be solved to global optimality efficiently using related commercial software such as BARON [77, 78]. Nonetheless, the current situation is not the same for global optimization of dynamic problems (i.e. GDO). Due to the embedded differential equations, such problems exhibit much more complexity than steady-state problems. This complexity often gives rise to many local solutions even for simple dynamic optimization problems

[53]. Although all the theoretical results developed for global optimization in general are also valid for GDO (provided a finite number of optimization variables), they are not sufficient. In particular, GDO have faced some theoretical barriers due to the presence of a dynamic model (e.g. ordinary differential equations (ODEs)) in its formulation. A major difficulty here is that no explicit solutions exist for a nonlinear ODE system in general. Rather, ODE systems are solved by a numerical integration procedure. On the other hand, conventional bounding techniques for computing interval bounds or convex and concave relaxations primarily work for functions with explicit representations (see §2.3.1-§2.3.3).

Note that the problem of lack of explicit solutions for ODEs arises only with the sequential approach of dynamic optimization. In this approach, the ODE structure is retained, and therefore, the implicit state variables need to be solved by numerical integration. On the other hand, the simultaneous approach eliminates the need for integration by converting the ODEs into a fully explicit system. Therefore, GDO with conventional bounding and relaxation techniques is readily achievable in the simultaneous framework. As an example, Esposito and Floudas [35] tackled a GDO problem by discretizing the ODEs into explicit equations, and then relaxing them using the αBB (α-based branch-and-bound) relaxation technique [1] [6, 5]. This use of the simultaneous approach for GDO is quite straightforward in that it does not require special treatment of the ODEs. Nevertheless, the theoretical guarantee of convergence to a global optimum can be established for the discretized, algebraic model, which is merely an approximation of the original dynamic model. As a result, GDO using the simultaneous approach may not produce rigorous results.

Esposito and Floudas [35] also initiated early work on GDO using the sequential approach, where they applied the αBB method to convexify state variables during ODE integration. However, choosing a rigorous α value that could prove convexity required a valid interval Hessian matrix of the functions being relaxed, and in turn, valid interval bounds on state variables and their first- and second-order sensitivities. In the absence of an explicit representation for the state variables to derive such bounds, Esposito and Floudas [35] resorted to a sampling scheme to estimate α values. Consequently, they could not establish theoretical guarantee of convexity and global optimality.

Despite the lack of explicit forms, the solutions of parametric ODEs can be still

[1] The αBB method convexifies a general nonconvex expression by adding to it a sufficiently convex quadratic term. A parameter α is used to adjust convexity of the quadratic term so that the convexity of the entire expression is ensured (hence the name αBB).

bounded using some special techniques such as differential inequalities [103]. In differential inequalities, the ODE solutions are bounded by solving two auxiliary ODE systems, which enclose the original ODEs from below and from above. Papamichail and Adjiman [70] and Chachuat and Latifi [26] incorporated differential inequalities in the αBB framework in order to obtain an interval Hessian matrix and compute rigorous α values. Nonetheless, these methods are computationally expensive as they involve bounding first- and second-order sensitivity or adjoint equations [70, 26] and/or introduction of new auxiliary variables and constraints [70]. Moreover, the α values determined this way are usually conservative [82], thereby leading to large overestimation.

Based on the concept of differential inequalities and the McCormick relaxation technique [58], Singer and Barton [95, 96] proposed to construct an auxiliary system of ODEs that describes pointwise-in-time affine relaxations of $\mathbf{x}(t)$ on $\mathbf{P}$. Then, the auxiliary ODE system is solved to yield the affine relaxations at discrete points in time. Recently, the idea of an auxiliary ODE system has been extended to enable nonlinear convex/concave relaxations of state variables [89, 87]. An advantage of these methods is that, unlike the αBB technique, McCormick's technique does not introduce any auxiliary variables. Also, it does not require an interval Hessian matrix, thereby removing the need for bounding first- and second-order sensitivity equations.

The above-mentioned relaxation methods, although are theoretically rigorous, are not computationally rigorous. This is due to the use of standard ODE solvers that do not feature guaranteed bounds on truncation errors, which result from a finite order Taylor series expansion. Consequently, invalid bounds on state variables and in turn on the functions participating in the optimization problem could result. Another limitation is that differential inequalities typically use natural interval extensions [62], which suffer from large overestimation and slow convergence [19]. The overestimation is due to the wrapping effect and dependency problem (see §2.3.1). The accumulation of overestimation typically leads to an explosion of the state bounds, ultimately causing the integration procedure to abort, unless special techniques relying on a priori knowledge of the system are employed [88].

The above limitations are dealt with systematically using so-called verified (also called validated or rigorous) ODE methods (e.g. [51, 13, 65, 49]). These methods bound truncation errors rigorously, and have features to mitigate the overestimations of interval computations. Also, they can be used to enclose the solutions of parametric ODEs. Lin and Stadtherr [47, 48] presented a GDO algorithm, where their validated ODE method [49] was used to compute bounds on the state variables, and in turn, the objective/constraint functions participating in the dynamic

optimization problem. Their ODE method builds upon the one proposed in [65], and features Taylor models [56] to mitigate the dependency problem of interval computations. Due to the typically tight and relatively fast-converging interval bounds enabled by the Taylor model-based ODE method, the GDO algorithm of Lin and Stadtherr [47, 48] presented a major advancement in convergence speed compared to the previous algorithms.

Despite these developments, solving small-scale dynamic optimization problems (having 5-6 variables) in a reasonable time can still be an issue. Practical dynamic optimization problems with a medium to large number of differential equations and optimization variables may not be even tractable. Therefore, the current GDO technology deserves further advancements toward being an effective tool for real-life dynamic optimization problems. This is precisely the motivation for the present research. In particular, the objective is to enhance the efficiency of SBB for GDO by developing algorithms that enable more effective bounds for the dynamic optimization model. Following the previous work, the focus here is on verified ODE methods that have the advantage of producing valid, relatively tight enclosures. However, these methods are designed to produce interval bounds only. On the other hand, it is known that the use of convex and concave relaxations typically greatly enhances convergence speed over simple interval bounds in global optimization [98, 42]. Therefore, this work considers incorporating relaxation techniques into validated ODE methods, thereby helping reduce the computational complexity of SBB for GDO.

1.2 Thesis Outline

The remaining chapters of this thesis are organized as follows.

- In Chapter 2, an overview of preliminaries pertinent to this research is provided, including the problem formulation, dynamic optimization methods, techniques for bounding the range of functions and ODE solutions, and branch-and-bound methods for global optimization. Also, efficiency requirements of branch-and-bound, particularly the need for computing tight bounds on the solution of the optimization problem, are discussed.
- In Chapter 3, a theory is developed for computing convex and concave relaxations of state variables by extending an interval-based method for bounding

solutions of ODEs. Also, a fully automatable procedure is proposed for computing such relaxations using McCormick's relaxation technique. Case studies are performed to demonstrate the advantage of the proposed approach over some other methods in providing tighter bounds on the solutions of ODEs. Also, the limitations of the proposed approach compared to an existing bounding method based on Taylor models are highlighted. This motivates further developments that follow in the next chapters.

- In Chapter 4, it is proposed to combine McCormick's relaxation technique with Taylor models in order to enable tighter convex/concave relaxations for a dynamic model. The combination is achieved in two variants, the effectiveness and computational aspects of each are investigated through case studies.

- In Chapter 5, it is proposed to incorporate polyhedral relaxations into the Taylor model methods presented in Chapter 4 so as to avoid some drawbacks of the nonlinear relaxations. Like McCormick's relaxation of Taylor models, the polyhedral relaxation of Taylor models is presented in two variants. Then, it is embedded into a branch-and-bound procedure to form a GDO algorithm. The software implementation of the algorithm is also presented in this chapter. The performance of the resulting algorithm is studied with a number of examples, and compared with a state-of-the-art algorithm that is only based on Taylor models.

- In Chapter 6, the contributions made in this research are summarized, and possible future work is discussed.

Chapter2

Background

Preliminary materials are covered in this chapter. In §2.1, the dynamic optimization problem of interest is formulated. Common techniques for numerical solution of dynamic optimization problems are outlined in §2.2. In §2.3, several techniques for computing enclosures on factorable functions and programs are presented. It is followed by a description of two methods for bounding solutions of ODEs in §2.4. Definitions and techniques pertinent to local and global optimization in continuous space are reviewed in §2.5.

2.1 Problem Formulation

Some mathematical notations throughout the thesis are as follows. Scalar variables are shown in italics; vector variables appear in boldface; capital letters are used for (closed) interval sets, e.g. $Z := [z^{\mathrm{L}}, z^{\mathrm{U}}]$ such that $\{z \in \mathbb{R} : z^{\mathrm{L}} \leq z \leq z^{\mathrm{U}}\}$. Assume that the following parametric initial value problem (IVP) in ODEs represent the dynamics of the process,

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{p}), \tag{2.1}$$

$$\mathbf{x}(t_0) = \mathbf{h}(\mathbf{p}), \tag{2.2}$$

where t is the independent variable (hereafter time), and $\mathbf{x}(t) \in D \subseteq \mathbb{R}^{n_x}$ with D an open connected set and $\mathbf{p} \in \mathbf{P} \subset \mathbb{R}^{n_p}$ are vectors of state variables and time-invariant parameters, respectively. In the context of dynamic optimization studied in this work, the parameters $\mathbf{p}$ are called *optimization variables* or simply *variables* hereafter. A non-autonomous ODE system, where the time t explicitly appears in the ODE right-hand-side, can be easily written in the above autonomous form by considering t as an additional state variable with its derivative equal to 1. For example, the non-autonomous ODE $\dot{x}(t) = x(t) + tp$ can be written as

$$\begin{aligned}\dot{x}_1(t) &= x_1(t) + x_2(t)p\\ \dot{x}_2(t) &= 1\end{aligned}$$

Now, the dynamic optimization problem is formulated as follows.

$$\begin{aligned}\min_{\mathbf{p}\in\mathbf{P}}\ & \mathcal{J}(\mathbf{p}) = \phi(\mathbf{x}(t_i), \mathbf{p}) \\ \text{s.t.}\ \ & \dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{p}),\\ & \mathbf{x}(t_0) = \mathbf{h}(\mathbf{p}),\\ & \mathbf{g}(\mathbf{x}(t_i), \mathbf{p}) \le 0,\\ & \mathbf{s}(\mathbf{x}(t_i), \mathbf{p}) = 0,\\ & t, t_i \in [t_0, t_\mathrm{f}],\end{aligned} \tag{2.3}$$

where $\mathbf{P} := [\mathbf{p}^\mathrm{L}, \mathbf{p}^\mathrm{U}] \subset \mathbb{R}^{\mathbf{n_p}}$ is an interval vector; $\mathbf{g}$ and $\mathbf{s}$ are constraint vectors of sizes n_g and n_s, respectively. The developments ensued in the thesis require some continuity and differentiability assumptions on the equations involved in (2.3), which will be stated as appropriate (see §2.4.1 and §2.4.2).

Remark 2.1. *Problems with time dependent optimization variables* $\mathbf{p}(t)$*, e.g. optimal control problems, can be approximated by the above optimization form via discretizing* $\mathbf{p}(t)$ *over* t*. See §2.2 for more details.*

Remark 2.2. *Problems where the final time* t_f *is itself an optimization variable can be handled via the above form by replacing* t *with* $t_0 + \tau(t_\mathrm{f} - t_0)$ *where* $0 \le \tau \le 1$*, and adding* t_f *to the vector of optimization variables. This change of variable will set the*

integration lower and upper bounds to 0 *and* 1*, respectively.*

Remark 2.3. *If an integral term such as* $\int_{t_0}^{t_f} \psi(\mathbf{x}(t), \mathbf{p})\, dt$ *occurs in the objective or constraint functions, it can be reformulated as the above form by replacing it with a new state variable. The derivative of the new state variable will be the integrand. For example,* $\int_{t_0}^{t_f} \psi(\mathbf{x}(t), \mathbf{p})\, dt$ *is replaced with* $x_l(t_f)$*, and* $\dot{x}_l = \psi(\mathbf{x}(t), \mathbf{p})$*,* $x_l(t_0) = 0$ *is added to the ODE system.*

Remark 2.4. *Problem* (2.3) *includes point constraints only; that is constraints that must be met only at specific points in time* $t_i \in [t_0, t_f]$*. However, a dynamic optimization problem may also have path constraints, e.g.* $g(\mathbf{x}(t), \mathbf{p}) \leq 0, \forall t \in [t_0, t_f]$*, which must be met over the entire time period. These constraints can be handled by discretizing them into a number of point constraints in* $t \in [t_0, t_f]$ *[28]. A disadvantage of this approach is that a large of number of discretization points may be required to ensure that the constraint remains satisfied over the entire time horizon. Another approach is reformulating the path constraint as a new point constraint, which integrates the constraint violation over time [100]. For example,* $g(\mathbf{x}(t), \mathbf{p}) \leq 0, \forall t \in [t_0, t_f]$*, can be replaced by* $g'(\mathbf{x}(t_f), \mathbf{p}) := \int_{t_0}^{t_f} \max(0, g(\mathbf{x}(t), \mathbf{p}))^{\kappa}\, dt \leq 0$*, where* κ *is usually 1 or 2 [101]. However, a major difficulty with this approach is that the* $\max(.)$ *operator introduces some degrees of non-differentiability into the model. This is particularly not desired for the developments presented in this thesis due to the differentiability requirements for Problem* (2.3)*, as discussed in Chapters 3 and 4. Alternatively, Zhao and Stadtherr [105] proposed to deal with inequality path constraints within their GDO algorithm by rigorously enclosing them over all* $t \in [t_0, t_f]$*.*

The goal of this thesis would be to solve (2.3) to guaranteed global optimality with as least computational effort as possible. A lower computational effort will allow larger dynamic optimization problems to be solved in a reasonable time.

2.2 Direct Solution Methods for Dynamic Optimization

Numerical techniques for solving dynamic optimization problems can be classified as *indirect* and *direct* methods. In indirect or variational methods, such as Pontrayagins Maximum Principle [71], calculus of variations [25] is employed to establish the necessary conditions of optimality for the dynamical system. An advantage of indirect methods is that the profile of time-dependent optimization variables $\mathbf{p}(t)$ needs not be discretized. However, one needs to guess the structure of the optimal solution, a task which may prove difficult and even impossible for practical problems. On the other hand, direct methods remove this need by discretizing the profile of the time-dependent optimization variables. This results in a finite dimensional optimization problem, which can be solved by NLP techniques. As a drawback, however, an optimal solution obtained by direct methods can be suboptimal for the original non-discretized problem.

The discretization of $\mathbf{p}(t)$ is typically performed by piecewise approximation of the $\mathbf{p}(t)$ profile over finite time stages $[t_{k-1}, t_k]$

$$0 = t_0 < t_1 < t_2 < \cdots < t_N = t_{\mathrm{f}}.$$

This approximation, which is called control vector parameterization [100], replaces $\mathbf{p}(t)$ with a number of new optimization variables that are now time-independent, thereby conforming to (2.3). There are various ways of performing the discretization, two of which are illustrated in Fig. 2.1. In the left plot, the $p(t)$ profile is parameterized into piecewise constant variables. In the right plot, the discretization is performed using piecewise affine functions $l_i(t) = a_i t + b_i, t \in [t_{i-1}, t_i]$ for $i = 1, ..., N$. Here, the new optimization variables would be the slopes a_i and intercepts b_i. Note that the time stages need not be evenly spaced, and can be decision variables too.

Now that a finite dimensional problem, as (2.3), is available, direct solution methods can be used. Observe from (2.3) that the values of state variables are needed for evaluation of the objective and constraint functions. The state variables in the direct solution methods can be handled using different approaches; two common approaches are outlined in the following.

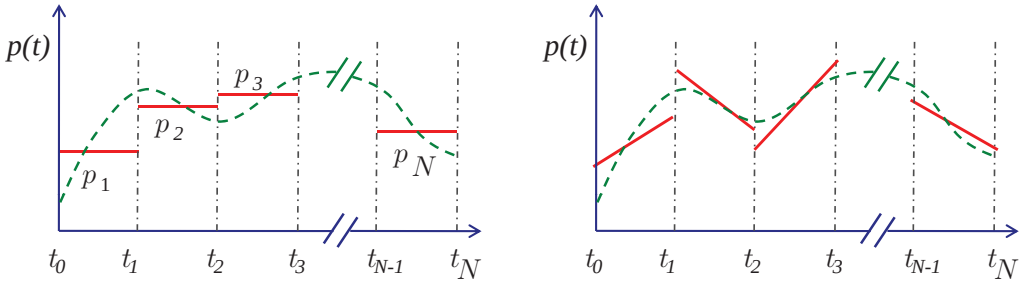

Figure 2.1: Two examples of control vector parameterization: piecewise constant discretization (left plot); piecewise affine discretization (right plot)

2.2.1 Simultaneous Approach

In this approach, also known as full discretization, the continuous state variables are discretized in order to approximate the ODEs by a set of nonlinear algebraic equations [17]. The state discretization is commonly performed using orthogonal collocation on finite elements [16]. Similar to control vector parameterization, this discretization introduces additional optimization variables, thereby leading to a large-scale NLP problem.

In the simultaneous approach, path constraints can be handled easily by enforcing them at the collocation points only. Therefore, they need not be reformulated to point constraints. However, satisfaction of path constraints between two collocation points cannot be guaranteed this way.

Due to the ODE system being converted to algebraic equality constraints, the ODEs are expected to be satisfied at the converged NLP solution only. A downside here is that any results, such as state profiles, obtained at intermediate NLP iterations may have no physical meaning [25]. The fact that the ODE system is not integrated directly makes the simultaneous approach suitable for problems with unstable ODEs, whose solution would terminate prematurely during integration [16].

2.2.2 Sequential Approach

In the sequential approach (also called direct single-shooting approach), the dynamic optimization problem is split in two modules, namely the ODE integration and the

optimization modules. The integration module handles the numerical solution of the parametric ODEs (2.1,2.2) with values of the variables $\mathbf{p}$ provided from the optimization module. The ODEs are often augmented by their sensitivity or adjoint equations [25, 22] to enable accurate derivative information. The state values and their sensitivity or adjoint information obtained from the integration module are sent to the optimization module, where the objective and constraint functions and their gradients are evaluated by the NLP solver. If the optimization module can establish optimality with the given values of the variables, the algorithm is stopped. Otherwise, a new iteration is performed with a new set of variable values sent to the integration module. This sequential procedure is continued until an optimal solution is found.

The sequential approach allows accurate solution of the ODE system using state-of-the-art ODE solvers. However, a downside is that the entire ODEs must be solved at each iteration. This is time-consuming, and may be even unnecessary if the current point is still far away from an optimum [25]. Therefore, the sequential approach may be rather inefficient for large-scale problems. Also, unstable systems are not handled efficiently with this framework due to repeated break-down of ODE integration [16, 22]. This can be mitigated by using the direct multiple-shooting approach by Bock and co-workers [18, 45]

2.3 Bounding and Relaxation of Factorable Functions and Programs

This section is organized as follows. In §2.3.1 and §2.3.2, interval analysis and Taylor models for computing interval bounds on the range of functions are discussed, respectively. In §2.3.3, convex/concave relaxations and McCormick's technique for their computation are presented. Convergence properties of these methods are reviewed in §2.3.4. Finally, the polyhedral relaxation technique for linear relaxation of factorable programs is described in §2.3.5.

2.3.1 Interval Analysis

The closed interval denoted by $[a, b]$ is the set of real variables given by $\{x \in \mathbb{R} : a \leq x \leq b\}$. Throughout this thesis, the term *interval* is understood as *closed interval* and the convention of denoting intervals by capital letters is adopted. The width, the midpoint, and the interior of the interval $X = [a, b]$ are $\mathrm{w}(X) = b - a$, $\mathrm{m}(X) = \frac{1}{2}(a + b)$, and $\mathrm{int}(X) = \{x \in X : a < x < b\}$, respectively.

For an interval vector. the width is defined as the largest of the widths of any of its component intervals, while the mid-point of an interval vector is the vector of the mid-points of its component intervals. Note that vectors are represented in boldface in this thesis, and equalities/inequalities between vector quantities are understood component-wise.

Basic arithmetic (binary) operations between two interval variables X and Y are defined as $X \diamond Y = \{x \diamond y : y \in Y, z \in Z\}$, where $\diamond$ denotes any of the binary operations $+$, $-$, $\times$ or $\div$. Likewise, univariate intrinsic functions (such as exponential, logarithm, inverse, square root, sine and cosine) of interval variables can be defined by treating them as unary operations.

A basis to interval analysis is the notion of *factorable functions* [58]. Factorable functions are those that are defined by a finite recursive composition of binary sums, binary products, and a given library of univariate intrinsic functions. They cover an extremely inclusive class of functions, containing nearly every function which can be represented finitely on a computer by means of a code list or a computational graph. For example, $f = x^2 \sin(x)$ is a factorable function that can be represented as $f = z_1 z_2$ where $z_1 = x^2$ and $z_2 = \sin(x)$.

In interval analysis, *natural interval extensions* [62] can be used for computing bounds on the range of any factorable function given bounds on their variables. An example is given below.

Example. *Consider the function* $f = x_1 x_2 - \frac{x_2}{x_1+1}$ *with* $x_1 \in [1, 2]$ *and* $x_2 \in [-1, 3]$.

The natural interval extension of f is computed as:

$$\begin{aligned} f &\in [1,2]\times[-1,3] - \frac{[-1,3]}{[1,2]+1} = [-2,6] - \frac{[-1,3]}{[2,3]} \\ &= [-2,6] - [-1,3]\times[\frac{1}{3},\frac{1}{2}] = [-2,6] - [-\frac{1}{2},\frac{3}{2}] \\ &= [-2,6] + [-\frac{3}{2},\frac{1}{2}] = [-\frac{7}{2},\frac{13}{2}] \end{aligned}$$

Other, more refined, interval forms can also be applied to obtain an enclosure for a factorable function, including the centered and the mean-value forms. The interested reader is referred to [8, 62] for a thorough discussion on interval analysis.

2.3.1.1 Overestimation in Interval Analysis

In performing interval computations, some overestimation is almost inevitable, which is due to the following reasons.

Dependency Problem. The dependency problem arises from the inability of interval analysis to recognize multiple occurrences of the same variable in a given expression [67, 62]; as a classical example, consider the natural interval extension of the simple expression $x-x$ with $x\in[-1,2]$ which is $[-1,2]-[-1,2]=[-3,3]$, while the actual enclosure set is of course $[0,0]$.

Wrapping effect. The wrapping effect, on the other hand, happens with interval vector computations. Specifically, the actual solution of an interval vector computation can be in any shape, while enclosing it by interval analysis always leads to an interval vector [62]. The wrapping of the actual solution set by such an interval vector will leave some overestimation. For example, consider the vector function $\mathbf{f}: x\to\mathbb{R}^2$ with $f_1(x)=x^2$ and $f_2(x)=-x^2$ and $x\in[0,1]$. The image of x under (f_1,f_2) is simply the red line in Fig. 2.2 ($f_2=-f_1$). However, interval analysis gives $f_1\times f_2\in[0,1]\times[0,1]$, that is an square with an area of 1.

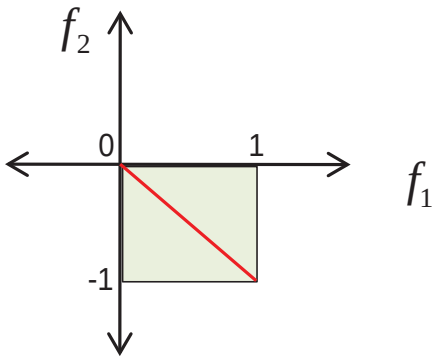

Figure 2.2: The wrapping effect of interval analysis

2.3.2 Taylor Models

Taylor models were introduced to reduce the overestimation in interval analysis, by combining interval arithmetic with symbolic computations [12, 55, 56]. Consider a function $f : \mathbf{P} \to \mathbb{R}$ defined on the set $\mathbf{P} \subset \mathbb{R}^{n_p}$, and let there be an n_p-variate polynomial $\mathcal{P}_f$ of order q and an interval $R_f := [r_f^{\mathrm{L}}, r_f^{\mathrm{U}}]$ such that:

$$f(\mathbf{p}) \in \mathcal{P}_f(\mathbf{p}) + R_f, \quad \text{for each } \mathbf{p} \in \mathbf{P}.$$

Then, $\mathcal{T}_f := \mathcal{P}_f + R_f$ is called a qth-order Taylor model of f on $\mathbf{P}$. In this context, $\mathbf{P}$ and R_f are known as the *domain interval* and the *remainder interval* of $\mathcal{T}_f$, respectively. The polynomial $\mathcal{P}_f$ is obtained from a qth-order Taylor series expansion of f in terms of $\mathbf{p}$, and the remainder interval R_f is obtained from bounding the truncated terms using the Taylor theorem [49].

The Taylor model $\mathcal{T}_f$ of a univariate function f on the interval $[p^{\mathrm{L}}, p^{\mathrm{U}}]$ is depicted in Fig. 2.3; observe, in particular, that $\mathcal{T}_f$ encloses f between two hypersurfaces.

Taylor models can be written in the equivalent so-called centered form, where the remainder interval is centered around zero. The notation $\mathcal{T}_f^{\mathsf{C}}$ is used to indicate a centered Taylor model of f, which is obtained as $\mathcal{P}_f^{\mathsf{C}} + R_f^{\mathsf{C}}$ with $\mathcal{P}_f^{\mathsf{C}} = \mathcal{P}_f + \mathrm{m}(R_f)$ and $R_f^{\mathsf{C}} = R_f - \mathrm{m}(R_f)$.

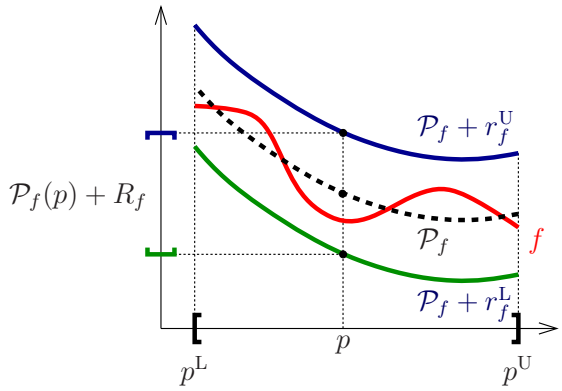

Figure 2.3: Illustration of Taylor models.

Taylor Model Arithmetic Taylor model arithmetic are defined in such a way to minimize interval computations, and instead use polynomials that preserve parametric dependency. Specifically, in computations that involve a Taylor model, the polynomial part is propagated by symbolic calculations wherever possible. On the other hand, the interval remainder term and all polynomial terms of order higher than q are processed according to the rules of interval arithmetic. Taylor model arithmetic has been formalized by Berz and coworkers [12, 56]. The rules for binary sum, binary product and univariate composition of Taylor models lend themselves naturally to automation within a computer program.
For illustration, consider two real-valued functions f_1 and f_2 defined on $\mathbf{P} \subset \mathbb{R}^{n_p}$, and suppose that $\mathcal{T}_{f_1}$ and $\mathcal{T}_{f_2}$ are qth-order Taylor models of f_1 and f_2, respectively, on $\mathbf{P}$. A qth-order Taylor model on $\mathbf{P}$ of $f_1 + f_2$ is easily obtained as $\mathcal{T}_{f_1+f_2} = \mathcal{T}_{f_1} + \mathcal{T}_{f_2}$, so that:

$$\mathcal{T}_{f_1+f_2}(\mathbf{p}) = \mathcal{P}_{f_1}(\mathbf{p}) + \mathcal{P}_{f_2}(\mathbf{p}) + R_{f_1} + R_{f_2}, \quad \text{for each } \mathbf{p} \in \mathbf{P}.$$

Binary product and univariate composition operations are more involved. For instance, the binary product between two qth-order Taylor models,

$$\mathcal{T}_{f_1} \times \mathcal{T}_{f_2} = \mathcal{P}_{f_1} \times \mathcal{P}_{f_2} + \mathcal{P}_{f_1} \times R_{f_1} + R_{f_1} \times \mathcal{P}_{f_2} + R_{f_1} \times R_{f_2}$$

yields a polynomial of order $2q$, while a Taylor model of order q is desired. Therefore, the polynomial $\mathcal{P}_{f_1} \times \mathcal{P}_{f_2}$ is written as $\mathcal{P}_{f_1 \times f_2} + \mathcal{P}_{\mathcal{H}_{f_1 \times f_2}}$, where $\mathcal{P}_{\mathcal{H}_{f_1 \times f_2}}$ contains higher

order terms, which will be included in the remainder term. As a result, the remainder term of the product is obtained as

$$R_{f_1 \times f_2} = \mathcal{P}_{\mathcal{H}_{f_1 \times f_2}} + \mathcal{P}_{f_1} \times R_{f_2} + \mathcal{P}_{f_2} \times R_{f_1} + R_{f_1} \times R_{f_2}.$$

Bounding the polynomial terms in the above expression makes some overestimation inevitable in computing the resulting Taylor model $\mathcal{T}_{f_1 \times f_2}$. In particular, $\{\mathcal{T}_{f_1 \times f_2}(\mathbf{p}) : \mathbf{p} \in \mathbf{P}\} \supseteq \{\mathcal{T}_{f_1}(\mathbf{p}) \times \mathcal{T}_{f_2}(\mathbf{p})\}$, and the equality does not hold in general.

Taylor models for the inverse and other intrinsic functions have also been developed (see e.g. [57]).

Taylor Model Range Bounder The *range* of a Taylor model $\mathcal{T}_f$ on $\mathbf{P}$ is the set $\{\mathcal{P}_f(\mathbf{p}) + R_f : \mathbf{p} \in \mathbf{P}\}$, which turns out to be an interval. Unfortunately, exact range bounding (a.k.a. united extension) of a multivariate polynomial is NP-hard, so one has to resort to some kind of over-approximation methods. Besides those applications that require an enclosure of the range of a function, an interval evaluation of the Taylor form is also necessary to compute binary product and univariate composition operations in Taylor model arithmetic. Therefore, the quality of the selected polynomial range bounder directly affects the quality of the computed Taylor models, and the use of naive interval evaluation usually produces poor results.
Various bounding schemes have been investigated in the literature [68, 56], which for the most part concentrate on the exact bounding of a polynomial's first- and second-order terms. The range bounder used in this work is the one proposed by [49], whereby only the first-order and the diagonal second-order terms are considered for exact bounding, and the remaining terms are directly evaluated via interval analysis. To do so, the polynomial is reformulated as follows.

$$\begin{aligned} \mathcal{P}_f(\mathbf{p}) &= \sum_{i=1}^{n_p} \left[a_i (p_i - p_i^0)^2 + b_i (p_i - p_i^0) \right] + \mathcal{R}(\mathbf{p}) \\ &= \underbrace{\sum_{i=1}^{n_p} \left[a_i \left(p_i - p_i^0 + \frac{b_i}{2a_i} \right)^2 - \frac{b_i^2}{4a_i} \right]}_{=: \mathcal{Q}(\mathbf{p})} + \mathcal{R}(\mathbf{p}), \end{aligned} \tag{2.4}$$

for some $\mathbf{p}^0 \in \mathbf{P}$, and with $\mathcal{R}$ a multivariate polynomial that contains the same terms as $\mathcal{P}_f$ but its first- and diagonal second-order terms. The term $\mathcal{Q}(\mathbf{p})$ can be bounded exactly as it does not have the dependency problem anymore. An interval bound of $\mathcal{P}_f$ on $\mathbf{P}$ is then obtained as:

$$\mathcal{P}_f(\mathbf{P}) = \mathcal{Q}(\mathbf{P}) + \mathcal{R}(\mathbf{P}),$$

In order to prevent division by zero in $\mathcal{Q}$, the rearrangement in (2.4) is considered only when $|a_i| \geq \epsilon$, where ϵ is a small positive number; otherwise, the range bounding of $\mathcal{P}_f$ is performed without the rearrangement in (2.4). With this compromise approach, the exact bounding of quadratic forms—which exhibits worst-case exponential complexity in the number of variables—is avoided.

2.3.3 Convex and Concave Relaxations

Convex and concave relaxations provide another way of computing tighter bounds on the range of functions than simple interval bounds. Prerequisite to the definition of convex/concave relaxations are convex sets and functions, which are defined first.

Definition 2.1 (**Convex set**). *A set $C \subset \mathbb{R}^n$ is said to be convex if for every point $x, y \in C$, the line segment $l := \beta x + (1-\beta)y$, $\forall \beta \in [0,1]$ also remains entirely in C [21].*

Figure 2.4 illustrates graphical representation of a convex (left plot) and a non-convex (right plot) set.

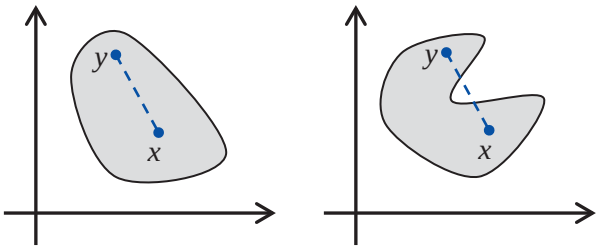

Figure 2.4: Convex (left plot) and non-convex (right plot) sets

Definition 2.2 (**Convex/concave functions**). *Given a convex set $C \subset \mathbb{R}^n$, a function $f : C \to \mathbb{R}$ is said to be convex if*

$$f(\beta x + (1-\beta)y) \leq \beta f(x) + (1-\beta)f(y), \tag{2.5}$$

for each $x, y \in C$ and each $\beta \in [0,1]$ [25]. Similarly, a function $f : C \to \mathbb{R}$ is said to be concave if the reverse of the above inequality holds.

In geometrical terms, a line connecting any two points on a convex function will remain on or above the function. On the other hand, a line connecting any two points on a non-convex function will be entirely or partly below the function. See Fig. 2.5 for a graphical representation.

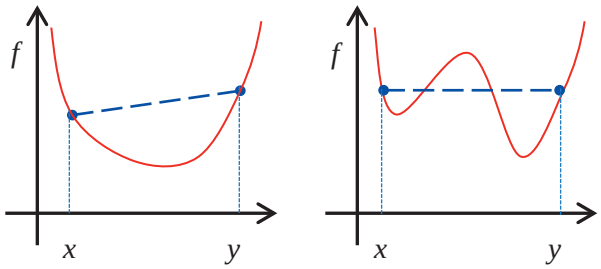

Figure 2.5: Convex (left plot) and non-convex (right plot) functions.

Now, convex/concave relaxations are defined below.

Definition 2.3 (Convex/concave relaxations). *Given a convex set $\mathbf{P} \subset \mathbb{R}^{n_p}$ and a function $f : \mathbf{P} \to \mathbb{R}$, a convex function $f^{\mathrm{cv}} : \mathbf{P} \to \mathbb{R}$ is called a* convex relaxation *of f on $\mathbf{P}$ if $f^{\mathrm{cv}}(\mathbf{p}) \leq f(\mathbf{p})$ for all $\mathbf{p} \in \mathbf{P}$; also, a concave function $f^{\mathrm{cc}} : \mathbf{P} \to \mathbb{R}$ is called a* concave relaxation *of f on $\mathbf{P}$ if $f^{\mathrm{cc}}(\mathbf{p}) \geq f(\mathbf{p})$ for all $\mathbf{p} \in \mathbf{P}$.*

In Fig. 2.6, the function f, a convex relaxation f^{cv} and a concave relaxation f^{cc} are represented in red, green and blue, respectively. Unlike intervals bounds, such as f^{L} and f^{U}, convex and concave relaxations are not generally constant on

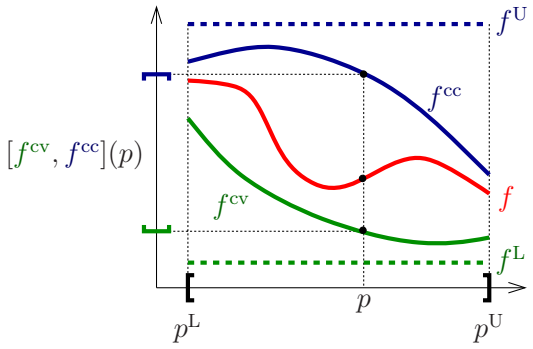

Figure 2.6: Illustration of convex/concave relaxations and bounds.

$\mathbf{P}$. The values of convex and concave relaxations of f at a given $\mathbf{p} \in \mathbf{P}$ are called *convex/concave bounds at* $\mathbf{p}$ and are denoted by $[f^{\mathrm{cv}}(\mathbf{p}), f^{\mathrm{cc}}(\mathbf{p})]$ or $[f^{\mathrm{cv}}, f^{\mathrm{cc}}](\mathbf{p})$.

For factorable functions, the construction of convex and concave bounds is possible through several methods (e.g. [58, 6, 97]). The focus in this research is on McCormick's relaxations [58] and their recent generalizations [90].

2.3.3.1 McCormick's Relaxation Technique

Similar to Taylor model arithmetic, the binary addition and binary multiplication operations along with composition with univariate intrinsic functions can be defined for convex/concave bounds. While the convex (concave) relaxation of a binary sum is trivially the sum of the convex (concave) relaxations of the two operands, the rules for the relaxation of binary products and univariate compositions are more elaborate. In general, this requires that, not only convex/concave bounds, but also interval bounds be available for the operands [58]. The rule for relaxation of binary products is used extensively in this thesis and is recalled subsequently; the rule for the relaxation of univariate compositions can be found in [58].
Consider two real-valued functions f_1 and f_2 defined on the convex set $\mathbf{P}$, and suppose that $[f_1^{\mathrm{L}}, f_1^{\mathrm{U}}]$ and $[{f_1}^{\mathrm{cv}}, {f_1}^{\mathrm{cc}}](\mathbf{p})$ are interval bounds and convex/concave bounds

for f_1, respectively, and $[f_2^{\rm L}, f_2^{\rm U}]$ and $[f_2{}^{\rm cv}, f_2{}^{\rm cc}](\mathbf{p})$ are the same for f_2. Then, convex/concave bounds for $g = f_1 \times f_2$ at $\mathbf{p}$ are given by:

$$\begin{aligned}[g^{\rm cv}, g^{\rm cc}](\mathbf{p}) = \big[&\max\{\alpha_1(\mathbf{p}) + \alpha_2(\mathbf{p}) - f_1^{\rm L} f_2^{\rm L}, \beta_1(\mathbf{p}) + \beta_2(\mathbf{p}) - f_1^{\rm U} f_2^{\rm U}\},\\ &\min\{\gamma_1(\mathbf{p}) + \gamma_2(\mathbf{p}) - f_1^{\rm U} f_2^{\rm L}, \delta_1(\mathbf{p}) + \delta_2(\mathbf{p}) - f_1^{\rm L} f_2^{\rm U}\}\big]\end{aligned}$$

where the intermediate functions $\alpha_1, \alpha_2, \beta_1, \beta_2, \gamma_1, \gamma_2, \delta_1, \delta_2$ are defined as

$$\begin{aligned}
\alpha_1(\mathbf{p}) &= \min\{f_2^{\rm L} f_1{}^{\rm cv}(\mathbf{p}), f_2^{\rm L} f_1{}^{\rm cc}(\mathbf{p})\}, & \alpha_2(\mathbf{p}) &= \min\{f_1^{\rm L} f_2{}^{\rm cv}(\mathbf{p}), f_1^{\rm L} f_2{}^{\rm cc}(\mathbf{p})\}\\
\beta_1(\mathbf{p}) &= \min\{f_2^{\rm U} f_1{}^{\rm cv}(\mathbf{p}), f_2^{\rm U} f_1{}^{\rm cc}(\mathbf{p})\}, & \beta_2(\mathbf{p}) &= \min\{f_1^{\rm U} f_2{}^{\rm cv}(\mathbf{p}), f_1^{\rm U} f_2{}^{\rm cc}(\mathbf{p})\}\\
\gamma_1(\mathbf{p}) &= \max\{f_2^{\rm L} f_1{}^{\rm cv}(\mathbf{p}), f_2^{\rm L} f_1{}^{\rm cc}(\mathbf{p})\}, & \gamma_2(\mathbf{p}) &= \max\{f_1^{\rm U} f_2{}^{\rm cv}(\mathbf{p}), f_1^{\rm U} f_2{}^{\rm cc}(\mathbf{p})\}\\
\delta_1(\mathbf{p}) &= \max\{f_2^{\rm U} f_1{}^{\rm cv}(\mathbf{p}), f_2^{\rm U} f_1{}^{\rm cc}(\mathbf{p})\}, & \delta_2(\mathbf{p}) &= \max\{f_1^{\rm L} f_2{}^{\rm cv}(\mathbf{p}), f_1^{\rm L} f_2{}^{\rm cc}(\mathbf{p})\}.
\end{aligned}$$

In sum, McCormick relaxations of factorable functions are constructed by the recursive application of rules for the range bounding *and* for the convex/concave relaxation of univariate composition, binary multiplication, and binary addition, without the introduction of auxiliary variables. The compact notation $\mathcal{M}_f(\mathbf{p}) := \big\{[f^{\rm L}, f^{\rm U}], [f^{\rm cv}, f^{\rm cc}](\mathbf{p})\big\}$ denotes the pair of interval bounds and convex/concave bounds in the McCormick relaxation of a function f at a point $\mathbf{p} \in \mathbf{P}$.

Generalized McCormick Relaxations. In addition to using the McCormick technique for constructing convex/concave relaxations of factorable functions, the methodology described in this research requires convex and concave relaxations of composite functions of the form $\phi(\boldsymbol{\xi}(\mathbf{p}))$, where the outer function ϕ is factorable, but not the inner function $\boldsymbol{\xi}$. Such relaxations cannot be obtained by applying the standard McCormick technique. Provided that convex and concave bounds (along with interval bounds) are known for $\boldsymbol{\xi}$ on $\mathbf{P}$,

$$\begin{aligned}
&\boldsymbol{\xi}^{\rm cv}(\mathbf{p}) \le \boldsymbol{\xi}(\mathbf{p}) \le \boldsymbol{\xi}^{\rm cc}(\mathbf{p}), \text{ for each } \mathbf{p} \in \mathbf{P},\\
&\boldsymbol{\xi}^{\rm cv} \text{ convex on } \mathbf{P}; \quad \boldsymbol{\xi}^{\rm cc} \text{ concave on } \mathbf{P},
\end{aligned}$$

the concept of *generalized McCormick relaxations* recently introduced by [90] provides a framework for the recursive computation of convex and concave bounds of $\phi(\boldsymbol{\xi}(\cdot))$

on $\mathbf{P}$, in the following form:

$$\phi^{\mathrm{cv}}\left(\boldsymbol{\xi}^{\mathrm{cv}}(\mathbf{p}),\boldsymbol{\xi}^{\mathrm{cc}}(\mathbf{p})\right)\leq\phi(\boldsymbol{\xi}(\mathbf{p}))\leq\phi^{\mathrm{cc}}\left(\boldsymbol{\xi}^{\mathrm{cv}}(\mathbf{p}),\boldsymbol{\xi}^{\mathrm{cc}}(\mathbf{p})\right),$$

where $\phi^{\mathrm{cv}}\left(\boldsymbol{\xi}^{\mathrm{cv}}(\cdot),\boldsymbol{\xi}^{\mathrm{cc}}(\cdot)\right)$ is convex on $\mathbf{P}$, and $\phi^{\mathrm{cc}}\left(\boldsymbol{\xi}^{\mathrm{cv}}(\cdot),\boldsymbol{\xi}^{\mathrm{cc}}(\cdot)\right)$ is concave on $\mathbf{P}$. A key property of the generalized McCormick relaxations is that the convex/concave bounds are at least as tight as the underlying interval bounds [90]. Generalized McCormick relaxations are considered throughout this thesis, and the qualifier 'generalized' is omitted for brevity.

In general, McCormick relaxations are nonsmooth; therefore, affine relaxations from subgradients are usually considered in branch-and-bound procedures.

Definition 2.4 (**Subgradient**). *Let $\mathbf{P}\subset\mathbb{R}^{n_p}$ be a nonempty convex set, let $f^{\mathrm{cv}}:\mathbf{P}\to\mathbb{R}$ be convex, and $f^{\mathrm{cc}}:\mathbf{P}\to\mathbb{R}$ be concave. A vector $\boldsymbol{\sigma}^{\mathrm{cv}}_{\underline{\mathbf{p}}}\in\mathbb{R}^{n_p}$ is called a subgradient of f^{cv} at $\underline{\mathbf{p}}$ if $f^{\mathrm{cv}}(\mathbf{p})\geq f^{\mathrm{cv}}(\underline{\mathbf{p}})+(\boldsymbol{\sigma}^{\mathrm{cv}}_{\underline{\mathbf{p}}})^{T}(\mathbf{p}-\underline{\mathbf{p}})$ for all $\mathbf{p}\in\mathbf{P}$. A vector $\boldsymbol{\sigma}^{\mathrm{cc}}_{\underline{\mathbf{p}}}\in\mathbb{R}^{n_p}$ is called a subgradient of f^{cc} at $\underline{\mathbf{p}}$ if $f^{\mathrm{cc}}(\mathbf{p})\leq f^{\mathrm{cc}}(\underline{\mathbf{p}})+(\boldsymbol{\sigma}^{\mathrm{cc}}_{\underline{\mathbf{p}}})^{T}(\mathbf{p}-\underline{\mathbf{p}})$ for all $\mathbf{p}\in\mathbf{P}$.*

Existence of subgradients in the interior of $\mathbf{P}$ is guaranteed, and for differentiable convex and concave functions, the unique subgradient is the gradient [40].

The systematic construction of subgradients for the McCormick relaxations of factorable functions has been described in [61]. Similar to the convex/concave relaxations, this construction relies on the recursive application of few rules, namely the calculation of subgradients for addition, multiplication and composition operations. This procedure remains applicable with the generalized McCormick relaxations. Subgradients at interior points can be calculated for any factorable function for which a (generalized) McCormick relaxation exists.

Definition 2.5 (**Affine Relaxation**). *Given a convex set $\mathbf{P}\subset\mathbb{R}^{n_p}$ and a function $f:\mathbf{P}\to\mathbb{R}$, let $f^{\mathrm{cv}}(\underline{\mathbf{p}})$ and $f^{\mathrm{cc}}(\underline{\mathbf{p}})$ be respectively convex and concave bounds on f at $\underline{\mathbf{p}}\in\mathbf{P}$ with their corresponding subgradients $\boldsymbol{\sigma}^{\mathrm{cv}}_{\underline{\mathbf{p}}}$ and $\boldsymbol{\sigma}^{\mathrm{cc}}_{\underline{\mathbf{p}}}$. The linear functions $f^{\mathrm{cv,l}}(\mathbf{p})=f^{\mathrm{cv}}(\underline{\mathbf{p}})+(\boldsymbol{\sigma}^{\mathrm{cv}}_{\underline{\mathbf{p}}})^{T}(\mathbf{p}-\underline{\mathbf{p}})$ and $f^{\mathrm{cc,l}}(\mathbf{p})=f^{\mathrm{cc}}(\underline{\mathbf{p}})+(\boldsymbol{\sigma}^{\mathrm{cc}}_{\underline{\mathbf{p}}})^{T}(\mathbf{p}-\underline{\mathbf{p}})$ are called, respectively, an affine relaxation from below and an affine relaxation from above on*

$f(\mathbf{p})$ *for all* $\mathbf{p} \in \mathbf{P}$.

The choice of $\underline{\mathbf{p}}$ can greatly affect how closely $f^{\mathrm{cv,l}}$ and $f^{\mathrm{cc,l}}$ bound f. In the case that f^{cv} or f^{cc} is not differentiable at $\underline{\mathbf{p}}$, the choice of subgradient also affects the affine bound.

McCormick relaxations are illustrated in the following example, where the MC++ software (see §5.4) is used for automatic computation of the results.

Example. *Consider the non-convex function* $f(p) = pe^{-p^2}$, *with* $p \in [-1, 1]$. *Interval bounds as well as convex/concave bounds and their subgradients are desired at* $\underline{p} = 0.5$. *The function is factored as follows:*

$$\begin{aligned} &z_1 = p, \quad z_2 = z_1^2, \quad z_3 = -z_2 \quad z_4 = e^{z_3}, \quad z_5 = z_1 z_4, \\ &f(p) = z_5. \end{aligned} \tag{2.6}$$

Interval bounds on the factors z_i *are computed using interval arithmetic [62], while convex/concave bounds are obtained by the McCormick technique. Finally, the subgradients are calculated using the forward automatic differentiation technique [38]. The recursive application of these techniques on the binary products and univariate compositions in* (2.6) *results in bounds, relaxations, and subgradients for* $f(p)$. *These results are given in Table 2.1.*

z_i	$[z_i^L, z_i^U]$	$[z_i^{cv}, z_i^{cc}](\underline{p})$	$\boldsymbol{\sigma}_{\underline{p}}^{\mathrm{cv}}$	$\boldsymbol{\sigma}_{\underline{p}}^{\mathrm{cc}}$
z_1	[-1.00, 1.00]	[0.50, 0.50]	1.00	1.00
z_2	[0.00, 1.00]	[0.25, 1.00]	1.00	0.00
z_3	[-1.00, 0.00]	[-1.00,-0.25]	0.00	-1.00
z_4	[0.37, 1.00]	[0.37, 0.84]	0.00	-0.63
z_5	[-1.00, 1.00]	[-0.13, 0.66]	1.00	-0.26

Table 2.1: Recursive computation of interval bounds, convex/concave bounds, and subgradients using MC++.

If the relaxation procedure is repeated for many $p \in [-1, 1]$, convex/concave relaxations can be plotted over the entire variable domain as shown in Fig. 2.7. Observe that the relaxations are not smooth at $p = 0$. This is due to the use of min, max, and mid[1] *operators by the McCormick technique. Also plotted in Fig. 2.7 are the affine relaxations obtained from the values of convex/concave bounds and their subgradients at $\underline{p} = 0.5$.*

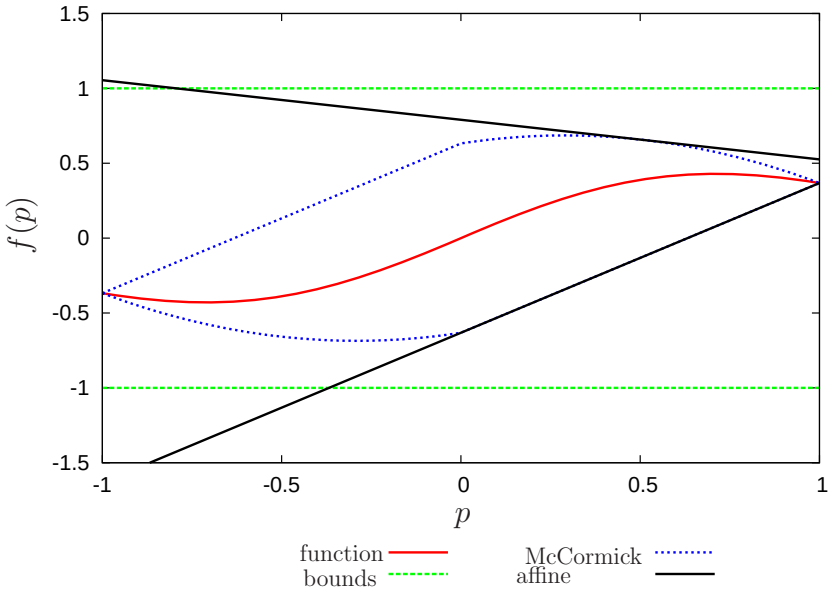

Figure 2.7: Interval bounds, McCormick relaxations, and affine relaxations for $f(p)$.

2.3.4 Convergence Properties of Bounding and Relaxation Methods

In addition to tightness, the convergence of a SBB procedure is influenced by the convergence rate of the underlying enclosures [85]; that is, how fast an approximate enclosure converges to the exact solution set (or equivalently the approximation

[1]Middle value of three numbers in magnitude.

error shrinks to zero) as the variable range shrinks. Bompadre and Mitsos [19] conducted a theoretical study of the convergence order of McCormick relaxations by considering the propagation of convergence orders through the binary addition, multiplication, and function composition in McCormick's technique [58]. Provided convergence orders of the factors, they derived lower bounds on the convergence order of the result of the McCormick operations. Recently, Bompadre et al. [20] carried out similar studies for the convergence of Taylor models with focus on the remainder term[2]. In the following, a number of definitions pertinent to the discussion of convergence orders are reviewed. Then, some of the main results on convergence orders of the bounding and relaxation techniques presented in §2.3 are summarized. The definitions and results presented in this subsection are extracted from [19, 20].

Definition 2.6 (Image and Inclusion Function)**.** *Consider a continuous function $z : Y \to \mathbb{R}$ with $Y \subset \mathbb{R}^n$. The image of an interval $\mathbf{P} \subset Y$ under z is indicated by the interval $\mathcal{I}_z(\mathbf{P}) := [\underline{z}, \overline{z}]$, where $\underline{z}$ and $\overline{z}$ are the exact lower and upper bounds on z over $\mathbf{P}$, respectively. Also, an interval-valued function $\mathcal{F}_z : Y \to \mathbb{R}$ is called an inclusion function of z on Y, if $\mathcal{F}_z(\mathbf{P}) := [z^{\mathrm{L}}, z^{\mathrm{U}}] \supset \mathcal{I}_z(\mathbf{P})$ for any interval $\mathbf{P} \subset Y$ [15, 19].*

Natural interval extension [62] is an example of inclusion functions. It is noted that the values of $\underline{z}$, $\overline{z}$, z^{L}, and z^{U} depend on the interval $\mathbf{P}$. However, this dependence is omitted for notation simplicity. An approximation error for an inclusion function can be defined as

$$E := \mathrm{w}\left(\mathcal{F}_z(\mathbf{P})\right) - \mathrm{w}\left(\mathcal{I}_z(\mathbf{P})\right), \tag{2.7}$$

or equivalently as

$$E := \max\{|z^{\mathrm{U}} - \overline{z}|, |z^{\mathrm{L}} - \underline{z}|\}. \tag{2.8}$$

The range order of an inclusion function is defined as follows.

[2]Bompadre et al. [20] also studied convergence of McCormick-Taylor models, which have been introduced in this research. However, the discussion of convergence of McCormick-Taylor models is postponed to §4.2.2.

Definition 2.7 (Range Order)**.** *An approximation of the continuous function* $z : Y \subset \mathbb{R}^n \to \mathbb{R}$ *with the inclusion function* $\mathcal{F}_z$ *on* Y *has range of order* $\alpha > 0$ *at a point* $\mathbf{p} \in Y$ *if there exists a constant* $\zeta \geq 0$ *for which the inequality*

$$\mathrm{w}\left(\mathcal{F}_z(\mathbf{P})\right) \leq \zeta\, \mathrm{w}\left(\mathbf{P}\right)^{\alpha} \tag{2.9}$$

holds for every interval $\mathbf{P} \subset Y$ *with* $\mathbf{p} \in \mathbf{P}$ *[20].*

The range order can also be defined for the function z itself, in which case $\mathcal{F}_z(\mathbf{P})$ in (2.9) is replaced by the image $\mathcal{I}_z(\mathbf{P})$. Also, an inclusion function (or a function) has a range of order $\alpha > 0$ on Y if it has the range order of (at least) α for every $\mathbf{p} \in Y$, with ζ independent of $\mathbf{p}$.

Definition 2.8 (Scheme of estimators)**.** *Let* $z : Y \to \mathbb{R}$ *be a function with* $Y \subset \mathbb{R}^n$ *a nonempty, convex set. Also, assume that for any interval* $\mathbf{P} \subset Y$*, a pair of convex and concave relaxations of* z *is available as* $z^{\mathrm{cv}}, z^{\mathrm{cc}} : Y \to \mathbb{R}$*, respectively. Then, the set of functions* z^{cv} *and* z^{cc} *is said to be a scheme of estimators of* z *on* Y *[20].*

By this definition, interval bounds are scheme of constant estimators. The inclusion function associated with a scheme of estimators is defined as

$$\mathcal{F}_z(\mathbf{P}) := [\inf_{\mathbf{p}\in\mathbf{P}} z^{\mathrm{cv}}(\mathbf{p}), \sup_{\mathbf{p}\in\mathbf{P}} z^{\mathrm{cc}}(\mathbf{p})], \tag{2.10}$$

with inf and sup being the infimum and supremum, respectively. In general, an inclusion function can be associated with a non-constant enclosure by obtaining interval bounds on that enclosure. For example, the inclusion function associated with a Taylor model of z on $\mathbf{P}$ as $\mathcal{T}_z = \mathcal{P}_z + R_z$ can be obtained by bounding the Taylor model as presented in §2.3.2, i.e.,

$$\mathcal{F}_z(\mathbf{P}) = [\mathcal{P}_z^L, \mathcal{P}_z^U] + R_z. \tag{2.11}$$

Note that exact bounding of the polynomial term can also be used in (2.11) although it is prohibitively expensive for multivariate polynomials. Figure 2.8 illustrates the

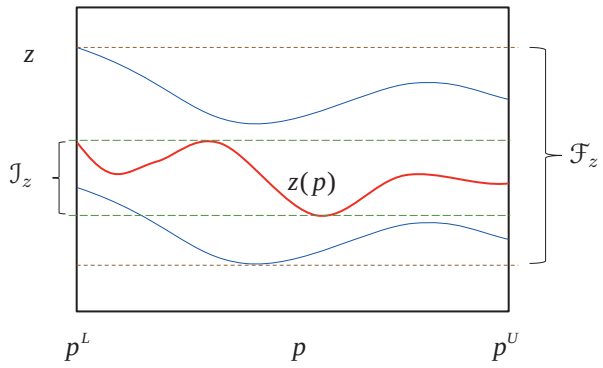

Figure 2.8: Image of the function $z(p)$ and the interval inclusion associated with an arbitrary enclosure of $z(p)$.

image of the function z and the inclusion function associated with an arbitrary enclosure for z.

In the following, two metrics for the convergence order of scheme of estimators are defined, namely the Hausdorff and pointwise metrics.

Definition 2.9 (Hausdorff convergence order)**.** *Given the continuous function* $z : Y \subset \mathbb{R}^n \to \mathbb{R}$ *with* Y *a nonempty, convex set, the inclusion function* $\mathcal{F}_z$ *associated with a scheme of estimators for* z *is Hausdorff convergent of order* $\beta > 0$ *at a point* $\mathbf{p} \in Y$ *if there exists a constant* $\kappa \geq 0$ *such that:*

$$E \leq \kappa \, \mathrm{w}\,(\mathbf{P})^{\beta}, \tag{2.12}$$

for every interval $\mathbf{P} \subset Y$ *with* $\mathbf{p} \in \mathbf{P}$ *[20]. Also,* $\mathcal{F}_z$ *is Hausdorff convergent of order* $\beta > 0$ *on* Y *if it is Hausdorff convergent of order (at least)* β *for every* $\mathbf{p} \in Y$*, with* κ *independent of* $\mathbf{p}$*.*

The Hausdorff metric provides a measure for the convergence of the error bounded over the *entire host set* $\mathbf{P}$. The pointwise metric [19], on the other hand, quantifies the convergence based on the error computed at specific points in $\mathbf{P}$, as defined below.

Definition 2.10 (Pointwise convergence order)**.** *Given the continuous function* $z : Y \subset \mathbb{R}^n \to \mathbb{R}$ *with* Y *a nonempty, convex set, a scheme of estimators for* z *is pointwise convergent of order* $\gamma > 0$ *at a point* $\mathbf{p}$ *if there exists a constant* $\lambda \geq 0$ *such that:*

$$\max\left\{\sup_{\mathbf{p}\in\mathbf{P}} |z(\mathbf{p}) - z^{\mathrm{cv}}(\mathbf{p})|, \sup_{\mathbf{p}\in\mathbf{P}} |z(\mathbf{p}) - z^{\mathrm{cc}}(\mathbf{p})|\right\} \leq \lambda\, \mathrm{w}\,(\mathbf{P})^{\gamma}, \tag{2.13}$$

for every interval $\mathbf{P} \subset Y$ *with* $\mathbf{p} \in \mathbf{P}$*. Also, a scheme of estimators is pointwise convergent of order* $\gamma > 0$ *on* Y *if it is pointwise convergent of order (at least)* γ *for every* $\mathbf{p} \in Y$*, with* λ *independent of* $\mathbf{p}$ *[19].*

For the special case of an interval bound $Z = [z^{\mathrm{L}}, z^{\mathrm{U}}]$, one has $z^{\mathrm{cv}} = z^{\mathrm{L}}$ and $z^{\mathrm{cc}} = z^{\mathrm{U}}$.

Now, some of the main results regarding the convergence order of interval analysis, McCormick relaxations, and Taylor models are summarized as follows.

i. Natural interval extensions have first-order Hausdorff convergence over their domain of approximation. However, the convergence at a point could be higher in some cases (depending on the location of the point in the domain interval, the interval expression used to estimate the function, etc) [8].

ii. A scheme approximating a smooth, nonlinear function can have pointwise convergence order of at most two over its domain of approximation. However, it is possible to have higher order pointwise convergence at specific points in the domain [19].

iii. McCormick relaxations are pointwise convergent of order 2 over their domain of approximation if their underlying interval bounds have quadratic convergence or if the estimators of the factors (of the factorable function) are quadratically convergent in the pointwise sense [20].

iv. Assume the function $z : Y \subset \mathbb{R}^n \to \mathbb{R}$ with Y a nonempty, convex set has range order of $\alpha > 0$ on Y. For a scheme of estimators of z, the following relationship holds between the Hausdorff and pointwise convergence orders $\beta > 0$ and $\gamma > 0$

on Y, respectively [20]:

$$\beta \geq \gamma \geq \min\{\alpha, \beta\}.$$

v. The interval remainder bound of a qth-order Taylor model has Hausdorff convergence of order $\beta \geq q+1$ and pointwise convergence of order $\gamma \geq q+1$ [20].

vi. Assume the polynomial and interval remainder in a qth-order Taylor model have Hausdorff convergence of orders $\beta_p \geq 1$ and $\beta_r \geq 1$, respectively, and pointwise convergence of orders $\gamma_{\mathcal{P}} \geq 1$ and $\gamma_r \geq 1$, respectively. Also, assume that the polynomial has range of order $\alpha_p \geq 1$. Then, the Taylor model has pointwise and Hausdorff convergence of orders [20]

$$\beta_{\mathcal{T}} \geq \min\{\max\{q+1, \alpha_{\mathcal{P}}\}, \beta_{\mathcal{P}}, \beta_r\}, \tag{2.14}$$
$$\gamma_{\mathcal{T}} = \min\{\gamma_{\mathcal{P}}, \gamma_r\}. \tag{2.15}$$

2.3.5 Polyhedral Relaxation of Factorable Programs

An optimization problem in which all the participating functions are factorable is called a factorable program. A factorable program can be approximated by an LP using polyhedral relaxations. Consider the following, potentially nonconvex, NLP:

$$\begin{aligned} \min_{\mathbf{p}} \ & f(\mathbf{p}) \\ \text{s.t. } & \mathbf{g}(\mathbf{p}) \leq \mathbf{0} \\ & \mathbf{s}(\mathbf{p}) = \mathbf{0} \\ & \mathbf{p}^{\mathrm{L}} \leq \mathbf{p} \leq \mathbf{p}^{\mathrm{U}} \end{aligned} \tag{2.16}$$

A polyhedral relaxation of Problem (2.16) is obtained through a three-step procedure as presented in the sequel.

Step 1: Decomposition. Problem (2.16) is reformulated to an equivalent problem, where the nonlinear terms are decomposed by introducing auxiliary variables

and constraints. As a result, the new formulation contains only nonlinear unary and binary terms, as given below (see e.g. [97]).

$$
\begin{aligned}
\min_{\mathbf{v}} \; & v_{\text{obj}} \\
\text{s.t. } & \mathbf{G}\,\mathbf{v} \leq \mathbf{b} \\
& \mathbf{H}\,\mathbf{v} = \mathbf{c} \\
& v_k = v_i v_j, \quad \forall (i,j,k) \in \mathcal{B} \\
& v_k = \frac{v_i}{v_j}, \quad \forall (i,j,k) \in \mathcal{F} \\
& v_k = \varphi_k(v_i), \quad \forall (i,k) \in \mathcal{U} \\
& \mathbf{v}^{\text{L}} \leq \mathbf{v} \leq \mathbf{v}^{\text{U}}
\end{aligned}
\tag{2.17}
$$

where the vector $\mathbf{v}$ contains both the original variables $\mathbf{p}$ and the auxiliary variables introduced during the reformulation; v_{obj} is the variable in the vector $\mathbf{v}$ that corresponds to the objective function f in (2.16); the matrix $\mathbf{G}$ and vector $\mathbf{b}$ represent the original linear inequality constraints (if any) as well as those obtained upon reformulating the nonlinear inequality constraints; similarly, the matrix $\mathbf{H}$ and vector $\mathbf{c}$ contain the original linear equality constraints (if any) as well as the linear relationships introduced upon reformulating both nonlinear equality and inequality constraints. The bounds on the auxiliary variables are also computed in this step by propagating the original variable bounds using interval analysis (see §2.3.1). Observe that the decomposition process has captured all the nonlinearities in bilinear terms ($\mathcal{B}$), binary fractions ($\mathcal{F}$), and univariate functions ($\mathcal{U}$).

Step 2: Relaxation. The nonlinear terms in (2.17) are relaxed so as to arrive at a convex lower-bounding problem. This is done as follows:

- The bilinear terms $v_k = v_i v_j$, $v_i^{\text{L}} \leq v_i \leq v_i^{\text{U}}$, $v_j^{\text{L}} \leq v_j \leq v_j^{\text{U}}$, are replaced by their polyhedral envelopes as [7]:

$$
v_k = v_i v_j \quad \xrightarrow{\text{relax.}} \quad \left\{ \begin{array}{l}
v_k \geq v_i^{\text{L}} v_j + v_j^{\text{L}} v_i - v_i^{\text{L}} v_j^{\text{L}} \\
v_k \geq v_i^{\text{U}} v_j + v_j^{\text{U}} v_i - v_i^{\text{U}} v_j^{\text{U}} \\
v_k \leq v_i^{\text{U}} v_j + v_j^{\text{L}} v_i - v_i^{\text{U}} v_j^{\text{L}} \\
v_k \leq v_i^{\text{L}} v_j + v_j^{\text{U}} v_i - v_i^{\text{L}} v_j^{\text{U}}
\end{array} \right.
$$

- The fractional terms $v_k = \frac{v_i}{v_j}$, $v_i^{\mathrm{L}} \le v_i \le v_i^{\mathrm{U}}$, $v_j^{\mathrm{L}} \le v_j \le v_j^{\mathrm{U}}$, $0 \notin [v_j^{\mathrm{L}}, v_j^{\mathrm{U}}]$ are rewritten as bilinear terms $v_i = v_k v_j$ [99], and relaxed as described earlier, with the following bounds for the variables v_k:

$$\min\left\{\frac{v_j^{\mathrm{L}}}{v_k^{\mathrm{L}}}, \frac{v_j^{\mathrm{L}}}{v_k^{\mathrm{U}}}, \frac{v_j^{\mathrm{U}}}{v_k^{\mathrm{L}}}, \frac{v_j^{\mathrm{U}}}{v_k^{\mathrm{U}}}\right\} =: v_k^{\mathrm{L}} \le v_k \le v_k^{\mathrm{U}} := \max\left\{\frac{v_j^{\mathrm{L}}}{v_k^{\mathrm{L}}}, \frac{v_j^{\mathrm{L}}}{v_k^{\mathrm{U}}}, \frac{v_j^{\mathrm{U}}}{v_k^{\mathrm{L}}}, \frac{v_j^{\mathrm{U}}}{v_k^{\mathrm{U}}}\right\}$$

  Note that the above approach does not give convex/concave envelopes of fractional terms in general.

- The univariate terms $v_k = \varphi(v_i)$, $v_i^{\mathrm{L}} \le v_i \le v_i^{\mathrm{U}}$, are relaxed depending on whether the function φ is convex, concave, or partly convex and partly concave (convexo-concave) on $[v_i^{\mathrm{L}}, v_i^{\mathrm{U}}]$. For convex or concave functions, convex/concave envelopes [58] are constructed as follows.

$$\text{convex case:}\quad v_k = \varphi(v_i) \xrightarrow{\text{relax}} \begin{cases} v_k \ge \varphi(v_i) \\ v_k \le \varphi(v_i^{\mathrm{L}}) + \frac{\varphi(v_i^{\mathrm{U}}) - \varphi(v_i^{\mathrm{L}})}{v_i^{\mathrm{U}} - v_i^{\mathrm{L}}}(v_i - v_i^{\mathrm{L}}) \end{cases}$$

$$\text{concave case:}\quad v_k = \varphi(v_i) \xrightarrow{\text{relax}} \begin{cases} v_k \ge \varphi(v_i^{\mathrm{L}}) + \frac{\varphi(v_i^{\mathrm{U}}) - \varphi(v_i^{\mathrm{L}})}{v_i^{\mathrm{U}} - v_i^{\mathrm{L}}}(v_i - v_i^{\mathrm{L}}) \\ v_k \le \varphi(v_i) \end{cases}$$

  For a convexo-concave function, the convex/concave envelopes can be obtained by piecewise application of the above formulas as follows:

$$v_k = \varphi(v_i) \xrightarrow{\text{relax}} \begin{cases} v_k \ge \begin{cases} \varphi(v_i), & \text{if } v_i \le v_{\mathrm{m}}^{\mathrm{cv}} \\ \varphi(v_i^{\mathrm{U}}) + \frac{\varphi(v_i^{\mathrm{U}}) - \varphi(v_{\mathrm{m}}^{\mathrm{cv}})}{v_i^{\mathrm{U}} - v_{\mathrm{m}}^{\mathrm{cv}}}(v_i - v_i^{\mathrm{U}}), & \text{otherwise} \end{cases} \\ v_k \le \begin{cases} \varphi(v_i), & \text{if } v_i \ge v_{\mathrm{m}}^{\mathrm{cc}} \\ \varphi(v_i^{\mathrm{L}}) + \frac{\varphi(v_i^{\mathrm{L}}) - \varphi(v_{\mathrm{m}}^{\mathrm{cc}})}{v_i^{\mathrm{L}} - v_{\mathrm{m}}^{\mathrm{cc}}}(v_i - v_i^{\mathrm{L}}), & \text{otherwise} \end{cases} \end{cases}$$

  with: $v_{\mathrm{m}}^{\mathrm{cv}}, v_{\mathrm{m}}^{\mathrm{cc}} \in [v_i^{\mathrm{L}}, v_i^{\mathrm{U}}]:\ \varphi'(v_{\mathrm{m}}^{\mathrm{cv}}) = \frac{\varphi(v_i^{\mathrm{U}}) - \varphi(v_{\mathrm{m}}^{\mathrm{cv}})}{v_i^{\mathrm{U}} - v_{\mathrm{m}}^{\mathrm{cv}}}$ and $\varphi'(v_{\mathrm{m}}^{\mathrm{cc}}) = \frac{\varphi(v_i^{\mathrm{L}}) - \varphi(v_{\mathrm{m}}^{\mathrm{cc}})}{v_i^{\mathrm{L}} - v_{\mathrm{m}}^{\mathrm{cc}}}$.

A schematic of convex/concave envelopes for the above three cases is depicted in Fig. 2.9.

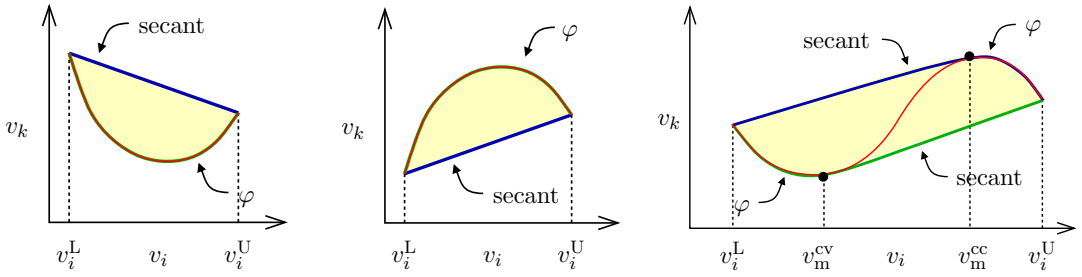

Figure 2.9: Convex and concave envelopes for a convex (left plot), concave (center plot), and convexo-concave (right plot) univariate function.

Step 3: Polyhedral outer-approximation. The bilinear and fractional terms in (2.17) are already replaced by linear relaxations in Step 2. However, the relaxed univariate terms are still nonlinear. Therefore, the last step involves constructing polyhedral outer-approximation for each nonlinear term in order to form a completely linear relaxation of Problem (2.17). The outer-approximation is performed by constructing affine relaxations (a.k.a cuts) at a number of well-chosen points in $[v_i^{\mathrm{L}}, v_i^{\mathrm{U}}]$. For the convex function $\varphi(v_i)$ in Fig. 2.10, it begins by constructing two tangential cuts at both end-points of the interval $[v_i^{\mathrm{L}}, v_i^{\mathrm{U}}]$. Then, an iterative scheme, known as the sandwich algorithm [99], can be applied to add more tangential cuts and reduce the gap between the outer-approximation and the nonlinear relaxation. A number of rules are available to choose the linearization points for the new cuts [99]. For example, the bisection rule and the maximum error rule are illustrated in Fig. 2.10. In the former, the new cut is constructed at the midpoint of $[v_i^{\mathrm{L}}, v_i^{\mathrm{U}}]$ (left plot), while in the latter, the new cut is constructed at the intersection of the two end-point cuts (right plot).

All the nonlinear terms in (2.17) can now be replaced with their polyhedral cuts generated during Steps 2 and 3. Adding these cuts, that have the form of inequality constraints, results in the following LP:

$$\begin{aligned} \min_{\mathbf{v}} \; & v_{\mathrm{obj}} \\ \text{s.t.} \; & \mathbf{G}'\,\mathbf{v} \leq \mathbf{b}' \\ & \mathbf{H}\,\mathbf{v} = \mathbf{c} \\ & \mathbf{v}^{\mathrm{L}} \leq \mathbf{v} \leq \mathbf{v}^{\mathrm{U}} \end{aligned} \tag{2.18}$$

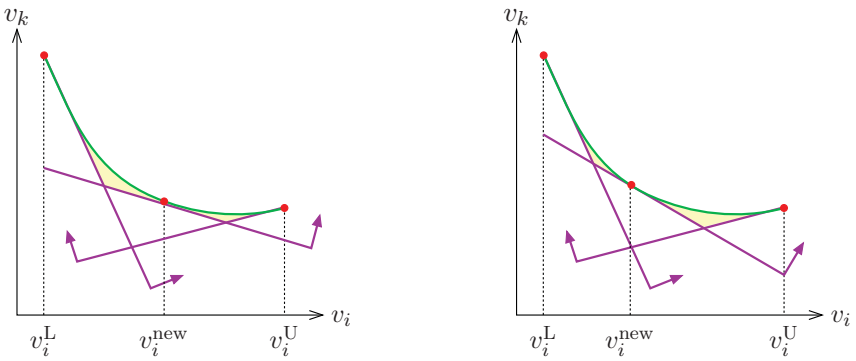

Figure 2.10: Outer-approximation of a convex function using polyhedral cuts; left plot: bisection rule, right plot: maximum error rule.

where the augmented matrix $\mathbf{G}'$ and vector $\mathbf{b}'$ now include respectively coefficients and constants corresponding to the newly added inequalities. Problem (2.18) is then solved to give a lower bound on the original nonconvex problem (2.16) that arises in the branch-and-bound procedure (see §2.5.3).

As will be discussed in §2.5.3.1, the bounds on the variables $\mathbf{v}$ can be potentially shrunk by propagating them through the problem constraints in (2.18). This procedure, called constraint propagation, can also be performed on the nonlinear constraints before they are relaxed in Step 2. In this case, not only can it shrink the variable bounds, but also results in a tighter convex and polyhedral relaxation problem. In turn, the gap between the solution of (2.18) and that of the actual problem (2.17) may be reduced. The three-step procedure for polyhedral relaxations is illustrated in the following example.

2.3.5.1 Illustrative Example

Suppose a lower bound on the global solution of the following factorable program is desired.

$$\begin{aligned} \min_{\mathbf{p}} \ & \mathcal{J}(\mathbf{p}) = p_1 - \exp(p_2) \\ \text{s.t. } & p_1 \exp(p_2) \leq 5 \\ & 3 \leq p_1 \leq 6 \\ & 0 \leq p_2 \leq 4. \end{aligned} \tag{2.19}$$

The first step in constructing polyhedral relaxations is to introduce auxiliary variables for the nonlinear terms and objective function in (2.19). This gives the equivalent program:

$$\begin{aligned} \min_{\mathbf{v}} \ & v_4 \\ \text{s.t. } & v_5 \leq 5 \\ & v_4 = v_1 - v_3 \\ & v_3 = \exp(v_2) \\ & v_5 = v_1 v_3 \\ & \mathbf{v}^{\mathrm{L}} \leq \mathbf{v} \leq \mathbf{v}^{\mathrm{U}}, \end{aligned} \tag{2.20}$$

where $(v_1, v_2) := (p_1, p_2)$ and v_3, v_4, and v_5 are the new auxiliary variables. Observe that the decomposition procedure identifies the identical expression $\exp(v_2)$ in the objective and constraint functions, and introduces the necessary auxiliary variable only once. By propagating the bounds on v_1 and v_2 through the new expressions, the bounds on v_3, v_4, and v_5 are obtained as:

$$\begin{aligned} 1 \leq & v_3 \leq 54.60 \\ -51.60 \leq & v_4 \leq 5 \\ 3 \leq & v_5 \leq 327.60. \end{aligned} \tag{2.21}$$

The next step involves relaxation of the nonlinear terms $v_5 = v_1 v_3$ and $v_3 = \exp(v_2)$. The relaxation of the bilinear term $v_1 v_3$ produces four linear inequalities as

$$\begin{aligned} v_5 - v_1 - 6v_3 &\leq -6 \\ v_5 - 54.60v_1 - 3v_3 &\leq -163.80 \\ -v_5 + 54.60v_1 + 6v_3 &\leq 327.60 \\ -v_5 + v_1 + 3v_3 &\leq 3. \end{aligned} \tag{2.22}$$

The exponential term $\exp(v_2)$ is already convex on $V_2 = [0, 4]$, and is outer-approximated from above using the following concave envelope:

$$4v_3 - 53.60v_2 \leq 4. \tag{2.23}$$

For the outer-approximation of $\exp(v_2)$ from below, two tangential cuts are constructed at the endpoints $v_2 = 0$ and $v_2 = 4$:

$$\begin{aligned} -v_3 + v_2 &\leq -1 \\ -0.0183v_3 + v_2 &\leq 3. \end{aligned} \tag{2.24}$$

Now, the bisection rule is applied to construct another tangential cut at $v_2 = 2$ as

$$-0.135v_3 + v_2 \leq 1. \tag{2.25}$$

Here the exponential term is underestimated by only three cuts for illustrative purposes. Finally, replacing the nonlinear constraints in (2.20) with (2.22), (2.23), (2.24),

and (2.25) gives the following LP problem:

$$
\begin{aligned}
\min_{\mathbf{v}} \ & v_4 \\
\text{s.t.} \quad & v_4 = v_1 - v_3 \\
& v_5 \leq 5 \\
& v_5 - v_1 - 6v_3 \leq -6 \\
& v_5 - 54.60v_1 - 3v_3 \leq -163.80 \\
& -v_5 + 54.60v_1 + 6v_3 \leq 327.60 \\
& -v_5 + v_1 + 3v_3 \leq 3. \\
& 4v_3 - 53.60v_2 \leq 4. \\
& -v_3 + v_2 \leq -1 \\
& -0.0183v_3 + v_2 \leq 3. \\
& -0.135v_3 + v_2 \leq 1. \\
& 3 \leq v_1 \leq 6 \\
& 0 \leq v_2 \leq 4 \\
& 1 \leq v_3 \leq 54.60 \\
& -51.60 \leq v_4 \leq 5 \\
& 3 \leq v_5 \leq 327.60.
\end{aligned}
$$

The solution of this LP gives a minimum objective value of $v_4 = 1.333$. This value serves as a lower bound for the global solution value of (2.19), which happens to be the same as $\mathcal{J}^* = 1.333$ for this simple problem. In general, polyhedral relaxations provide a weaker lower bound than nonlinear convex relaxations. On the other hand, despite introducing a large number of extra variables and constraints, polyhedral relaxations can be solved faster and more reliably than nonlinear relaxations.

2.4 Bounding Solutions of Parametric ODEs

In this section, two validated methods for bounding solutions of nonlinear parametric ODEs are reviewed. The first method, that is based on interval analysis, is presented in §2.4.1. The second method, that incorporates Taylor models, is described in §2.4.2.

2.4.1 Validated Solution of ODEs using Interval Analysis

Consider the IVP in ODE (2.1,2.2). Also, consider a grid $t_0 < t_1 < \cdots < t_m = T$, not necessarily equally spaced, and denote the stepsize from t_j to t_{j+1} by h_j ($h_j = t_{j+1} - t_j$). The solution to (2.1) with an initial condition $\mathbf{x}_j$ at t_j and variable value $\mathbf{p}$ is denoted by $\mathbf{x}(t; t_j, \mathbf{x}_j, \mathbf{p})$; omission of the arguments t_j and $\mathbf{x}_j$ as in $\mathbf{x}(t; \mathbf{p})$ designates the solution to the ODEs (2.1) from the initial conditions (2.2). Likewise, the set of solutions to (2.1) corresponding to initial conditions at t_j in $\mathbf{X}_j$ and variable values in $\mathbf{P}$ is denoted by $\mathbf{x}(t; t_j, \mathbf{X}_j, \mathbf{P}) := \{\mathbf{x}(t; t_j, \mathbf{x}_j, \mathbf{p}) : \mathbf{x}_j \in \mathbf{X}_j, \mathbf{p} \in \mathbf{P}\}$; the set of solutions to the IVP (2.1,2.2) for variable values in $\mathbf{P}$ is represented by $\mathbf{x}(t; \mathbf{P}) := \{\mathbf{x}(t; \mathbf{p}) : \mathbf{p} \in \mathbf{P}\}$.

Assumption 2.1. *The IVP (2.1,2.2) satisfies the following conditions:*

1. $\mathbf{h} : \mathbf{P} \to D$ *is continuous on* $\mathbf{P}$*,*

2. $\mathbf{f} : D \times \mathbf{P} \to \mathbb{R}^{n_x}$ *is* $(k-1)$ *times continuously differentiable on* $D \times \mathbf{P}$*, with* $k \geq 2$.

The following sequence of functions

$$\mathbf{f}^{[0]}(\mathbf{x}(t), \mathbf{p}) = \mathbf{x}(t); \qquad \mathbf{f}^{[i]}(\mathbf{x}(t), \mathbf{p}) = \frac{1}{i}\left[\frac{\partial \mathbf{f}^{[i-1]}}{\partial \mathbf{x}}\mathbf{f}\right](\mathbf{x}(t), \mathbf{p}), \quad \text{for } i \geq 1, \tag{2.26}$$

denotes the ith Taylor coefficient of the solution of the ODEs (2.1), expanded with respect to t, so that for each x_ι, $\iota = 1, 2, \ldots, n_x$:

$$x_\iota(t+h) = \sum_{i=0}^{k-1} h^i f_\iota^{[i]}(\mathbf{x}(t)) + h^k f_\iota^{[k]}(\mathbf{x}(t+\tau_\iota)), \quad \text{for some } \tau_\iota \in [0, h]. \tag{2.27}$$

With the aforementioned notations and assumptions, the validated method is described below.

2.4.1.1 Validated Solution of Initial Value Problems

Given the (autonomous) ODEs

$$\dot{\mathbf{x}}(t) = \mathbf{z}(\mathbf{x}(t)),\ \forall t \in (t_0, t_N], \tag{2.28}$$

the problem is to determine a guaranteed enclosure, $\mathbf{X}_j \supseteq \mathbf{x}(t_j; t_0, \mathbf{X}_0)$, for the solutions at $t_1, \ldots, t_N$, given initial values $\mathbf{x}(t_0) \in \mathbf{X}_0 \subset D$. By 'guaranteed enclosure', it is understood that truncation errors are accounted for in computing the enclosures. A variety of approaches have been developed over the years to tackle this problem, which mainly rely on interval analysis and/or Taylor models [51, 30, 13, 65, 49].

The emphasis in this chapter is on the high-order enclosure (HOE) method by [65]. Assuming that the vector function $\mathbf{z}$ in (2.28) is $(k-1)$ times continuously differentiable with respect to $\mathbf{x}$ and factorable, this method proceeds in two phases:

a) In Phase I, existence and uniqueness of the solution on $[t_j, t_{j+1}]$ are established and an a priori enclosure $\widetilde{\mathbf{X}}_j \supseteq \{\mathbf{x}(t; t_j, \mathbf{x}_j) : t_j \leq t \leq t_{j+1}, \mathbf{x}_j \in \mathbf{X}_j\}$ is computed, given the enclosure $\mathbf{X}_j \supseteq \mathbf{x}(t_j; t_0, \mathbf{X}_0)$ at t_j.

b) In Phase II, a tightened enclosure $\mathbf{X}_{j+1} \supseteq \mathbf{x}(t_{j+1}; t_0, \mathbf{X}_0)$ is computed based on high-order Taylor series expansion with the a priori enclosure $\widetilde{\mathbf{X}}_j$ and the previous enclosure $\mathbf{X}_j$.

This two-phase procedure is depicted in Fig. 2.11 and further detailed below.

Phase I: Obtaining a Priori State Bounds. The HOE method is based on the following theorem.

Theorem 2.1 ([30]). *Let* $\widetilde{\mathbf{X}}_j^0 \subseteq D$, *and let* $\mathbf{x}_j \in \mathrm{int}(\widetilde{\mathbf{X}}_j^0)$. *If*

$$\left\{ \sum_{i=0}^{k-1} (t - t_j)^i \mathbf{z}^{[i]}(\mathbf{x}_j) + (t - t_j)^k \mathbf{z}^{[k]}(\widetilde{\mathbf{X}}_j^0) : t_j \leq t \leq t_{j+1} \right\} \subseteq \widetilde{\mathbf{X}}_j^0,$$

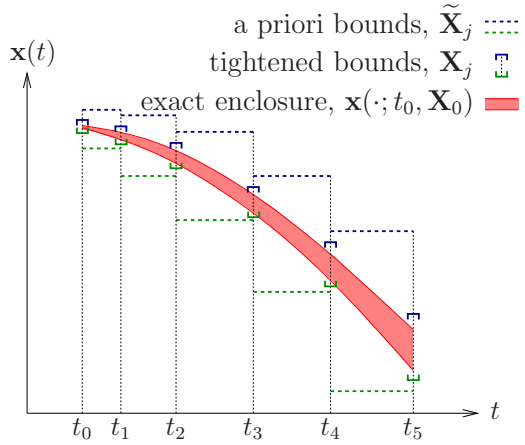

Figure 2.11: Two-phase interval series method.

then

$$\mathbf{x}(t; t_j, \mathbf{x}_j) \in \sum_{i=0}^{k-1} (t-t_j)^i \mathbf{z}^{[i]}(\mathbf{x}_j) + (t-t_j)^k \mathbf{z}^{[k]}(\widetilde{\mathbf{X}}_j^0), \quad \textit{for all } t \in [t_j, t_{j+1}].$$

From a practical viewpoint, Theorem 2.1 is useful to establish existence and uniqueness of the solutions of (2.28) on $[t_j, t_{j+1}]$. It also provides a mechanism for computing an a priori enclosure as:

$$\widetilde{\mathbf{X}}_j = \sum_{i=0}^{k-1} [0, h_j]^i \mathbf{z}^{[i]}(\mathbf{X}_j) + [0, h_j]^k \mathbf{z}^{[k]}(\widetilde{\mathbf{X}}_j^0),$$

with $h_j := t_{j+1} - t_j$. An algorithm that implements this approach with variable stepsize can be found, e.g., in [66].

Phase II: Tightening State Bounds. The Taylor series expansion of the solutions $\mathbf{x}$ to (2.28) between times t_j and t_{j+1} reads

$$x_\iota(t_{j+1}) = \sum_{i=0}^{k-1} h_j^i z_\iota^{[i]}(\mathbf{x}(t_j)) + h_j^k z_\iota^{[k]}(\mathbf{x}(t_j + \tau_\iota)), \tag{2.29}$$

for some $\tau_\iota \in [0, h_j]$ and $\iota = 1, 2, \ldots, n_x$. Given the enclosure $\mathbf{X}_j$ at t_j, along with the a priori bounds $\widetilde{\mathbf{X}}_j$ on $[t_j, t_{j+1}]$, an enclosure $\mathbf{X}_{j+1}$ of $\mathbf{x}(t_{j+1}; t_0, \mathbf{X}_0)$ is most easily computed as

$$\mathbf{X}_{j+1} = \sum_{i=0}^{k-1} h_j^i \mathbf{z}^{[i]}(\mathbf{X}_j) + h_j^k \mathbf{z}^{[k]}(\widetilde{\mathbf{X}}_j).$$

However, the widths of the successive interval enclosures can only grow with this approach, $\mathrm{w}(\mathbf{X}_{j+1}) \geq \mathrm{w}(\mathbf{X}_j)$, even in cases where the actual solution set would be contracting. This deficiency can be circumvented by applying the mean-value theorem to (2.29) in terms of the variables $\mathbf{x}$,

$$\begin{aligned} x_\iota(t_{j+1}) = \sum_{i=0}^{k-1} \left(h_j^i z_\iota^{[i]}(\hat{\mathbf{x}}_j) + h_j^i \frac{\partial z_\iota^{[i]}}{\partial \mathbf{x}} (\eta_\iota^i (\mathbf{x}(t_j) - \hat{\mathbf{x}}_j) + \hat{\mathbf{x}}_j)) \left[\mathbf{x}(t_j) - \hat{\mathbf{x}}_j \right] \right) \\ + h_j^k z_\iota^{[k]}(\mathbf{x}(t_j + \tau_\iota)), \quad \text{for some } \eta_\iota^i \in [0,1] \text{ and } \iota = 1, 2, \ldots, n_x, \end{aligned}$$

where $\hat{\mathbf{x}}_j$ can be any reference point in $\mathbf{X}_j$. Then, considering the interval extension of the foregoing expression gives:

$$\mathbf{X}_{j+1} = \underbrace{\sum_{i=0}^{k-1} h_j^i \mathbf{z}^{[i]}(\hat{\mathbf{x}}_j)}_{=:\, \mathbf{v}_{j+1}} + \underbrace{h_j^k \mathbf{z}^{[k]}(\widetilde{\mathbf{X}}_j)}_{=:\, \mathbf{R}_{j+1}} + \underbrace{\sum_{i=0}^{k-1} h_j^i \frac{\partial \mathbf{z}^{[i]}}{\partial \mathbf{x}} (\mathbf{X}_j) \left[\mathbf{X}_j - \hat{\mathbf{x}}_j \right]}_{=:\, \mathbf{J}_{j+1}}. \tag{2.30}$$

Although this approach has the capability to follow contracting solution sets, for example when some diagonal elements of $\frac{\partial \mathbf{z}^{[i]}}{\partial \mathbf{x}}$ are negative, it is subject to the wrapping effect. The interval-matrix-vector product $\mathbf{J}_{j+1}[\mathbf{X}_j - \hat{\mathbf{x}}_j]$ in (2.30) is the critical

term contributing to this effect, because its direct evaluation often leads to significant overestimation. One way to mitigate it is by changing the order of the interval operations as follows:

$$\boldsymbol{\Delta}_{j+1} = \mathbf{A}_{j+1}^{-1}\left[\mathbf{R}_{j+1} - \mathrm{m}(\mathbf{R}_{j+1})\right] + \left[\mathbf{A}_{j+1}^{-1}(\mathbf{J}_{j+1}\mathbf{A}_j)\right]\boldsymbol{\Delta}_j, \tag{2.31}$$

$$\mathbf{X}_{j+1} = \mathbf{v}_{j+1} + \mathbf{R}_{j+1} + (\mathbf{J}_{j+1}\mathbf{A}_j)\boldsymbol{\Delta}_j, \tag{2.32}$$

$$\hat{\mathbf{x}}_{j+1} = \mathbf{v}_{j+1} + \mathrm{m}(\mathbf{R}_{j+1}), \tag{2.33}$$

where $\mathbf{A}_j \in \mathbb{R}^{n_x \times n_x}$ are nonsingular transformation matrices. The approach is initialized with $\boldsymbol{\Delta}_0 = \mathbf{X}_0 - \hat{\mathbf{x}}_0$ and $\mathbf{A}_0 = \mathbf{I}$. At subsequent steps, the transformation matrix $\mathbf{A}_{j+1}$ in (2.31),(2.32) can be taken as the orthogonal matrix, $\mathbf{Q}_{j+1}$, obtained from QR decomposition of $\mathrm{m}(\mathbf{J}_{j+1}\mathbf{A}_j)$. This approach, known as *Lohner's QR method* [51], is often regarded as the best known method in terms of numerical stability [74, 41].

2.4.1.2 Validated Solution of Parametric Initial Value Problems

The two-phase algorithm outlined previously can be readily extended to determine a guaranteed enclosure, $\mathbf{X}_j \supseteq \mathbf{x}(t_j; \mathbf{P})$, for the solutions of parametric IVPs of the form (2.1),(2.2) at given times $t_k, k = 1, \ldots, N$. In Phase I, the approach is essentially the same, except that variable dependence is now explicitly accounted for. The following corollary extends Theorem 2.1 to parametric IVPs.

Corollary 2.1. *Let* $\widetilde{\mathbf{X}}_j^0 \subseteq D$, *and* $\mathbf{x}_j \in \mathrm{int}(\widetilde{\mathbf{X}}_j^0)$. *If*

$$\left\{\sum_{i=0}^{k-1}(t-t_j)^i\mathbf{f}^{[i]}(\mathbf{x}_j,\mathbf{p}) + (t-t_j)^k\mathbf{f}^{[k]}(\widetilde{\mathbf{X}}_j^0,\mathbf{p}) : t_j \le t \le t_{j+1}, \mathbf{p}\in\mathbf{P}\right\} \subseteq \widetilde{\mathbf{X}}_j^0, \tag{2.34}$$

then

$$\mathbf{x}(t;t_j,\mathbf{x}_j,\mathbf{p}) \in \sum_{i=0}^{k-1}(t-t_j)^i\mathbf{f}^{[i]}(\mathbf{x}_j,\mathbf{p}) + (t-t_j)^k\mathbf{f}^{[k]}(\widetilde{\mathbf{X}}_j^0,\mathbf{p}), \tag{2.35}$$

for all $t \in [t_j, t_{j+1}]$ *and all* $\mathbf{p} \in \mathbf{P}$.

Proof. *Let* $\underline{\mathbf{p}}$ *be any point in* $\mathbf{P}$. *From* (2.34),

$$\sum_{i=0}^{k-1}(t-t_j)^i \mathbf{f}^{[i]}(\mathbf{x}_j, \underline{\mathbf{p}}) + (t-t_j)^k \mathbf{f}^{[k]}(\widetilde{\mathbf{X}}_j^0, \underline{\mathbf{p}}) \in \widetilde{\mathbf{X}}_j^0, \quad \textit{for all } t_j \le t \le t_{j+1}.$$

Applying Theorem 2.1 with $\mathbf{z}(\mathbf{x}) \leftarrow \mathbf{f}(\mathbf{x}, \underline{\mathbf{p}})$ *gives*

$$\mathbf{x}(t; t_j, \mathbf{x}_j, \underline{\mathbf{p}}) \in \sum_{i=0}^{k-1}(t-t_j)^i \mathbf{f}^{[i]}(\mathbf{x}_j, \underline{\mathbf{p}}) + (t-t_j)^k \mathbf{f}^{[k]}(\widetilde{\mathbf{X}}_j^0, \underline{\mathbf{p}}),$$

for all $t_j \le t \le t_{j+1}$, *therefore proving the result.*

A practical implication of Corollary 2.1 is that an a priori enclosure can be obtained as

$$\widetilde{\mathbf{X}}_j = \sum_{i=0}^{k-1}[0, h_j]^i \mathbf{f}^{[i]}(\mathbf{X}_j, \mathbf{P}) + [0, h_j]^k \mathbf{f}^{[k]}(\widetilde{\mathbf{X}}_j^0, \mathbf{P}), \tag{2.36}$$

where $\widetilde{\mathbf{X}}_j^0$ and $h_j = t_{j+1} - t_j$ are such that (2.34) is satisfied for each $\mathbf{x}_j \in \mathbf{X}_j$.

In Phase II, a refined enclosure can be obtained from:

$$\begin{aligned}\mathbf{X}_{j+1} = &\underbrace{\sum_{i=0}^{k-1} h_j^i \mathbf{f}^{[i]}(\hat{\mathbf{x}}_j), \hat{\mathbf{p}})}_{=:\,\mathbf{v}_{j+1}} + \underbrace{h_j^k \mathbf{f}^{[k]}(\widetilde{\mathbf{X}}_j, \mathbf{P})}_{=:\,\mathbf{R}_{j+1}} + \underbrace{\sum_{i=0}^{k-1} h_j^i \frac{\partial \mathbf{f}^{[i]}}{\partial \mathbf{x}}(\mathbf{X}_j, \mathbf{P})\,[\mathbf{X}_j - \hat{\mathbf{x}}_j]}_{=:\,\mathbf{J}_{j+1}^{\mathbf{x}}} \\ &+ \underbrace{\sum_{i=0}^{k-1} h_j^i \frac{\partial \mathbf{f}^{[i]}}{\partial \mathbf{p}}(\mathbf{X}_j, \mathbf{P})\,[\mathbf{P} - \hat{\mathbf{p}}]}_{=:\,\mathbf{J}_{j+1}^{\mathbf{p}}},\end{aligned} \tag{2.37}$$

for any reference point $(\hat{\mathbf{x}}_j, \hat{\mathbf{p}}) \in \mathbf{X}_j \times \mathbf{P}$. Moreover, it is easily checked that the order of the interval operations in the interval-matrix-vector product $\mathbf{J}_{j+1}^{\mathbf{x}}[\mathbf{X}_j - \hat{\mathbf{x}}_j]$

can be changed as follows in order to mitigate the wrapping effect:

$$\begin{aligned}\boldsymbol{\Delta}_{j+1} &= \mathbf{A}_{j+1}^{-1}\left[\mathbf{R}_{j+1} - \mathrm{m}(\mathbf{R}_{j+1})\right] + \left[\mathbf{A}_{j+1}^{-1}(\mathbf{J}_{j+1}^{\mathbf{x}}\mathbf{A}_j)\right]\boldsymbol{\Delta}_j \\ &\quad + \left[\mathbf{A}_{j+1}^{-1}\mathbf{J}_{j+1}^{\mathbf{p}}\right](\mathbf{P} - \hat{\mathbf{p}}) \qquad (2.38)\\ \mathbf{X}_{j+1} &= \mathbf{v}_{j+1} + \mathbf{R}_{j+1} + \left[\mathbf{J}_{j+1}^{\mathbf{x}}\mathbf{A}_j\right]\boldsymbol{\Delta}_j + \mathbf{J}_{j+1}^{\mathbf{p}}\left[\mathbf{P} - \hat{\mathbf{p}}\right], \qquad (2.39)\end{aligned}$$

with $\hat{\mathbf{x}}_{j+1}$ given by (2.33). Here again, the transformation matrix $\mathbf{A}_{j+1} \in \mathbb{R}^{n_x \times n_x}$ can be taken as the orthogonal matrix in the QR decomposition of $\mathrm{m}(\mathbf{J}_{j+1}^{\mathbf{x}}\mathbf{A}_j)$.

2.4.2 Validated Solution of ODEs using Taylor Models

The validated ODE method presented in §2.4.1 incorporates mechanisms to efficiently mitigate the wrapping effect, yet it does not address the dependency problem. Failure to account for the latter can result in overly pessimistic bounds and lead to premature termination of the bounding procedure, especially for highly nonlinear dynamic systems. In response to this, Lin and Stadtherr [49] proposed to use Taylor models instead of interval bounds within the HOE method. Here, Lin and Stadtherr's algorithm is presented as an extension of the two-phase procedure in §2.4.1; as such, it contains a few differences with respect to their original algorithm, which will be pointed out.

Assumption 2.2. *In addition to having the same prerequisites as the HOE method (see Assumption 2.1), Taylor model-based methods require the vector function* $\mathbf{f} : D\times\mathbf{P} \to \mathbb{R}^{n_x}$ *to be* $(q+1)$ *times continuously differentiable in* $\mathbf{p}$ *on* $D\times\mathbf{P}$*, respectively, with* $q \geq 1$ *the selected order for the Taylor models, as well as the vector function* $\mathbf{h} : \mathbf{P} \to D$ *to be* $(q+1)$ *times continuously differentiable on* $\mathbf{P}$*.*

The computations start with a Taylor model of $\mathbf{x}(t_0)$ on $\mathbf{P}$ obtained as

$$\mathcal{T}_{\mathbf{x}_0}(\mathbf{p}) = \mathcal{P}_{\mathbf{h}}(\mathbf{p}) + R_{\mathbf{h}}, \quad \text{for each } \mathbf{p} \in \mathbf{P}. \qquad (2.40)$$

Then, a two-phase procedure is applied at each grid point t_k, $k = 1, \ldots, N$, similar to the one described in §2.4.1.

Phase I Given a Taylor model $\mathcal{T}_{\mathbf{x}_j}$ of $\mathbf{x}(t_j;\cdot)$ on $\mathbf{P}$, a stepsize h_j and a Taylor model $\mathcal{T}_{\widetilde{\mathbf{x}}_j}$ are computed such that $\mathcal{T}_{\widetilde{\mathbf{x}}_j}$ encloses $\mathbf{x}$ on $[t_j, t_j+h_j]\times\mathbf{P}$. Such an a priori Taylor model enclosure can be obtained as

$$\mathcal{T}_{\widetilde{\mathbf{x}}_j}(\mathbf{p}) = \sum_{i=0}^{k-1}[0,h_j]^i\mathcal{T}_{\mathbf{f}^{[i]}}(\mathcal{T}_{\mathbf{x}_j}(\mathbf{p}),\mathbf{p}) + [0,h_j]^k\mathbf{f}^{[k]}(\widetilde{\mathbf{X}}_j^0,\mathbf{P}), \tag{2.41}$$

where $\widetilde{\mathbf{X}}_j^0$ and $h_j = t_{j+1}-t_j$ satisfy the condition (2.34) in Corollary 2.1, for each $\mathbf{x}_j \in \mathbf{X}_j$. It is not hard to see from the Taylor expansion (2.27) and the properties of Taylor models that (2.41) yields a valid enclosure of $\mathbf{x}$ on $[t_j, t_{j+1}]\times\mathbf{P}$. Notice that the a priori enclosure $\widetilde{\mathbf{X}}_j$ in Lin and Stadtherr's algorithm [49] is simply obtained by (2.36), which is computationally cheaper and provides similar bounding quality. However, the algorithm presented here is meant to be a full Taylor model version of the one in §2.4.1. Then, Lin and Stadtherr's algorithm can be obtained upon necessary simplifications.

Phase II Given a Taylor model $\mathcal{T}_{\mathbf{x}_j}$ at t_j and an a priori Taylor model $\mathcal{T}_{\widetilde{\mathbf{x}}_j}$ on $[t_j, t_{j+1}]$, a Taylor model $\mathcal{T}_{\mathbf{x}_{j+1}}$ can be computed for $\mathbf{x}(t_{j+1};\cdot)$ on $\mathbf{P}$ based on the high-order Taylor series expansion with the mean-value theorem given by (3.10). It reads

$$\begin{aligned}\mathcal{T}_{\mathbf{x}_{j+1}}(\mathbf{p}) =& \sum_{i=0}^{k-1} h_j^i\mathcal{T}_{\mathbf{f}^{[i]}}(\hat{\mathbf{x}}_j,\hat{\mathbf{p}}) + h_j^k\mathcal{T}_{\mathbf{f}^{[k]}}(\mathcal{T}_{\widetilde{\mathbf{x}}_j}(\mathbf{p}),\mathbf{p}) \\ &+ \sum_{i=0}^{k-1} h_j^i\Bigg(\mathcal{T}_{\frac{\partial\mathbf{f}^{[i]}}{\partial\mathbf{x}}}\left(\mathcal{T}_{\boldsymbol{\mu}_j}(\mathbf{p}),\mathcal{T}_{\boldsymbol{\rho}}(\mathbf{p})\right)\left[\mathcal{T}_{\mathbf{x}_j}(\mathbf{p})-\hat{\mathbf{x}}_j\right] \\ &\qquad + \mathcal{T}_{\frac{\partial\mathbf{f}^{[i]}}{\partial\mathbf{p}}}\left(\mathcal{T}_{\boldsymbol{\mu}_j}(\mathbf{p}),\mathcal{T}_{\boldsymbol{\rho}}(\mathbf{p}))\right)[\mathbf{p}-\hat{\mathbf{p}}]\Bigg),\end{aligned} \tag{2.42}$$

where

$$\begin{aligned}\mathcal{T}_{\boldsymbol{\mu}_j}(\mathbf{p}) &= \hat{\mathbf{x}}_j + [0,1](\mathcal{T}_{\mathbf{x}_j}(\mathbf{p})-\hat{\mathbf{x}}_j)\\ \mathcal{T}_{\boldsymbol{\rho}}(\mathbf{p}) &= \hat{\mathbf{p}} + [0,1](\mathcal{T}_{\mathbf{p}}-\hat{\mathbf{p}}) = \hat{\mathbf{p}} + [0,1](\mathbf{p}-\hat{\mathbf{p}}),\end{aligned}$$

and $\mathcal{T}_{\frac{\partial \mathbf{f}^{[i]}}{\partial \mathbf{x}}}$ and $\mathcal{T}_{\frac{\partial \mathbf{f}^{[i]}}{\partial \mathbf{p}}}$ are Taylor models of the Jacobian matrices $\frac{\partial \mathbf{f}^{[i]}}{\partial \mathbf{x}}$ and $\frac{\partial \mathbf{f}^{[i]}}{\partial \mathbf{p}}$, respectively. Note that taking the variable reference point $\hat{\mathbf{p}} = \mathbf{p}$ will reduce (2.42) as:

$$\begin{aligned}\mathcal{T}_{\mathbf{x}_{j+1}}(\mathbf{p}) =& \sum_{i=0}^{k-1} h_j^i \mathcal{T}_{\mathbf{f}^{[i]}}(\hat{\mathbf{x}}_j, \mathbf{p}) + h_j^k \mathcal{T}_{\mathbf{f}^{[k]}}(\mathcal{T}_{\tilde{\mathbf{x}}_j}(\mathbf{p}), \mathbf{p}) \\ &+ \sum_{i=0}^{k-1} h_j^i \mathcal{T}_{\frac{\partial \mathbf{f}^{[i]}}{\partial \mathbf{x}}} \left(\mathcal{T}_{\boldsymbol{\mu}_j}(\mathbf{p}), \mathbf{p}\right) \left[\mathcal{T}_{\mathbf{x}_j}(\mathbf{p}) - \hat{\mathbf{x}}_j\right].\end{aligned} \tag{2.43}$$

A difficulty with (2.43) is that $\{\eta(\boldsymbol{\xi} - \hat{\mathbf{x}}_j) + \hat{\mathbf{x}}_j : \boldsymbol{\xi} \in \mathcal{T}_{\mathbf{x}_j}(\mathbf{p}), \mathbf{p} \in \mathbf{P}, 0 \le \eta \le 1\} \supseteq \{\mathcal{T}_{\mathbf{x}_j}(\mathbf{p}) : \mathbf{p} \in \mathbf{P}\}$, but equality may not hold in general. This extra overestimation can be circumvented by choosing a reference point $\hat{\mathbf{x}}_j$ that is a function of the variables $\mathbf{p}$ [49]. With the particular variable reference $\hat{\mathbf{x}}_j(\mathbf{p}) = \mathcal{P}^{\mathsf{C}}_{\mathbf{x}_j}(\mathbf{p})$, where $\mathcal{P}^{\mathsf{C}}_{\mathbf{x}_j}(\mathbf{p})$ is a centered Taylor polynomial of $\mathbf{x}_j$ (see §2.3.2), one has:

$$\begin{aligned}&\mathcal{P}^{\mathsf{C}}_{\mathbf{x}_j}(\mathbf{p}) + [0,1]\left(\boldsymbol{\xi} - \mathcal{P}^{\mathsf{C}}_{\mathbf{x}_j}(\mathbf{p})\right) \subset \mathcal{P}^{\mathsf{C}}_{\mathbf{x}_j}(\mathbf{p}) + [0,1] R^{\mathsf{C}}_{\mathbf{x}_j} = \mathcal{P}^{\mathsf{C}}_{\mathbf{x}_j}(\mathbf{p}) + R^{\mathsf{C}}_{\mathbf{x}_j} \\ &= \mathcal{T}^{\mathsf{C}}_{\mathbf{x}_j}(\mathbf{p}) = \mathcal{T}_{\mathbf{x}_j}(\mathbf{p}), \quad \text{for all } \boldsymbol{\xi} \in \mathcal{T}_{\mathbf{x}_j}(\mathbf{p}).\end{aligned}$$

It follows that tighter Taylor models for $\mathbf{x}(t_{j+1}; \cdot)$ on $\mathbf{P}$ can be computed as:

$$\begin{aligned}\mathcal{T}_{\mathbf{x}_{j+1}}(\mathbf{p}) =& \underbrace{\sum_{i=0}^{k-1} h_j^i \mathcal{T}_{\mathbf{f}^{[i]}}(\mathcal{P}^{\mathsf{C}}_{\mathbf{x}_j}(\mathbf{p}), \mathbf{p})}_{=: \mathcal{T}_{\mathbf{V}_{j+1}}(\mathbf{p})} + \underbrace{h_j^k \mathcal{T}_{\mathbf{f}^{[k]}}(\mathcal{T}_{\tilde{\mathbf{x}}_j}(\mathbf{p}), \mathbf{p})}_{=: \mathcal{T}_{\mathbf{R}_{j+1}}(\mathbf{p})} \\ &+ \underbrace{\left[\sum_{i=0}^{k-1} h_j^i \mathcal{T}_{\frac{\partial \mathbf{f}^{[i]}}{\partial \mathbf{x}}}(\mathcal{T}_{\mathbf{x}_j}(\mathbf{p}), \mathbf{p})\right]}_{=: \mathcal{T}_{\mathbf{J}_{j+1}}(\mathbf{p})} R^{\mathsf{C}}_{\mathbf{x}_j}.\end{aligned} \tag{2.44}$$

In order to avoid direct evaluation of the matrix/vector product terms in (2.44) that can lead to large overestimation due to the wrapping effect, the order of the product

operations can be rearranged in the following way:

$$\mathcal{T}_{\mathbf{x}_{j+1}}(\mathbf{p}) = \mathcal{T}_{\mathbf{V}_{j+1}}(\mathbf{p}) + \mathcal{T}_{\mathbf{R}_{j+1}}(\mathbf{p}) + \left[\mathcal{T}_{\mathbf{J}_{j+1}}(\mathbf{p})\,\mathbf{A}_j\right]\mathcal{T}_{\mathbf{\Delta}_j}(\mathbf{p}) \tag{2.45}$$

$$\begin{aligned}\mathcal{T}_{\mathbf{\Delta}_{j+1}}(\mathbf{p}) = {} & \left[\mathbf{A}_{j+1}^{-1}\left(\mathcal{T}_{\mathbf{J}_{j+1}}(\mathbf{p})\,\mathbf{A}_j\right)\right]\mathcal{T}_{\mathbf{\Delta}_j}(\mathbf{p}) \\ & + \mathbf{A}_{j+1}^{-1}\left[\mathcal{T}_{\mathbf{R}_{j+1}}(\mathbf{p}) + \mathcal{T}_{\mathbf{V}_{j+1}}(\mathbf{p}) - \mathcal{P}^{\mathsf{C}}_{\mathbf{x}_{j+1}}(\mathbf{p})\right],\end{aligned} \tag{2.46}$$

with initial values given by (2.40), $\mathcal{T}_{\mathbf{\Delta}_0}(\mathbf{p}) = R^{\mathsf{C}}_{\mathbf{x}_0}$ and $\mathbf{A}_0 = \mathbf{I}$. Regarding the choice of the transformation matrix $\mathbf{A}_{j+1}$, one option consists in bounding the range of $\mathcal{T}_{\mathbf{J}_{j+1}}\mathbf{A}_j$ on $\mathbf{P}$ first, say $\mathbf{J}_{j+1}\mathbf{A}_j$, and then taking the orthogonal matrix in the QR decomposition of $\mathrm{m}(\mathbf{J}_{j+1}\mathbf{A}_j)$.

This approach of handling the wrapping effect is different from the one proposed by [49], which uses a Taylor model with parallelepiped remainder bounds instead of interval remainder bounds; however, the numerical results usually turn out to be similar (see Fig. 4.7 in §4.2.1). Yet another approach is considered by [14] that involves a type of preconditioning method.

Finally, once a Taylor model $\mathcal{T}_{\mathbf{x}_j}$ has been computed for $\mathbf{x}(t_j;\cdot)$ on $\mathbf{P}$, an interval enclosure $\mathbf{X}_j$ can be readily obtained by bounding the range of this Taylor model as explained in §2.3.2.

Note that Phase II in Lin and Stadtherr's procedure can be seen as a simplification of (2.44) to reduce expensive Taylor model computations. Particularly the Taylor model evaluation of the Jacobian matrix $\frac{\partial \mathbf{f}^{[i]}}{\partial \mathbf{x}}$ is found to be dominant in terms of computational effort. Since $\mathcal{T}_{\frac{\partial \mathbf{f}^{[i]}}{\partial \mathbf{x}}}(\mathcal{T}_{\mathbf{x}_j}(\mathbf{p}),\mathbf{p}) \subseteq \frac{\partial \mathbf{f}^{[i]}}{\partial \mathbf{x}}(\mathbf{X}_j,\mathbf{P})$, Lin and Stadtherr [49] evaluated the Jacobian using interval arithmetic. Also, they used an interval form for the remainder $\mathbf{R}_{j+1}$ based on the a priori interval bound $\widetilde{\mathbf{X}}_j$. These changes lead to the following formula for computing a tighter Taylor model for $\mathbf{x}(t_{j+1};\cdot)$ on $\mathbf{P}$:

$$\begin{aligned}\mathcal{T}_{\mathbf{x}_{j+1}}(\mathbf{p}) = {} & \sum_{i=0}^{k-1} h_j^i \mathcal{T}_{\mathbf{f}^{[i]}}(\mathcal{P}^{\mathsf{C}}_{\mathbf{x}_j}(\mathbf{p}),\mathbf{p}) + h_j^k \mathbf{f}^{[k]}(\widetilde{\mathbf{X}}_j,\mathbf{P}) \\ & + \left[\sum_{i=0}^{k-1} h_j^i \frac{\partial \mathbf{f}^{[i]}}{\partial \mathbf{x}}(\mathbf{X}_j,\mathbf{P})\right] R^{\mathsf{C}}_{\mathbf{x}_j}.\end{aligned} \tag{2.47}$$

Here again, the wrapping effect can be mitigated with the same rearrangement given by (2.45) and (2.46), but with the Taylor models $\mathcal{T}_{\mathbf{R}_{j+1}}(\mathbf{p})$ and $\mathcal{T}_{\mathbf{J}_{j+1}}(\mathbf{p})$ replaced by

their interval counterparts.

2.5 Continuous Local and Global Optimization

This section reviews some topics in local and global optimization. The notions of global and local optima along with a classification of optimization methods are presented in §2.5.1. Convex optimization problems are defined in §2.5.2. The SBB technique for global optimization is reviewed in §2.5.3.

2.5.1 Global and Local Solution

In this section, the notions of global and local optima are formalized. Consider the following minimization problem

$$\min_{\mathbf{p}\in D} \mathcal{J}(\mathbf{p}) \tag{2.48}$$

with $D \subset \mathbb{R}^{n_p}$ being the feasible set, which is obtained by intersecting all of the problem constraints.

Definition 2.11 (Global minimum)**.** *A point* $\mathbf{p}^* \in D$ *is said to be a global minimum of* $\mathcal{J}$ *on* D *if*

$$\mathcal{J}(\mathbf{p}^*) \leq \mathcal{J}(\mathbf{p}) \qquad \forall \mathbf{p} \in D. \tag{2.49}$$

Then, $\mathcal{J}(\mathbf{p}^*)$ *is called the global* solution *value of the minimization problem* (2.48)*. For a global maximum in a maximization problem, the inequality* (2.49) *is reversed.*

Remark 2.5. *Definition 2.11 also holds without the qualifier "global". This is because "minimum" implicitly means a feasible point at which the* smallest *objective*

function value is achieved. However, the qualifier global is often used to emphasize that the minimum holds over the entire *feasible set.*

Remark 2.6. *Problem* (2.48) *can have more than one or even an infinite number of global solutions* $\mathbf{p}^*$*. However, the global solution value* $\mathcal{J}(\mathbf{p}^*)$ *is unique. For example, the global solution value of* $\min_p sin(p)$ *is -1, which is attained at an infinite number of points* $p^* = -\pi/2 + k\pi$, $k = 1, 2, \ldots, \infty$.

Unlike global optima which are defined over the entire feasible set, local optima are defined for a neighborhood in the feasible set surrounding them.

Definition 2.12 (Local minimum)**.** *A point* $\mathbf{p}^* \in D$ *is said to be a local minimum of* $\mathcal{J}$ *on* D *if*

$$\exists \delta > 0 \quad such\ that \quad \mathcal{J}(\mathbf{p}^*) \leq \mathcal{J}(\mathbf{p}) \qquad \forall \mathbf{p} \in D \quad And \quad \|\mathbf{p} - \mathbf{p}^*\| \leq \delta \tag{2.50}$$

where $\|.\|$ *is any norm on* $\mathbb{R}^n$*. For a local maximum in a maximization problem, the inequality* $\mathcal{J}(\mathbf{p}^*) \geq \mathcal{J}(\mathbf{p})$ *holds instead.*

In other words, a local optimum is a feasible solution with a sufficiently small neighborhood where no other feasible solutions can give a superior objective function value [73].

Remark 2.7. *From Definitions 2.11 and 2.12, it is obvious that any global optimum is also a local optimum with its neighborhood being as large as the entire feasible set.*

It is usually easier to obtain a local optimum with certainty than a global optimum. Optimization algorithms based on local search can certify finding a local optimum only. For example, consider minimization of the objective function in Fig. 2.12. Gradient-based methods typically start from an initial point p^0, and use gradient information to identify a minimum within a neighborhood of the initial point. There are three valleys in Fig. 2.12, two of which are only local solutions. Depending on the location of the initial point, the search may end up in any of the local

solutions. Even if the global solution is found, it cannot be certified due to the lack of global information about the feasible domain and the objective function. Optimization algorithms with this property are classified as *incomplete* algorithms [69]
There is a second class of algorithms, called *asymptotically complete* algorithms [69],

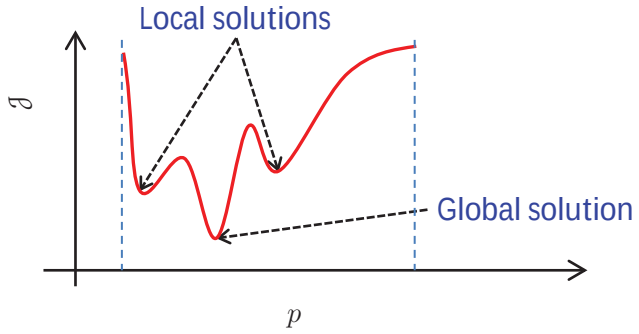

Figure 2.12: Minimization of a function with multiple local optima

which can certify a global solution if run for an indefinitely long time (which is not possible in practice), but cannot estimate or bound the distance between the current best solution and the global solution values after a finite runtime. Therefore, the result obtained after a finite runtime by these algorithms essentially gives no useful information about the global solution value. Examples of asymptotically complete algorithms are those based on sampling schemes (e.g. genetic algorithm [37]).

The third class of algorithms can certify finding a global optimum if given infinite runtime, and can bound the distance between the current best solution and the global solution values after a finite runtime. This is a desired property as it allows the algorithm to run finitely and locate the global solution value within a pre-specified tolerance. This class of algorithms, known as *complete* algorithms [69] is the focus of this research. SBB is a common example of complete algorithms (see §2.5.3).

Note that the bounds on the global solution value in a complete algorithm are guaranteed assuming exact computations. Approximations introduced by numerical solutions (such as truncation errors) and computer arithmetic (i.e. round-off errors) must be accounted for in order to guarantee rigorous results. A complete algorithm that also takes these approximations into account is classified as a *rigorous* algorithm.

2.5.2 Convex Optimization Problem

Prerequisites to the definition of convex problems are the notions of convex sets and convex functions, as defined in §2.3.3. A convex optimization problem is then defined as follows.

Definition 2.13 (Convex Optimization Problem)**.** *Problem* (2.48) *is convex if* D *is a nonempty convex set and* $\mathcal{J}(\mathbf{p})$ *is a convex function on* D*.*

Suppose the feasible set D results from the inequality and equality constraints $\mathbf{g}(\mathbf{p}) \leq 0$ and $\mathbf{s}(\mathbf{p}) = 0$, respectively. Then, the convexity of D requires the functions $\mathbf{g}(.)$ to be convex, and the functions $\mathbf{s}(.)$ to be affine on D [21].

In a convex problem, any local minimum is also a global minimum [25]. Thus, any optimization method that certifies local optimality can be used to obtain a global optimum to a convex problem. This is a fundamental result, which is used in branch-and-bound methods, as discussed in §2.5.3.

2.5.3 Branch-and-Bound Search for Global Optimization

Most general-purpose complete search algorithms rely on the SBB principle [69]. The basic idea in SBB is to find a guaranteed enclosure on the global solution value, and refine this enclosure successively until it is within the desired accuracy. A guaranteed enclosure means a pair of lower bound and upper bound that is theoretically proven to contain the global solution value. For a minimization problem, the upper bound can be any feasible solution, for instance, a local optimum value. A lower bound can be obtained by solving a (linear or nonlinear) convex relaxation problem; that is, a convex optimization problem whose solution underestimates (or is the same as) the global solution of the original problem. Also, the lower bound can be obtained from constant interval bounds[3].

If the initial enclosure is within the desired accuracy, the SBB algorithm stops. Otherwise, it proceeds by branching on the variable set in order to generate smaller

[3]According to Definition 2.3, interval bounds are a special type of convex/concave relaxations. However, in this thesis, the term "relaxation" refers only to those enclosures that are generally not constant over the variable domain.

subproblems (also called nodes), and obtain refined enclosures for each subproblem. The refined enclosures help identify subproblems that are infeasible or guaranteed not to contain a global solution. Specifically, if the convex relaxation turns out to be infeasible, then the corresponding subproblem is infeasible too. Also, if the convex relaxation has an inferior solution to the best upper bound found so far (a.k.a incumbent), then the corresponding subproblem cannot contain a global solution. By eliminating infeasible/inferior subproblems from the search, the SBB procedure gradually narrows down to the global solution value.

To illustrate SBB, consider the following bound constrained minimization:

$$\min_{p\in P} \mathcal{J}(p)$$

where $P := [p^{\mathrm{L}}, p^{\mathrm{U}}]$. A schematic of the function $\mathcal{J}$ is depicted in Fig. 2.13. To begin with, a local search can be performed on P to obtain an upper bound UB_0 on the global solution value. This upper bound is considered the incumbent. Also, a convex relaxation of $\mathcal{J}$ is constructed on P, and solved to give a lower bound LB_0 on the global solution value. These bounds are illustrated on the left plot in Fig. 2.13, which corresponds to the root node (N_0) in the SBB procedure. Since the difference between the two bounds is too big, the variable interval $[p^{\mathrm{L}}, p^{\mathrm{U}}]$ is split into two subintervals $[p^{\mathrm{L}}, p^{m,0}]$ and $[p^{m,0}, p^{\mathrm{U}}]$. There are various ways to choose the branching point $p^{m,0}$ (see §2.5.3.2). Here, it is simply chosen as the midpoint of the interval $[p^{\mathrm{L}}, p^{\mathrm{U}}]$. The bounding procedure is repeated for each resulting node N_1 and N_2. Observe that the local solution UB_2 is superior to UB_0; thus, the incumbent is updated as UB_2. Also, it is seen that the lower bound for Node N_1 is greater than the incumbent. This ensures that Node N_1 cannot contain a global solution. Therefore, it is removed from the search, and the SBB is continued on Node N_2 by branching it into N_3 and N_4. In the branch-and-bound terminology, N_2 is often called the parent node for N_3 and N_4. Also, N_3 and N_4 are siblings to each other, and called child nodes of N_2. The sequence of the nodes examined by SBB can be visualized using the so-called branch-and-bound tree, as illustrated in Fig. 2.14. Observe that the search path on the left of the root node is terminated by removing N_1 from the branch-and-bound tree.

The convergence of SBB exhibits worst-case exponential complexity with the number of optimization variables [69]. Thus, extensive branching is often required to identify infeasible/inferior portions of the variable set. Therefore, being able to generate sufficiently tight, yet computationally cheap, bounds on the subproblems is

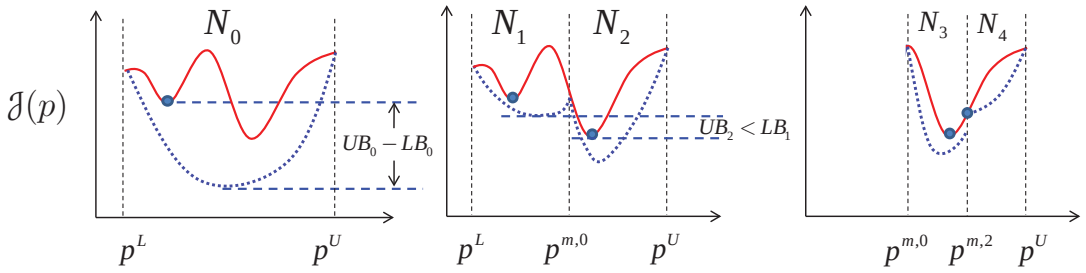

Figure 2.13: Spatial branch-and-bound for a single-variable problem. Left plot: the root node (N_0), center plot: first branching, right plot: second branching. The solid and dotted curves represent the function and its convex relaxation, respectively; the dots on the function represent local solutions at the corresponding nodes.

a major factor on the performance of SBB. Other factors include the choice of the branching variable (in multidimensional problems) [4], the choice of the branching point, the order at which the nodes are solved [29, 50], and the use of domain-reduction techniques. These factors are presented in the subsequent subsections.

2.5.3.1 Domain Reduction Techniques

As mentioned earlier, SBB usually requires a large number of branching operations to shrink the search space and converge to the global solution value. Consequently, the convergence of SBB is slowed down particularly for large-scale problems. To mitigate this issue and accelerate SBB, domain reduction techniques are often employed[4]. These techniques attempt to shrink the variable set corresponding to each subproblem *without* branching it. This is done by identifying and eliminating portions of the variable set that are infeasible or inferior in terms of their optimal solution value. Domain reduction can potentially reduce the number of branchings in two ways (i) it removes the need for branching on the portions of the variable set that are already eliminated due to the shrinking, (ii) it helps compute a tighter lower bound on the optimization problem over the remaining portion. This is due to the fact that the quality of the lower bound on a subproblem depends on the size of its variable set [79]; a smaller variable set can result in a tighter lower bound.

A variety of domain reduction techniques have been proposed in the literature

[4]SBB procedures employing domain reduction techniques are called branch-and-reduce procedures by some researchers [76, 75, 79].

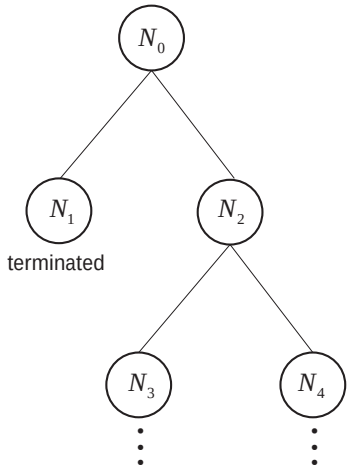

Figure 2.14: The branch-and-bound tree for the example shown in Fig 2.13.

(see e.g. [75, 99, 69, 59]). Constraint propagation can be used as a domain reduction technique, where the problem constraints (or some additional constraints as discussed in §5.2.1) are exploited to identify portions of the variable set in which there exists no feasible solution or no solution better than the incumbent. For example, suppose the optimization problem of interest is subject to the constraint $p_1 p_2 \leq 2$, where p_1,p_2 are the optimization variables with $p_1 \in [0, 6]$ and $p_2 \in [3, 5]$. The constraint implies that

$$p_1^{\mathrm{U}} \leq \frac{2}{p_2^{\mathrm{L}}} = \frac{2}{3}.$$

The upper bound $p_1^{\mathrm{U}} = \frac{2}{3}$ is tighter than the initial value 6. Thus, bounds on p_1 can be reduced to $[0, \frac{2}{3}]$. This simple technique resulted in eliminating more than 88% of the p_1 initial set, without performing a costly branch-and-bound procedure for the removed protion.

There are other techniques that rely on optimization results to exclude inferior portions of the variable set. Ryoo and Sahinidis [75] proposed a number of feasibility- and optimality-based reduction techniques based on the solution of the convex relaxation problem. When this solution lies on the boundary of a particular constraint (including variable bound constraints), the corresponding Lagrange multiplier is used in a simple calculation to potentially shrink the variable set. However, this technique

requires solving an auxiliary convex optimization problem if the solution of the convex problem is not on the boundary of the constraint [75].

Generally speaking, bound reduction for each variable can be posed as its minimization for reducing its lower bound, and its maximization for reducing its upper bound. Both of these optimization problems are subject to a convex relaxation of the original problem constraints and some auxiliary constraints in order to exclude infeasible and inferior portions of the variable set, respectively. See §5.2.1 for more details.

It is common that domain reduction is repeated on a particular node if its previous attempt has been significantly successful in shrinking the variable set. However, some domain reduction techniques are computationally costly. Therefore, a trade-off may be required between the time spent for the domain reduction and the amount of the reduction achieved.

2.5.3.2 Decisions Involved in SBB

A SBB procedure involves making decisions about selecting the next node in the branch-and-bound tree, the branching variable, and the branching point. In this regard, a variety of heuristics have been proposed and studied in the literature. In the following, the most common ones are briefly reviewed.

Selection of next node. During SBB, the nodes that have been created, but not been solved yet are stored in a list of active nodes. Three heuristics for selecting an active node are described below.

- *Depth-first*: In this approach, the node which is deepest in the branch-and-bound tree is solved first [31], i.e. a last-in first-out strategy. When the search path is terminated, the search backtracks to the parent node and applies the same strategy to the sibling nodes of the terminated path [9]. This approach is attractive in terms of memory requirements because the number of nodes that need to be stored in the list grow linearly with respect to the search depth [104]. Nonetheless, it usually needs to explore a large number of nodes before the SBB procedure is terminated.
- *Best-first*: In this approach, active nodes are sorted based on the lower bound value of their parent nodes (assuming a minimization problem) [44]. Then, a

node whose parent has the best (smallest) lower bound is first selected to be solved, and the node list is reordered based on the new results. As a special case of this approach, both sibling nodes of the best parent are solved before reordering the list of active nodes. The best-first strategy increases the likelihood of visiting a node with a global solution early on SBB; although there is no guarantee. Therefore, it can reduce the number of SBB nodes required for convergence. On the downside, this approach has been known for its high memory requirements [104, 44] because it can leave a large number of undecided nodes simultaneously in the branch-and-bound tree.

- *Depth-forward best-back*: This approach performs the depth-first approach along a search path, and when the search path is terminated, starts the new search from the node whose parent has the best lower bound value [9].

Selection of branching variable. A good choice of the branching variable is the one that reduces the total number of SBB nodes [4]. To this end, the merit of each variable can be quantified using a score function; the variable with the highest score is chosen for branching. There are various strategies for the branching variable that differ in their score functions. Three of these strategies are reviewed below.

- *Largest absolute width*: selects the variable with the largest absolute width $p_i^{U,K} - p_i^{L,K}$, with $i = 1, 2, \ldots, n_p$ and K the index of the current node. The absolute width is computed after domain reduction if applied.

- *Largest relative width*: selects the variable with the largest relative width computed as $\frac{p_i^{U,K} - p_i^{L,K}}{p_i^{U,0} - p_i^{L,0}}$, where the denominator represents the original variable bound width at the root node.

- *Largest width reduction*: selects the variable with the largest reduction in its width after performing the domain reduction technique.

There are more advanced strategies that attempt to find a good branching variable by estimating its success in improving the lower bound value of the node. Examples of this type are *pseudocost branching*, *strong branching*, and their combination forms such as *reliability branching* [4]. These strategies often require extra computations, e.g. solving auxiliary optimization problems, at each node. See [4, 50] for technical details.

Selection of branching point. After the branching variable is chosen, a branching point must be selected. Such a point can be taken as the midpoint of the variable interval (bisection approach), the solution point of the incumbent if it is contained within the variable interval (incumbent approach), the point at which the solution of the lower-bounding problem lies (omega approach) [92, 93], or a combination of these.

Chapter3

Convex/Concave Relaxation of Dynamic Models using Interval-based ODE Methods

In this chapter, a new algorithm to compute convex and concave bounds for the solutions of nonlinear parametric ODEs is presented. It can be seen as an extension of the interval ODE method described in §2.4.1 to incorporate convex/concave state bounds. The algorithm is detailed in §3.1, and is demonstrated via numerical examples in §3.2. Finally, the chapter is concluded in §3.3.

3.1 Interval Analysis based State Relaxation Algorithm

In this section, the proposed algorithm for constructing pointwise-in-time convex/-convex relaxations of state variables is presented. The procedure builds upon the

validated ODE solver described in §2.4.1. The algorithm is classified as a *discretize-then-relax* algorithm; that is the ODE system is first discretized in time by the validated solver, and then convex and concave relaxations of the state variables are constructed in each time step. This is to make a distinction with those algorithms based on differential inequalities, where a relaxation of the ODE system is first obtained, and then it is discretized to yield convex/concave state bounds in each time step.

For a specified variable value $\mathbf{p} \in \mathbf{P}$, convex and concave bounds $[\mathbf{x}_j^{\rm cv}, \mathbf{x}_j^{\rm cc}](\mathbf{p})$, are computed at given integration steps t_j, $j = 0, \ldots, n$; that is,

(i) $\mathbf{x}_j^{\rm cv}(\mathbf{p}) \leq \mathbf{x}(t_j; \mathbf{p}) \leq \mathbf{x}_j^{\rm cc}(\mathbf{p})$; and

(ii) $\mathbf{x}_j^{\rm cv}(\mathbf{p})$ and $\mathbf{x}_j^{\rm cc}(\mathbf{p})$ describe, respectively, convex and concave functions when $\mathbf{p}$ is varied in $\mathbf{P}$.

Specifically, the discretize-then-relax procedure starts by constructing the convex/-concave state bounds $[\mathbf{x}_0^{\rm cv}, \mathbf{x}_0^{\rm cc}](\mathbf{p})$ at t_0. Provided that the function $\mathbf{h}$ in (2.2) is factorable, $\mathbf{x}_0^{\rm cv}(\mathbf{p})$ and $\mathbf{x}_0^{\rm cc}(\mathbf{p})$ are easily computed, e.g. by applying the McCormick relaxation technique. Next, the convex/concave state bounds $[\mathbf{x}_j^{\rm cv}, \mathbf{x}_j^{\rm cc}](\mathbf{p})$ are propagated through each integration step t_j, $j = 1, \ldots, n$, based on a two-phase procedure similar to the one described in §2.4.1. The two-phase procedure is illustrated in Fig. 3.1 and the computations in each phase are detailed in the following.

3.1.1 Phase I: A Priori State Convex/Concave Relaxations

Given $\mathbf{p} \in \mathbf{P}$ and convex/concave bounds $[\mathbf{x}_j^{\rm cv}, \mathbf{x}_j^{\rm cc}](\mathbf{p})$ enclosing $\mathbf{x}(t_j; \mathbf{p})$, Phase I computes a priori convex/concave bounds $[\widetilde{\mathbf{x}}_j^{\rm cv}, \widetilde{\mathbf{x}}_j^{\rm cc}](\mathbf{p})$ that enclose $\mathbf{x}(t; \mathbf{p})$ for $t \in [t_j, t_{j+1}]$; that is,

(i) $[\widetilde{\mathbf{x}}_j^{\rm cv}, \widetilde{\mathbf{x}}_j^{\rm cc}](\mathbf{p}) \supseteq \{\mathbf{x}(t; \tau, \boldsymbol{\xi}, \mathbf{p}) : t_j \leq \tau \leq t_{j+1}, x_j^{\rm cv}(\mathbf{p}) \leq \boldsymbol{\xi} \leq x_j^{\rm cc}(\mathbf{p})\}$; and

(ii) $\widetilde{\mathbf{x}}_j^{\rm cv}(\mathbf{p})$ and $\widetilde{\mathbf{x}}_j^{\rm cc}(\mathbf{p})$ describe, respectively, convex and concave functions when $\mathbf{p}$ is varied in $\mathbf{P}$.

The computation of a priori convex/concave bounds requires the following theorem.

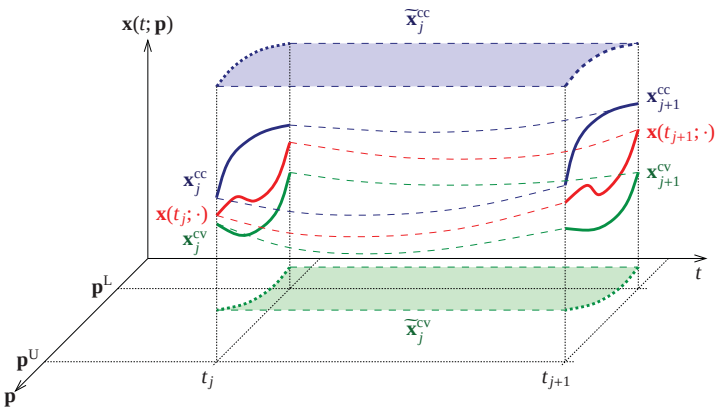

Figure 3.1: Two-phase algorithm for convex/concave relaxations of parametric ODEs illustrated in both time and the variable dimensions. The red curve shows the actual solution set at given times, while the green and blue curves show its convex and concave relaxations, respectively. Similarly, the green and blue surfaces represent a priori convex and concave relaxations, respectively. Observe that they are respectively convex and concave in $\mathbf{p}$ over the *entire* time step. The dash lines between t_j and t_{j+1} are drawn to graphically correspond the results at t_j to those at t_{j+1}, and thus, should not be confused with enclosures.

Theorem 3.1. *Let* $\widetilde{\mathbf{x}}_j^{cv,0}, \widetilde{\mathbf{x}}_j^{cc,0} : \mathbf{P} \to D$ *be, respectively, convex and concave functions on* $\mathbf{P}$. *Let also* $\mathbf{x}_j^{cv}, \mathbf{x}_j^{cc} : \mathbf{P} \to D$ *be, respectively, convex and concave functions on* $\mathbf{P}$, *and such that:*

$$\left[\mathbf{x}_j^{cv}, \mathbf{x}_j^{cc}\right](\mathbf{p}) \subseteq \text{int}\left(\left[\widetilde{\mathbf{x}}_j^{cv,0}, \widetilde{\mathbf{x}}_j^{cc,0}\right](\mathbf{p})\right), \quad \textit{for all } \mathbf{p} \in \mathbf{P}.$$

Suppose that functions $\mathbf{f}^{[i],cv}, \mathbf{f}^{[i],cc} : D \times D \times \mathbf{P} \to \mathbb{R}^{n_x}$, $i = 0, \ldots, k-1$, *are available such that* $\mathbf{f}^{[i],cv}(\mathbf{x}_j^{cv}(\cdot), \mathbf{x}_j^{cc}(\cdot), \cdot)$ *and* $\mathbf{f}^{[i],cc}(\mathbf{x}_j^{cv}(\cdot), \mathbf{x}_j^{cc}(\cdot), \cdot)$ *are, respectively, convex and concave on* $\mathbf{P}$, *and*

$$\mathbf{f}^{[i],cv}\left(\mathbf{x}_j^{cv}(\mathbf{p}), \mathbf{x}_j^{cc}(\mathbf{p}), \mathbf{p}\right) \leq \mathbf{f}^{[i]}(\mathbf{x}_j, \mathbf{p}) \leq \mathbf{f}^{[i],cc}\left(\mathbf{x}_j^{cv}(\mathbf{p}), \mathbf{x}_j^{cc}(\mathbf{p}), \mathbf{p}\right),$$

for all $\mathbf{x}_j \in \left[\mathbf{x}_j^{\rm cv}, \mathbf{x}_j^{\rm cc}\right](\mathbf{p})$ *and all* $\mathbf{p} \in \mathbf{P}$. *Suppose also that functions* $\mathbf{f}^{[k],\rm cv}, \mathbf{f}^{[k],\rm cc} : D \times D \times \mathbf{P} \to \mathbb{R}^{n_x}$ *are available such that* $\mathbf{f}^{[k],\rm cv}(\widetilde{\mathbf{x}}_j^{\rm cv,0}(\cdot), \widetilde{\mathbf{x}}_j^{\rm cc,0}(\cdot), \cdot)$ *and* $\mathbf{f}^{[k],\rm cc}(\widetilde{\mathbf{x}}_j^{\rm cv,0}(\cdot), \widetilde{\mathbf{x}}_j^{\rm cc,0}(\cdot), \cdot)$ *are, respectively, convex and concave on* $\mathbf{P}$, *and*

$$\mathbf{f}^{[k],\rm cv}\left(\widetilde{\mathbf{x}}_j^{\rm cv,0}(\mathbf{p}), \widetilde{\mathbf{x}}_j^{\rm cc,0}(\mathbf{p}), \mathbf{p}\right) \le \mathbf{f}^{[k]}(\mathbf{x}_j, \mathbf{p}) \le \mathbf{f}^{[k],\rm cc}\left(\widetilde{\mathbf{x}}_j^{\rm cv,0}(\mathbf{p}), \widetilde{\mathbf{x}}_j^{\rm cc,0}(\mathbf{p}), \mathbf{p}\right),$$

for all $\mathbf{x}_j \in \left[\widetilde{\mathbf{x}}_j^{\rm cv,0}, \widetilde{\mathbf{x}}_j^{\rm cc,0}\right](\mathbf{p})$ *and all* $\mathbf{p} \in \mathbf{P}$. *If*

$$\begin{aligned}\widetilde{\mathbf{x}}_j^{\rm cv,1}(t,\mathbf{p}) :=& \sum_{i=0}^{k-1}(t-t_j)^i \mathbf{f}^{[i],\rm cv}\left(\mathbf{x}_j^{\rm cv}(\mathbf{p}), \mathbf{x}_j^{\rm cc}(\mathbf{p}), \mathbf{p}\right)\\ &+ (t-t_j)^k \mathbf{f}^{[k],\rm cv}\left(\widetilde{\mathbf{x}}_j^{\rm cv,0}(\mathbf{p}), \widetilde{\mathbf{x}}_j^{\rm cc,0}(\mathbf{p}), \mathbf{p}\right) \ge\ \widetilde{\mathbf{x}}_j^{\rm cv,0}(\mathbf{p}), \qquad (3.1)\\ \widetilde{\mathbf{x}}_j^{\rm cc,1}(t,\mathbf{p}) :=& \sum_{i=0}^{k-1}(t-t_j)^i \mathbf{f}^{[i],\rm cc}\left(\mathbf{x}_j^{\rm cv}(\mathbf{p}), \mathbf{x}_j^{\rm cc}(\mathbf{p}), \mathbf{p}\right)\\ &+ (t-t_j)^k \mathbf{f}^{[k],\rm cc}\left(\widetilde{\mathbf{x}}_j^{\rm cv,0}(\mathbf{p}), \widetilde{\mathbf{x}}_j^{\rm cc,0}(\mathbf{p}), \mathbf{p}\right) \le\ \widetilde{\mathbf{x}}_j^{\rm cc,0}(\mathbf{p}), \qquad (3.2)\end{aligned}$$

for all $\mathbf{p} \in \mathbf{P}$ *and all* $t \in [t_j, t_{j+1}]$, *then*

(i) $\widetilde{\mathbf{x}}_j^{\rm cv,1}(t,\mathbf{p}) \le \mathbf{x}(t; t_j, \mathbf{x}_j, \mathbf{p}) \le \widetilde{\mathbf{x}}_j^{\rm cc,1}(t,\mathbf{p})$, *for all* $\mathbf{x}_j \in [\mathbf{x}_j^{\rm cv}, \mathbf{x}_j^{\rm cc}](\mathbf{p})$, *all* $\mathbf{p} \in \mathbf{P}$, *and all* $t \in [t_j, t_{j+1}]$;

(ii) the functions $\widetilde{\mathbf{x}}_j^{\rm cv,1}(t,\cdot)$ *and* $\widetilde{\mathbf{x}}_j^{\rm cc,1}(t,\cdot)$ *are, respectively, convex and concave on* $\mathbf{P}$ *for each* $t \in [t_j, t_{j+1}]$.

Proof. *The proof of point (ii) follows directly from the convexity/concavity assumptions for* $\mathbf{f}^{[i],\rm cv}(\mathbf{x}_j^{\rm cv}(\cdot), \mathbf{x}_j^{\rm cc}(\cdot), \cdot)$ *and* $\mathbf{f}^{[i],\rm cc}(\mathbf{x}_j^{\rm cv}(\cdot), \mathbf{x}_j^{\rm cc}(\cdot), \cdot)$, $i = 0, \ldots, k-1$, *as well as for* $\mathbf{f}^{[k],\rm cv}(\widetilde{\mathbf{x}}_j^{\rm cv,0}(\cdot), \widetilde{\mathbf{x}}_j^{\rm cc,0}(\cdot), \cdot)$ *and* $\mathbf{f}^{[k],\rm cc}(\widetilde{\mathbf{x}}_j^{\rm cv,0}(\cdot), \widetilde{\mathbf{x}}_j^{\rm cc,0}(\cdot), \cdot)$ *on* $\mathbf{P}$.

Next, a proof of point (i) is given. Let $\underline{\mathbf{p}}$ *be any point in* $\mathbf{P}$, *and let* $\underline{\mathbf{x}}_j \in$

$[\mathbf{x}_j^{\rm cv}, \mathbf{x}_j^{\rm cc}](\underline{\mathbf{p}})$. *By the assumptions on* $\mathbf{f}^{[i],\rm cv}$ *and* $\mathbf{f}^{[i],\rm cc}$,

$$\mathbf{f}^{[i]}(\underline{\mathbf{x}}_j, \underline{\mathbf{p}}) \in \left[\mathbf{f}^{[i],\rm cv}\left(\mathbf{x}_j^{\rm cv}(\underline{\mathbf{p}}), \mathbf{x}_j^{\rm cc}(\underline{\mathbf{p}}), \underline{\mathbf{p}}\right), \mathbf{f}^{[i],\rm cc}\left(\mathbf{x}_j^{\rm cv}(\underline{\mathbf{p}}), \mathbf{x}_j^{\rm cc}(\underline{\mathbf{p}}), \underline{\mathbf{p}}\right)\right],$$

for each $i = 0, \ldots, k-1$, *and by the assumptions on* $\mathbf{f}^{[k],\rm cv}$ *and* $\mathbf{f}^{[k],\rm cc}$,

$$\begin{aligned}&\mathbf{f}^{[k]}\left(\left[\widetilde{\mathbf{x}}_j^{\rm cv,0}(\underline{\mathbf{p}}), \widetilde{\mathbf{x}}_j^{\rm cc,0}(\underline{\mathbf{p}})\right], \underline{\mathbf{p}}\right)\\ &\quad\subseteq \left[\mathbf{f}^{[k],\rm cv}\left(\widetilde{\mathbf{x}}_j^{\rm cv,0}(\underline{\mathbf{p}}), \widetilde{\mathbf{x}}_j^{\rm cc,0}(\underline{\mathbf{p}}), \underline{\mathbf{p}}\right), \mathbf{f}^{[k],\rm cc}\left(\widetilde{\mathbf{x}}_j^{\rm cv,0}(\underline{\mathbf{p}}), \widetilde{\mathbf{x}}_j^{\rm cc,0}(\underline{\mathbf{p}}), \underline{\mathbf{p}}\right)\right].\end{aligned}$$

Adding-up the foregoing $(k+1)$ *inclusions, after pre-multiplying each one of them by the nonnegative scalar* $(t-t_j)^i$, *and then using* (3.1),(3.2), *gives*

$$\begin{aligned}&\sum_{i=0}^{k-1}(t-t_j)^i\mathbf{f}^{[i]}(\underline{\mathbf{x}}_j, \underline{\mathbf{p}}) + (t-t_j)^k\mathbf{f}^{[k]}\left(\left[\widetilde{\mathbf{x}}_j^{\rm cv,0}(\underline{\mathbf{p}}), \widetilde{\mathbf{x}}_j^{\rm cc,0}(\underline{\mathbf{p}})\right], \underline{\mathbf{p}}\right) \qquad (3.3)\\ &\subseteq \left[\widetilde{\mathbf{x}}_j^{\rm cv,1}(t, \underline{\mathbf{p}}), \widetilde{\mathbf{x}}_j^{\rm cc,1}(t, \underline{\mathbf{p}})\right] \subseteq \left[\widetilde{\mathbf{x}}_j^{\rm cv,0}(\underline{\mathbf{p}}), \widetilde{\mathbf{x}}_j^{\rm cc,0}(\underline{\mathbf{p}})\right] \qquad \textit{for all } t \in [t_j, t_{j+1}].\end{aligned}$$

Applying Theorem 2.1, with $\mathbf{z}(\mathbf{x}) \leftarrow \mathbf{f}(\underline{\mathbf{x}}, \underline{\mathbf{p}})$ *and* $\widetilde{\mathbf{X}}_j^0 \leftarrow [\widetilde{\mathbf{x}}_j^{\rm cv,0}(\underline{\mathbf{p}}), \widetilde{\mathbf{x}}_j^{\rm cc,0}(\underline{\mathbf{p}})]$, *gives*

$$\begin{aligned}\mathbf{x}(t; t_j, \underline{\mathbf{x}}_j, \underline{\mathbf{p}}) \in &\sum_{i=0}^{k-1}(t-t_j)^i\mathbf{f}^{[i]}(\underline{\mathbf{x}}_j, \underline{\mathbf{p}}) \qquad (3.4)\\ &+ (t-t_j)^k\mathbf{f}^{[k]}\left(\left[\widetilde{\mathbf{x}}_j^{\rm cv,0}(\underline{\mathbf{p}}), \widetilde{\mathbf{x}}_j^{\rm cc,0}(\underline{\mathbf{p}})\right], \underline{\mathbf{p}}\right),\end{aligned}$$

for all $t \in [t_j, t_{j+1}]$. *In particular, combining* (3.3) *and* (3.4) *shows that* $\mathbf{x}(t; t_j, \underline{\mathbf{x}}_j, \underline{\mathbf{p}}) \in [\widetilde{\mathbf{x}}_j^{\rm cv,1}(t, \underline{\mathbf{p}}), \widetilde{\mathbf{x}}_j^{\rm cc,1}(t, \underline{\mathbf{p}})]$, $t_j \le t \le t_{j+1}$. □

Theorem 3.1 can be seen as the generalization of Theorem 2.1 to encompass convex/concave bounds. A key implication is that it supports the computation of a

priori convex/concave bounds as

$$\begin{aligned}\left[\widetilde{\mathbf{x}}_j^{\mathrm{cv}}, \widetilde{\mathbf{x}}_j^{\mathrm{cc}}\right](\mathbf{p}) = &\sum_{i=0}^{k-1} [0,h_j]^i \odot \left[\mathbf{f}^{[i],\mathrm{cv}}, \mathbf{f}^{[i],\mathrm{cc}}\right]\left(\mathbf{x}_j^{\mathrm{cv}}(\mathbf{p}), \mathbf{x}_j^{\mathrm{cc}}(\mathbf{p}), \mathbf{p}\right) \\ &+ [0,h_j]^k \odot \left[\mathbf{f}^{[k],\mathrm{cv}}, \mathbf{f}^{[k],\mathrm{cc}}\right]\left(\widetilde{\mathbf{x}}_j^{\mathrm{cv},0}(\mathbf{p}), \widetilde{\mathbf{x}}_j^{\mathrm{cc},0}(\mathbf{p}), \mathbf{p}\right),\end{aligned}$$

where $\odot$ stands for the binary product operation in convex/concave bounds arithmetics (see related discussions in §2.3.3.1.). The above relationship is subject to $[\widetilde{\mathbf{x}}_j^{\mathrm{cv},0}, \widetilde{\mathbf{x}}_j^{\mathrm{cc},0}](\mathbf{p})$ and $h_j = t_{j+1} - t_j$ satisfying the conditions (3.1) and (3.2) in Theorem 3.1 for all $\mathbf{p} \in \mathbf{P}$ and all $t \in [t_j, t_{j+1}]$. Unfortunately, checking all of the foregoing conditions (infinitely many) is impractical. Rather, the following corollary provides an alternative way of computing a priori convex/concave bounds that relies on an easy-to-check sufficient condition. It also removes the need for selecting the convex/concave bounds $\widetilde{\mathbf{x}}_j^{\mathrm{cv},0}$ and $\widetilde{\mathbf{x}}_j^{\mathrm{cc},0}$.

Corollary 3.1. *Let* $\widetilde{\mathbf{X}}_j^0 := [\widetilde{\mathbf{x}}_j^{\mathrm{L},0}, \widetilde{\mathbf{x}}_j^{\mathrm{U},0}] \subseteq D$ *and* $\mathbf{X}_j \subseteq \mathrm{int}(\widetilde{\mathbf{X}}_j^0)$*, and let the functions* $\mathbf{x}_j^{\mathrm{cv}}, \mathbf{x}_j^{\mathrm{cc}} : \mathbf{P} \to \mathbf{X}_j$ *be, respectively, convex and concave on* $\mathbf{P}$*. Suppose that functions* $\mathbf{f}^{[i],\mathrm{cv}}, \mathbf{f}^{[i],\mathrm{cc}} : D \times D \times \mathbf{P} \to \mathbb{R}^{n_x}$*,* $i = 0, \ldots, k-1$*, are available as defined in Theorem 3.1 above, and that interval bounds* $\mathbf{F}^{[i]} := [\mathbf{f}^{[i],\mathrm{L}}, \mathbf{f}^{[i],\mathrm{U}}] \supseteq \mathbf{f}^{[i]}(\mathbf{X}_j, \mathbf{P})$*,* $i = 0, \ldots, k-1$*, and* $\widetilde{\mathbf{F}}^{[k]} := [\widetilde{\mathbf{f}}^{[k],\mathrm{L}}, \widetilde{\mathbf{f}}^{[k],\mathrm{U}}] \supseteq \mathbf{f}^{[k]}(\widetilde{\mathbf{X}}_j^0, \mathbf{P})$ *are also available. If*

$$\left\{ \sum_{i=0}^{k-1} (t-t_j)^i \mathbf{F}^{[i]} + (t-t_j)^k \widetilde{\mathbf{F}}^{[k]} : t_j \le t \le t_{j+1} \right\} \subseteq \widetilde{\mathbf{X}}_j^0, \tag{3.5}$$

then

(i) $\widetilde{\mathbf{x}}_j^{\mathrm{cv},2}(t,\mathbf{p}) \le \mathbf{x}(t; t_j, \mathbf{x}_j, \mathbf{p}) \le \widetilde{\mathbf{x}}_j^{\mathrm{cc},2}(t,\mathbf{p})$*, for all* $\mathbf{x}_j \in [\mathbf{x}_j^{\mathrm{cv}}, \mathbf{x}_j^{\mathrm{cc}}](\mathbf{p})$*, all* $\mathbf{p} \in \mathbf{P}$*, and all* $t \in [t_j, t_{j+1}]$*;*

(ii) the functions $\widetilde{\mathbf{x}}_j^{\mathrm{cv},2}(t,\cdot)$ *and* $\widetilde{\mathbf{x}}_j^{\mathrm{cc},2}(t,\cdot)$ *are, respectively, convex and concave on* $\mathbf{P}$ *for each* $t \in [t_j, t_{j+1}]$*;*

with $\widetilde{\mathbf{x}}_j^{\mathrm{cv},2}(t,\mathbf{p})$ *and* $\widetilde{\mathbf{x}}_j^{\mathrm{cc},2}(t,\mathbf{p})$ *defined on* $[t_j,t_{j+1}]\times\mathbf{P}$ *by:*

$$\begin{aligned}\widetilde{\mathbf{x}}_j^{\mathrm{cv},2}(t,\mathbf{p}) :=& \sum_{i=0}^{k-1}(t-t_j)^i \max\left\{\mathbf{f}^{[i],\mathrm{cv}}\left(\mathbf{x}_j^{\mathrm{cv}}(\mathbf{p}),\mathbf{x}_j^{\mathrm{cc}}(\mathbf{p}),\mathbf{p}\right),\mathbf{f}^{[i],\mathrm{L}}\right\}\\ &+(t-t_j)^k\ \widetilde{\mathbf{f}}^{[k],\mathrm{L}}, \qquad (3.6)\\ \widetilde{\mathbf{x}}_j^{\mathrm{cc},2}(t,\mathbf{p}) :=& \sum_{i=0}^{k-1}(t-t_j)^i \min\left\{\mathbf{f}^{[i],\mathrm{cc}}\left(\mathbf{x}_j^{\mathrm{cv}}(\mathbf{p}),\mathbf{x}_j^{\mathrm{cc}}(\mathbf{p}),\mathbf{p}\right),\mathbf{f}^{[i],\mathrm{U}}\right\}\\ &+(t-t_j)^k\ \widetilde{\mathbf{f}}^{[k],\mathrm{U}}. \qquad (3.7)\end{aligned}$$

Proof. *From* (3.5), (3.6), *and* (3.7),

$$\begin{aligned}\widetilde{\mathbf{x}}_j^{\mathrm{cv},2}(t,\mathbf{p}) &\geq \sum_{i=0}^{k-1}(t-t_j)^i\ \mathbf{f}^{[i],\mathrm{L}}+(t-t_j)^k\ \widetilde{\mathbf{f}}^{[k],\mathrm{L}} \geq \widetilde{\mathbf{x}}_j^{\mathrm{L},0}\\ \widetilde{\mathbf{x}}_j^{\mathrm{cc},2}(t,\mathbf{p}) &\leq \sum_{i=0}^{k-1}(t-t_j)^i\ \mathbf{f}^{[i],\mathrm{U}}+(t-t_j)^k\ \widetilde{\mathbf{f}}^{[k],\mathrm{U}} \leq \widetilde{\mathbf{x}}_j^{\mathrm{U},0},\end{aligned}$$

for all $\mathbf{p}\in\mathbf{P}$ *and all* $t\in[t_j,t_{j+1}]$. *Moreover, it is readily checked that the assumptions of Theorem 3.1 are satisfied with the following replacements.*

$$\begin{aligned}&\widetilde{\mathbf{x}}_j^{\mathrm{cv},0}(\mathbf{p})\leftarrow\widetilde{\mathbf{x}}_j^{\mathrm{L},0},\\ &\widetilde{\mathbf{x}}_j^{\mathrm{cc},0}(\mathbf{p})\leftarrow\widetilde{\mathbf{x}}_j^{\mathrm{U},0},\\ &\mathbf{f}^{[i],\mathrm{cv}}\left(\mathbf{x}_j^{\mathrm{cv}}(\mathbf{p}),\mathbf{x}_j^{\mathrm{cc}}(\mathbf{p}),\mathbf{p}\right)\leftarrow\max\left\{\mathbf{f}^{[i],\mathrm{cv}}\left(\mathbf{x}_j^{\mathrm{cv}}(\mathbf{p}),\mathbf{x}_j^{\mathrm{cc}}(\mathbf{p}),\mathbf{p}\right),\mathbf{f}^{[i],\mathrm{L}}\right\},\\ &\mathbf{f}^{[i],\mathrm{cc}}\left(\mathbf{x}_j^{\mathrm{cv}}(\mathbf{p}),\mathbf{x}_j^{\mathrm{cc}}(\mathbf{p}),\mathbf{p}\right)\leftarrow\min\left\{\mathbf{f}^{[i],\mathrm{cc}}\left(\mathbf{x}_j^{\mathrm{cv}}(\mathbf{p}),\mathbf{x}_j^{\mathrm{cc}}(\mathbf{p}),\mathbf{p}\right),\mathbf{f}^{[i],\mathrm{U}}\right\},\\ &\mathbf{f}^{[k],\mathrm{cv}}\left(\mathbf{x}_j^{\mathrm{cv}}(\mathbf{p}),\mathbf{x}_j^{\mathrm{cc}}(\mathbf{p}),\mathbf{p}\right)\leftarrow\widetilde{\mathbf{f}}^{[k],\mathrm{L}},\\ &\mathbf{f}^{[k],\mathrm{cc}}\left(\mathbf{x}_j^{\mathrm{cv}}(\mathbf{p}),\mathbf{x}_j^{\mathrm{cc}}(\mathbf{p}),\mathbf{p}\right)\leftarrow\widetilde{\mathbf{f}}^{[k],\mathrm{U}}.\end{aligned}$$

In particular, a proof of the properties (i) and (ii) directly follows from the application

of Theorem 3.1. □

The main practical implication of Corollary 3.1 is that valid a priori convex/concave bounds can be obtained as:

$$\begin{aligned}\left[\widetilde{\mathbf{x}}_j^{\mathrm{cv}},\widetilde{\mathbf{x}}_j^{\mathrm{cc}}\right](\mathbf{p}) = &\sum_{i=0}^{k-1}[0,h_j]^i \odot \left\{\left[\mathbf{f}^{[i],\mathrm{cv}},\mathbf{f}^{[i],\mathrm{cc}}\right]\left(\mathbf{x}_j^{\mathrm{cv}}(\mathbf{p}),\mathbf{x}_j^{\mathrm{cc}}(\mathbf{p}),\mathbf{p}\right)\cap\mathbf{F}^{[i]}\right\}\\ &+[0,h_j]^k\mathbf{f}^{[k]}(\widetilde{\mathbf{X}}_j^0,\mathbf{P}),\end{aligned} \tag{3.8}$$

provided that the interval $\widetilde{\mathbf{X}}_j^0$ and the step size $h_j = t_{j+1} - t_j$ satisfy the condition (3.5). This latter condition is also sufficient to establish the existence and uniqueness of the solutions of (2.1) on $[t_j, t_{j+1}]$, for any initial value $\mathbf{x}_j \in \mathbf{X}_j$ at t_j and any variable value $\mathbf{p} \in \mathbf{P}$.

Clearly, the computation of a priori convex/concave state bounds via (3.8) hinges on the ability to compute convex/concave bounds for Taylor coefficients of the form $\mathbf{f}^{[i]}(\boldsymbol{\xi}(\mathbf{p}),\mathbf{p})$, $i = 1,\ldots,k$. Observe that, since $\mathbf{f}$ is factorable and k-times continuously differentiable, the outer functions $\mathbf{f}^{[i]}$ as obtained from (2.26) are themselves factorable. On the other hand, the inner function $\boldsymbol{\xi}$ is generally nonfactorable since an explicit solution in typically unavailable for the parametric ODEs. Yet, convex/concave bounds $[\boldsymbol{\xi}^{\mathrm{cv}},\boldsymbol{\xi}^{\mathrm{cc}}](\mathbf{p})$ are known for the inner function from previous computations. A way to compute the required convex/concave bounds $[\mathbf{f}^{[i],\mathrm{cv}},\mathbf{f}^{[i],\mathrm{cc}}](\boldsymbol{\xi}^{\mathrm{cv}}(\mathbf{p}),\boldsymbol{\xi}^{\mathrm{cc}}(\mathbf{p}),\mathbf{p})$ is therefore by applying the (generalized) McCormick technique, as discussed in §2.3.3.1. Interestingly, because such convex/concave relaxations are guaranteed to be no looser than the underlying interval bounds [90], the computation of a priori convex/concave bounds with the McCormick technique can be further simplified as:

$$\begin{aligned}\left[\widetilde{\mathbf{x}}_j^{\mathrm{cv}},\widetilde{\mathbf{x}}_j^{\mathrm{cc}}\right](\mathbf{p}) = &\sum_{i=0}^{k-1}[0,h_j]^i \odot \left[\mathbf{f}^{[i],\mathrm{cv}},\mathbf{f}^{[i],\mathrm{cc}}\right]\left(\mathbf{x}_j^{\mathrm{cv}}(\mathbf{p}),\mathbf{x}_j^{\mathrm{cc}}(\mathbf{p}),\mathbf{p}\right)\\ &+[0,h_j]^k\mathbf{f}^{[k]}(\widetilde{\mathbf{X}}_j^0,\mathbf{P}).\end{aligned} \tag{3.9}$$

Another prerequisite for the computation of convex/concave bounds on $[t_j, t_{j+1}]$ is the availability of interval bounds $\mathbf{X}_j$ enclosing the solutions of (2.1),(2.2) at t_j. Therefore, the proposed state relaxation algorithm must not only propagate convex/concave bounds on the ODE solutions at each integration step t_j, but also

simple interval enclosures. In this work, this is done by applying the two-phase approach described earlier in §2.4.1.2.

3.1.2 Phase II: Tight State Convex/Concave Relaxations

Given $\mathbf{p} \in \mathbf{P}$, convex/concave bounds $[\mathbf{x}_j^{\rm cv}, \mathbf{x}_j^{\rm cc}](\mathbf{p})$ for $\mathbf{x}(t_j;\mathbf{p})$, and a priori convex/-concave bounds $[\widetilde{\mathbf{x}}_j^{\rm cv}, \widetilde{\mathbf{x}}_j^{\rm cc}](\mathbf{p})$ for $\mathbf{x}(t;\mathbf{p})$ on $[t_j, t_{j+1}]$, Phase II computes convex/concave bounds $[\widetilde{\mathbf{x}}_{j+1}^{\rm cv}, \widetilde{\mathbf{x}}_{j+1}^{\rm cc}](\mathbf{p})$ that enclose $\mathbf{x}(t_{j+1};\mathbf{p})$; that is,

(i) $\mathbf{x}_{j+1}^{\rm cv}(\mathbf{p}) \le \mathbf{x}(t_{j+1};\mathbf{p}) \le \mathbf{x}_{j+1}^{\rm cc}(\mathbf{p})$; and

(ii) $\mathbf{x}_{j+1}^{\rm cv}(\mathbf{p})$ and $\mathbf{x}_{j+1}^{\rm cc}(\mathbf{p})$ describe, respectively, convex and concave functions when $\mathbf{p}$ is varied in $\mathbf{P}$.

Let $\mathbf{X}_j$ be intervals state bounds for $\mathbf{x}(t_j;\mathbf{p})$; for example, such bounds can be obtained from the procedure outlined in §2.4.1.2. The derivation of convex/concave state bounds is based on the mean-value form of the Taylor expansion of the solutions to (2.1) between t_j and t_{j+1}. Given any reference point $(\hat{\mathbf{x}}_j, \hat{\mathbf{p}}) \in \mathbf{X}_j \times \mathbf{P}$, there exist $\tau_\iota \in [0, h_j]$ and $\eta_\iota^i \in [0,1]$ for $\iota = 1,\ldots,n_x$ and $i = 0,\ldots,k-1$ such that:

$$\begin{aligned} x_\iota(t_{j+1};\mathbf{p}) = \sum_{i=0}^{k-1} h_j^i \Bigg(& f_\iota^{[i]}(\hat{\mathbf{x}}_j,\hat{\mathbf{p}}) + \frac{\partial f_\iota^{[i]}}{\partial \mathbf{x}}(\boldsymbol{\mu}_{\iota,j}^i(\mathbf{p}), \boldsymbol{\rho}_\iota^i(\mathbf{p}))\,[\mathbf{x}(t_j) - \hat{\mathbf{x}}_j] \\ & + \frac{\partial f_\iota^{[i]}}{\partial \mathbf{p}}(\boldsymbol{\mu}_{\iota,j}^i(\mathbf{p}), \boldsymbol{\rho}_\iota^i(\mathbf{p}))\,[\mathbf{p} - \hat{\mathbf{p}}] \Bigg) \\ & + h_j^k f_\iota^{[k]}(\mathbf{x}(t_j+\tau_\iota;\mathbf{p}),\mathbf{p}), \end{aligned} \tag{3.10}$$

with the functions $\boldsymbol{\mu}_{\iota,j}^i : \mathbf{P} \to \mathbf{X}_j$ and $\boldsymbol{\rho}_\iota^i : \mathbf{P} \to \mathbf{P}$ given by:

$$\boldsymbol{\mu}_{\iota,j}^i(\mathbf{p}) = \hat{\mathbf{x}}_j + \eta_\iota^i\,(\mathbf{x}(t_j;\mathbf{p}) - \hat{\mathbf{x}}_j) \tag{3.11}$$

$$\boldsymbol{\rho}_\iota^i(\mathbf{p}) = \hat{\mathbf{p}} + \eta_\iota^i\,(\mathbf{p} - \hat{\mathbf{p}})\,. \tag{3.12}$$

From (3.10), it becomes clear that convex/concave state bounds at t_{j+1} can be computed as

$$\begin{aligned}\left[\mathbf{x}_{j+1}^{\mathrm{cv}}, \mathbf{x}_{j+1}^{\mathrm{cc}}\right](\mathbf{p}) = &\sum_{i=0}^{k-1} h_j^i \mathbf{f}^{[i]}(\hat{\mathbf{x}}_j, \hat{\mathbf{p}}) + h_j^k \left[\mathbf{f}^{[k],\mathrm{cv}}, \mathbf{f}^{[k],\mathrm{cc}}\right]\left(\widetilde{\mathbf{x}}_j^{\mathrm{cv}}(\mathbf{p}), \widetilde{\mathbf{x}}_j^{\mathrm{cc}}(\mathbf{p}), \mathbf{p}\right) \\ &+ \sum_{i=0}^{k-1} h_j^i \left[\frac{\partial \mathbf{f}^{[i],\mathrm{cv}}}{\partial \mathbf{x}}, \frac{\partial \mathbf{f}^{[i],\mathrm{cc}}}{\partial \mathbf{x}}\right]\Big(\left[\boldsymbol{\mu}_j^{\mathrm{cv}}, \boldsymbol{\mu}_j^{\mathrm{cc}}\right](\mathbf{p}), \left[\boldsymbol{\rho}^{\mathrm{cv}}, \boldsymbol{\rho}^{\mathrm{cc}}\right](\mathbf{p})\Big) \\ &\qquad \odot \left(\left[\mathbf{x}_j^{\mathrm{cv}}, \mathbf{x}_j^{\mathrm{cc}}\right](\mathbf{p}) - \hat{\mathbf{x}}_j\right) \\ &+ \sum_{i=0}^{k-1} h_j^i \left[\frac{\partial \mathbf{f}^{[i],\mathrm{cv}}}{\partial \mathbf{p}}, \frac{\partial \mathbf{f}^{[i],\mathrm{cc}}}{\partial \mathbf{p}}\right]\left(\left[\boldsymbol{\mu}_j^{\mathrm{cv}}, \boldsymbol{\mu}_j^{\mathrm{cc}}\right](\mathbf{p}), \left[\boldsymbol{\rho}^{\mathrm{cv}}, \boldsymbol{\rho}^{\mathrm{cc}}\right](\mathbf{p})\right) \\ &\qquad \odot (\mathbf{p} - \hat{\mathbf{p}})\end{aligned} \tag{3.13}$$

where the functions $\boldsymbol{\mu}_j^{\mathrm{cv}}, \boldsymbol{\mu}_j^{\mathrm{cc}} : \mathbf{P} \to \mathbf{X}_j$ and $\boldsymbol{\rho}^{\mathrm{cv}}, \boldsymbol{\rho}^{\mathrm{cc}} : \mathbf{P} \to \mathbf{P}$ are given by

$$\begin{aligned}&\left[\boldsymbol{\mu}_j^{\mathrm{cv}}, \boldsymbol{\mu}_j^{\mathrm{cc}}\right](\mathbf{p}) = \hat{\mathbf{x}}_j + [0,1] \odot \left(\left[\mathbf{x}_j^{\mathrm{cv}}, \mathbf{x}_j^{\mathrm{cc}}\right](\mathbf{p}) - \hat{\mathbf{x}}_j\right), \\ &\left[\boldsymbol{\rho}^{\mathrm{cv}}, \boldsymbol{\rho}^{\mathrm{cc}}\right](\mathbf{p}) = \hat{\mathbf{p}} + [0,1] \odot (\mathbf{p} - \hat{\mathbf{p}}).\end{aligned}$$

Interestingly, the interval-bound counterparts of $[\boldsymbol{\mu}_j^{\mathrm{cv}}, \boldsymbol{\mu}_j^{\mathrm{cc}}]$ and $[\boldsymbol{\rho}^{\mathrm{cv}}, \boldsymbol{\rho}^{\mathrm{cc}}]$ can be easily shown to be $\mathbf{X}_j$ and $\mathbf{P}$, respectively, which is consistent with the expression (2.37) giving $\mathbf{X}_{j+1}$.

The terms $\frac{\partial \mathbf{f}^{[i],\mathrm{cv}}}{\partial \mathbf{x}}$, $\frac{\partial \mathbf{f}^{[i],\mathrm{cc}}}{\partial \mathbf{x}}$ and $\frac{\partial \mathbf{f}^{[i],\mathrm{cv}}}{\partial \mathbf{p}}$, $\frac{\partial \mathbf{f}^{[i],\mathrm{cc}}}{\partial \mathbf{p}}$ in (3.13) stand for convex/concave bounds of the partial derivatives $\frac{\partial \mathbf{f}^{[i]}}{\partial \mathbf{x}}$ and $\frac{\partial \mathbf{f}^{[i]}}{\partial \mathbf{p}}$ of the Taylor coefficients, respectively, for each $i = 0, \ldots, k-1$. Such convex/concave bounds can be computed, similar to convex/concave bounds for the Taylor coefficients $\mathbf{f}^{[i]}$, by applying the (generalized) McCormick technique, provided that convex/concave bounds are known for their arguments (see §2.3.3.1).

For brevity, denote

$$\left[\mathbf{J}^{\mathbf{x},\mathrm{cv}}_{j+1},\mathbf{J}^{\mathbf{x},\mathrm{cc}}_{j+1}\right](\mathbf{p}) = \sum_{i=0}^{k-1} h_j^i \left[\frac{\partial \mathbf{f}^{[i],\mathrm{cv}}}{\partial \mathbf{x}},\frac{\partial \mathbf{f}^{[i],\mathrm{cc}}}{\partial \mathbf{x}}\right]\left(\left[\boldsymbol{\mu}^{\mathrm{cv}}_j,\boldsymbol{\mu}^{\mathrm{cc}}_j\right](\mathbf{p}),\left[\boldsymbol{\rho}^{\mathrm{cv}},\boldsymbol{\rho}^{\mathrm{cc}}\right](\mathbf{p})\right)$$

$$\left[\mathbf{J}^{\mathbf{p},\mathrm{cv}}_{j+1},\mathbf{J}^{\mathbf{p},\mathrm{cc}}_{j+1}\right](\mathbf{p}) = \sum_{i=0}^{k-1} h_j^i \left[\frac{\partial \mathbf{f}^{[i],\mathrm{cv}}}{\partial \mathbf{p}},\frac{\partial \mathbf{f}^{[i],\mathrm{cc}}}{\partial \mathbf{p}}\right]\left(\left[\boldsymbol{\mu}^{\mathrm{cv}}_j,\boldsymbol{\mu}^{\mathrm{cc}}_j\right](\mathbf{p}),\left[\boldsymbol{\rho}^{\mathrm{cv}},\boldsymbol{\rho}^{\mathrm{cc}}\right](\mathbf{p})\right)$$

$$\left[\mathbf{R}^{\mathrm{cv}}_{j+1},\mathbf{R}^{\mathrm{cc}}_{j+1}\right](\mathbf{p}) = h_j^k \left[\mathbf{f}^{[k],\mathrm{cv}},\mathbf{f}^{[k],\mathrm{cc}}\right]\left(\widetilde{\mathbf{x}}^{\mathrm{cv}}_j(\mathbf{p}),\widetilde{\mathbf{x}}^{\mathrm{cc}}_j(\mathbf{p}),\mathbf{p}\right).$$

As with interval state bounds, the product term $[\mathbf{J}^{\mathbf{x},\mathrm{cv}}_{j+1},\mathbf{J}^{\mathbf{x},\mathrm{cc}}_{j+1}](\mathbf{p})\odot([\mathbf{x}^{\mathrm{cv}}_j,\mathbf{x}^{\mathrm{cc}}_j](\mathbf{p})-\hat{\mathbf{x}}_j)$ is a major contributor to the wrapping effect. To mitigate it, the order of the operations can be changed according to:

$$\begin{aligned}\left[\boldsymbol{\Delta}^{\mathrm{cv}}_{j+1},\boldsymbol{\Delta}^{\mathrm{cc}}_{j+1}\right](\mathbf{p}) = {}& \left[\mathbf{A}^{-1}_{j+1}\odot\left(\left[\mathbf{J}^{\mathbf{x},\mathrm{cv}}_{j+1},\mathbf{J}^{\mathbf{x},\mathrm{cc}}_{j+1}\right](\mathbf{p})\odot\mathbf{A}_j\right)\right]\odot\left[\boldsymbol{\Delta}^{\mathrm{cv}}_j,\boldsymbol{\Delta}^{\mathrm{cc}}_j\right](\mathbf{p})\\ &+\mathbf{A}^{-1}_{j+1}\odot\left(\left[\mathbf{R}^{\mathrm{cv}}_{j+1},\mathbf{R}^{\mathrm{cc}}_{j+1}\right](\mathbf{p})-\mathrm{m}(\mathbf{R}_{j+1})\right) \qquad (3.14)\\ &+\mathbf{A}^{-1}_{j+1}\odot\left(\left[\mathbf{J}^{\mathbf{p},\mathrm{cv}}_{j+1},\mathbf{J}^{\mathbf{p},\mathrm{cc}}_{j+1}\right](\mathbf{p})\right)\odot(\mathbf{p}-\hat{\mathbf{p}})\\ \left[\mathbf{x}^{\mathrm{cv}}_{j+1},\mathbf{x}^{\mathrm{cc}}_{j+1}\right](\mathbf{p}) = {}& \mathbf{v}_{j+1}+\left[\mathbf{R}^{\mathrm{cv}}_{j+1},\mathbf{R}^{\mathrm{cc}}_{j+1}\right](\mathbf{p})\\ &+\left(\left[\mathbf{J}^{\mathbf{x},\mathrm{cv}}_{j+1},\mathbf{J}^{\mathbf{x},\mathrm{cc}}_{j+1}\right](\mathbf{p})\odot\mathbf{A}_j\right)\odot\left[\boldsymbol{\Delta}^{\mathrm{cv}}_j,\boldsymbol{\Delta}^{\mathrm{cc}}_j\right](\mathbf{p}) \qquad (3.15)\\ &+\left[\mathbf{J}^{\mathbf{p},\mathrm{cv}}_{j+1},\mathbf{J}^{\mathbf{p},\mathrm{cc}}_{j+1}\right](\mathbf{p})\odot[\mathbf{p}-\hat{\mathbf{p}}],\end{aligned}$$

with $\mathbf{v}_{j+1}$ and $\mathbf{R}_{j+1}$ defined by (2.37), $\hat{\mathbf{x}}_{j+1}$ given by (2.33), and $\mathbf{A}_{j+1}$ taken as the orthogonal matrix in the QR decomposition of $\mathrm{m}(\mathbf{J}^{\mathbf{x}}_{j+1}\mathbf{A}_j)$. Like in Phase I, a prerequisite to the application of Phase II of the proposed state relaxation algorithm is therefore that interval bounds $\mathbf{X}_j$ and $\widetilde{\mathbf{X}}_j$ be available at each integration step; see §2.4.1.2.

3.1.2.1 Further Tightening via Affine Relaxations

Both Phase I and Phase II in the proposed state relaxation algorithm require that interval state bounds be propagated along with the convex/concave bounds at each integration step. Therefore, the tightness of the resulting relaxations directly depends upon the tightness of the underlying interval bounds. A tightening procedure, wherein the state relaxations at a given step are used to refine the interval state bounds at that step, is presented in this subsection. This procedure can therefore

be seen as a feedback mechanism from the convex/concave bounds on the interval bounds.

Let $\mathbf{p}^{\text{ref}} \in \text{int}(\mathbf{P})$ be a (constant) reference point, and let $\mathbf{x}^{\text{cv}}_{j+1}$ and $\mathbf{x}^{\text{cc}}_{j+1}$ denote, respectively, convex and concave relaxations for $\mathbf{x}(t_{j+1};\cdot)$ on $\mathbf{P}$. Suppose that a subgradient vector $\boldsymbol{\sigma}^{\text{cv}}_{j+1}$ of $\mathbf{x}^{\text{cv}}_{j+1}$ at $\mathbf{p}^{\text{ref}}$ and a subgradient vector $\boldsymbol{\sigma}^{\text{cc}}_{j+1}$ of $\mathbf{x}^{\text{cc}}_{j+1}$ at $\mathbf{p}^{\text{ref}}$ are available. Note that the existence of subgradients in the interior of $\mathbf{P}$ is guaranteed, for the functions $\mathbf{x}^{\text{cv}}_{j+1}$ and $\mathbf{x}^{\text{cc}}_{j+1}$ are convex and a concave on $\mathbf{P}$.

Affine relaxations $\mathbf{x}^{\text{cv,l}}_{j+1}, \mathbf{x}^{\text{cc,l}}_{j+1} : \mathbf{P} \to \mathbb{R}^{n_x}$ are readily obtained as:

$$\begin{aligned}
\mathbf{x}^{\text{cv,l}}_{j+1}(\mathbf{p}) &= \mathbf{x}^{\text{cv}}_{j+1}(\mathbf{p}^{\text{ref}}) + (\boldsymbol{\sigma}^{\text{cv}}_{j+1})^{\mathsf{T}}(\mathbf{p} - \mathbf{p}^{\text{ref}}),\\
\mathbf{x}^{\text{cc,l}}_{j+1}(\mathbf{p}) &= \mathbf{x}^{\text{cc}}_{j+1}(\mathbf{p}^{\text{ref}}) + (\boldsymbol{\sigma}^{\text{cc}}_{j+1})^{\mathsf{T}}(\mathbf{p} - \mathbf{p}^{\text{ref}}).
\end{aligned} \tag{3.16}$$

These relaxations provide a cheap way to obtain affine –and thus convex/concave– bounds $[\mathbf{x}^{\text{cv,l}}_{j+1}, \mathbf{x}^{\text{cc,l}}_{j+1}](\mathbf{p})$ at *every* point $\mathbf{p} \in \mathbf{P}$, based solely on the computation of convex/concave state bounds and subgradients at $\mathbf{p}^{\text{ref}}$. Because,

$$\inf\left\{\mathbf{x}^{\text{cv,l}}_{j+1}(\boldsymbol{\theta}) : \boldsymbol{\theta} \in \mathbf{P}\right\} \leq \mathbf{x}(t_{j+1};\mathbf{p}) \leq \sup\left\{\mathbf{x}^{\text{cc,l}}_{j+1}(\boldsymbol{\theta}) : \boldsymbol{\theta} \in \mathbf{P}\right\},$$

for each $\mathbf{p} \in \mathbf{P}$, the affine relaxations (3.16) can also be used to compute valid interval bounds for $\mathbf{x}(t_{j+1};\cdot)$ on $\mathbf{P}$. In particular, $\mathbf{x}^{\text{cv,l}}_{j+1}$ and $\mathbf{x}^{\text{cc,l}}_{j+1}$ attain their infimum and supremum on $\mathbf{P}$, respectively, at the points $\mathbf{p}^{\text{cv,l}}$ and $\mathbf{p}^{\text{cc,l}}$ given by:

$$p_i^{\text{cv,l}} = \begin{cases} p_i^{\text{L}} & \text{if } \boldsymbol{\sigma}^{\text{cv}}_{j+1} \geq 0,\\ p_i^{\text{U}} & \text{otherwise,}\end{cases} \qquad p_i^{\text{cc,l}} = \begin{cases} p_i^{\text{L}} & \text{if } \boldsymbol{\sigma}^{\text{cc}}_{j+1} \leq 0,\\ p_i^{\text{U}} & \text{otherwise,}\end{cases}$$

for each $i = 1, \ldots, n_p$. The resulting relaxation-based bounds

$$\mathbf{X}^{\ell}_{j+1} := [\mathbf{x}^{\text{cv,l}}_{j+1}(\mathbf{p}^{\text{cv,l}}), \mathbf{x}^{\text{cc,l}}_{j+1}(\mathbf{p}^{\text{cc,l}})]$$

can then be intersected with the interval bounds $\mathbf{X}_{j+1}$ obtained from (2.37), thereby yielding refined state bounds as $\mathbf{X}_{j+1} \cap \mathbf{X}^{\ell}_{j+1}$.

Observe that any reference point $\mathbf{p}^{\text{ref}}$ in the interior of $\mathbf{P}$ can be selected to generate affine relaxations, and the choice of the reference point matters since different

reference points result in different relaxation-based bounds. In order to increase the likelihood of improving the interval state bounds, it is of course possible to consider multiple reference points and repeat the tightening procedure for each of these points. It should be clear, however, that this is at the expense of computing convex/concave bounds and subgradients at every reference point. These considerations are illustrated by a numerical case study in the subsequent section.

3.2 Numerical Case Studies

In this section, the discretize-then-relax algorithm is demonstrated by two case studies. The first example, a simple scalar ODE problem, is used to illustrate both Phase I and Phase II of the algorithm, the convex/concave bounds and their subgradients, and the tightening procedure based on affine relaxations. Comparisons with the solvers VNODE-LP [63], which implements an interval ODE method, and VSPODE [49], which implements a Taylor model ODE method, are also presented for this example. The interval ODE method in VNODE-LP is a more elaborate variant of the one presented in §2.4.1, in which Phase II incorporates an interval Hermite-Obreschkoff method [64] to enable tighter enclosures. Also, the Taylor model ODE method in VSPODE is the one described by (2.36) and (2.47) with the wrapping effect in Phase II treated by the parallelepiped approach as presented in [49]. The second case study is a prey-predator model (Lotka-Volterra system), which is a popular benchmark problem for validated ODE solvers [49]. Both a single variable and a modified multiple variable problem are investigated here.

A Taylor expansion order of $k = 10$ is used in the numerical case studies, unless otherwise noted. The integration stepsizes are controlled by the *local excess per unit step* (LEPUS) approach [66]. This approach maintains a balance between a faster integration (but more overestimation due to larger steps) and less overestimation (but slower integration due to smaller steps). Unless specified otherwise, the initial stepsize $h_0 = 0.01$ and absolute and relative tolerances of 10^{-5} are used in the LEPUS approach. The computations are performed on a PC with Intel Core 2 Duo 2.4GHz CPU, 4GB RAM, and running Linux and gcc version 4.3.2.

3.2.1 A Simple Scalar ODE

Consider the following scalar IVP in ODEs with a single variable [70, 26, 49]:

$$\dot{x}(t;p) = -x(t;p)^2 + p; \quad x(0;p) = 9 \tag{3.17}$$

with $t \in [0,1]$, and the variable $p \in P := [-1,1]$.

Because the discretize-then-relax algorithm builds upon interval ODE methods, the tightness of the underlying interval bounds directly affects the tightness of the convex and concave relaxations. The results of both the a priori (Phase I) and tightened (Phase II) state bounds are shown in Fig. 3.2, along with the actual solution set of (3.17). As is typical with interval ODE solvers, the computed state bounds are conservative, although the overestimation remains limited in this case. Notice also the effect of the stepsize control strategy, which takes less than 20 steps to reach the final time $t = 1$, with the default settings.

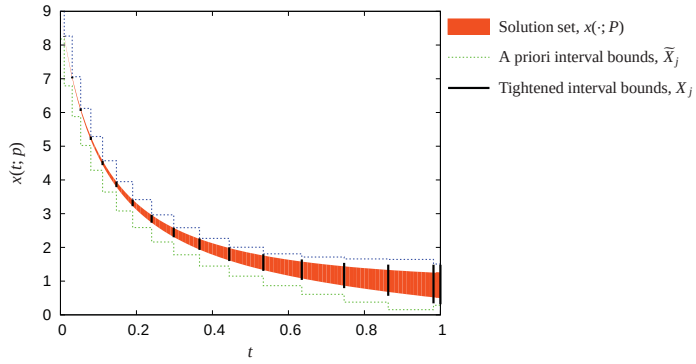

Figure 3.2: A priori and tightened interval bounds for the state variable x in the scalar ODE problem (3.17).

Convex and concave bounds $[x_j^{\mathrm{cv}}, x_j^{\mathrm{cc}}](p)$ are propagated, along with interval bounds X_j, at each integration step. Comparisons between interval bounds and convex/concave bounds are reported in Table 3.1, for different Taylor series expansion orders k and a constant stepsize of $h = 0.01$. Observe first that the order k has no appreciable effect on the resulting interval enclosure width, probably due to the small stepsize used here. Naturally, increasing the expansion order k increases

the computational time for performing an integration step; roughly, a 3-fold increase between $k = 5$ and $k = 20$. It is also seen that the overhead for propagating convex/-concave bounds along with interval bounds roughly doubles the integration time.

A second comparison is shown in Table 3.2, between the proposed discretize-then-relax approach and the solvers VNODE-LP [63] and VSPODE [49]. In all three solvers, a Taylor series expansion order of $k = 10$ is used, and various absolute/relative tolerances are considered for stepsize adaptation (the LEPUS approach [66]). Quite expectedly, a smaller stepsize tolerance results in a tighter enclosure, yet it requires a larger number of integration steps and, therefore, a higher computational burden. Because the interval ODE method used to develop the relax-then-discretize approach is rather basic, i.e. it does not implement an interval Hermite-Obreschkoff method as VNODE-LP or a Taylor model method as VSPODE, the interval enclosures computed with VNODE-LP and VSPODE are consistently tighter. In terms of the computational effort, it is found that the proposed approach is superior to VSPODE, but about three-times slower than VNODE-LP. The difference with VNODE-LP is not entirely caused by the overhead of computing convex/concave bounds because such overhead only doubles the computational time (see Table 3.1). Furthermore, the interval method used in VNODE-LP is more involved, and should be more expensive than the one used in this work. Rather, the difference with VNODE-LP is partly attributed to the fewer integration steps in VNODE-LP and a different computer implementation.

The convex and concave relaxations of $x(t_j;\cdot)$ can be plotted by computing the convex and concave bounds $[x_j^{\rm cv}, x_j^{\rm cc}](p)$ repeatedly at a large number of points $p \in P$. The convex and concave relaxations obtained this way at $t = 1$ are shown in Fig. 3.3. It can be seen that they describe convex and concave functions and validly enclose the solutions on P. Moreover, not only are the convex/concave bounds contained in

Table 3.1: Comparison between interval and convex/concave bounds in Problem (3.17), for Taylor series expansion orders $k = 5, 7, 10, 20$ and a constant stepsize of $h = 0.01$ (100 steps).

Taylor Series Expansion Order	5	7	10	20
Interval Width at $t = 1$	0.914	0.914	0.914	0.914
CPU Time for Interval Bounds	5.3 ms	6.4 ms	8.1 ms	14.3 ms
CPU Time for Convex/Concave Bounds	8.7 ms	11.0 ms	14.5 ms	29.0 ms

Table 3.2: Comparison between the proposed discretize-then-relax approach and the solvers VNODE-LP [63] and VSPODE [49] in Problem (3.17), for various (absolute and relative) tolerances of stepsize adaptation and a Taylor series expansion order of $k = 10$.

Method	Discretize-then-Relax			VNODE-LP			VSPODE[a]		
Stepsize Tolerance	10^{-5}	10^{-7}	10^{-9}	10^{-5}	10^{-7}	10^{-9}	10^{-5}	10^{-7}	10^{-9}
Interval Width at $t = 1$	1.170	1.033	0.973	1.001	0.963	0.939	0.748	0.748	0.748
Steps Taken	17	25	38	15	22	34	18	28	43
CPU Time	2.5 ms	3.6 ms	5.6 ms	0.8 ms	1.1 ms	1.7 ms	3.4 ms	5.1 ms	7.7 ms

[a] 4th-order Taylor models are used in VSPODE, in addition to the 10th-order Taylor series expansion.

their interval counterparts (as asserted by the theory), but they are also significantly tighter. A further comparison with the bounds computed with VSPODE shows that the convex/concave relaxations still provide a tighter enclosure of the actual solution set on a large portion of the variable set P.

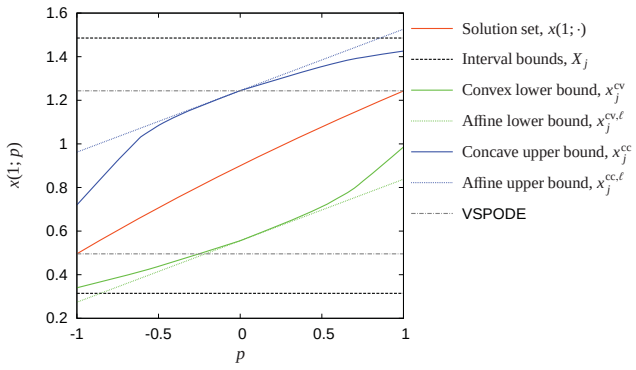

Figure 3.3: Interval bounds, convex/concave relaxations, and affine relaxations (for $p^{\text{ref}} = 0$) for the state variable x at $t = 1$ in the scalar ODE problem (3.17).

Also represented in Fig. 3.3 are a pair of affine relaxations for $x(t; \cdot)$ at $t = 1$. To generate them, it is necessary to first propagate subgradients for both the convex and concave relaxations at a given reference point. Then, the affine relaxations are readily obtained from (3.16). The reference point in Fig. 3.3 is chosen as the midpoint

$p^{\text{ref}} = 0$. It is clear that the choice of the reference point has a strong influence on the resulting affine relaxations.

At first sight, the convex and concave relaxations in Fig. 3.3 seem to be smooth. However, this smoothness is only apparent, and the nonsmooth behavior is best seen in Fig. 3.4, which displays subgradients of the convex and concave relaxations computed at various points in P. At points of nonsmoothness, such as $p = 0$, the subgradient is no longer a singleton but a proper interval. This nonsmoothness can be partly attributed to the McCormick relaxations themselves since they involve nonsmooth operations such as min and max, but also to the use of the transformation matrix $\mathbf{A}$ for mitigating the wrapping effect. Note also that the subgradients in Fig. 3.4 establish the convexity and concavity of the relaxations since they are, respectively, nondecreasing and nonincreasing on P.

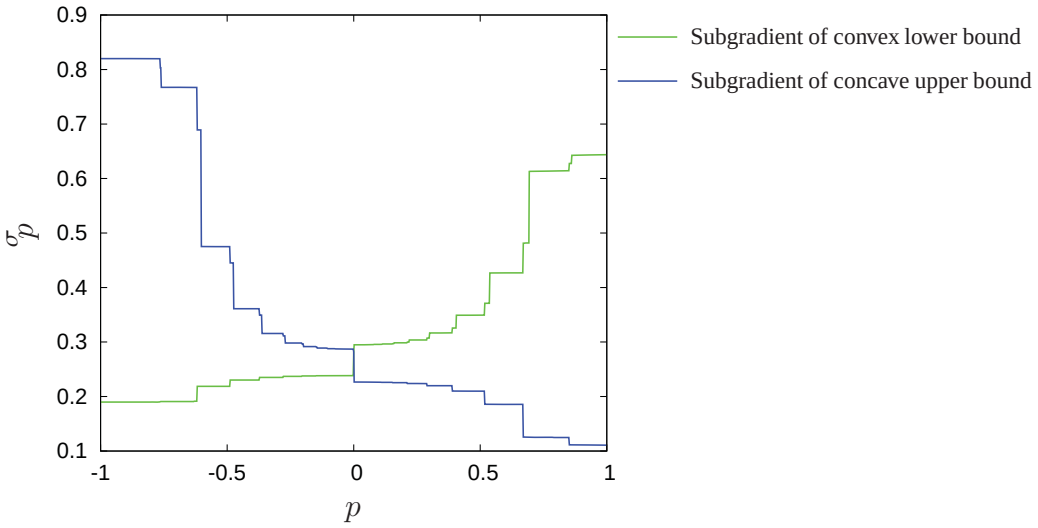

Figure 3.4: Subgradients of the convex and concave relaxations for the state variable x at $t = 1$ in the scalar ODE problem (3.17).

One can take advantage of the affine relaxations in order to further tighten the interval bounds at each integration step, as discussed in §3.1.2.1. This refinement strategy is tested here for various sets of reference points in $\text{int}(P) = (-1, 1)$. The results reported in Table 3.3 correspond to $N = 0, 1, 2, 5$ and 10 reference points,

with the positions of the points chosen as

$$p_k^{\text{ref}} = p^{\text{L}} + \frac{k}{N+1}(p^{\text{U}} - p^{\text{L}}) = 2\frac{k}{N+1} - 1, \quad k = 1, \ldots, N.$$

Both the relative reduction in widths of the interval bounds at $t = 1$ and the corresponding CPU time are reported Table 3.3.

Table 3.3: Refined interval bounds for x at $t = 1$ with $N = 0, 1, 2, 5$ and 10 reference points in the scalar ODE problem (3.17).

N	Reduction (%)	CPU time (ms)
0	–	8
1	–	14
2	4.2%	20
5	10.4%	40
10	10.5%	75

It can be seen that the use of several reference points effectively tightens the interval bounds, with a reduction in width up to about 10%. Yet, this is at the expense of significantly larger computational effort, since each additional reference point requires the computation of convex/concave bounds and subgradients at that point (about 7 ms for each reference point). The refined interval bounds and convex/concave relaxations for $N = 10$ points are shown in Fig. 3.5.

3.2.2 Lotka-Volterra System

Consider the following Lotka-Volterra problem:

$$\dot{x}_1(t) = p x_1(t)\,[1 - x_2(t)]\,; \quad x_1(0) = 1.2 \tag{3.18}$$
$$\dot{x}_2(t) = p x_2(t)\,[x_1(t) - 1]\,; \quad x_2(0) = 1.1 \tag{3.19}$$

with $t \geq 0$, and the variable $p \in P := [2.95, 3.05]$.

The bound and relaxation trajectories corresponding to the state variables x_1 and x_2 are displayed in Fig. 3.6. The left plots show comparisons between interval bounds

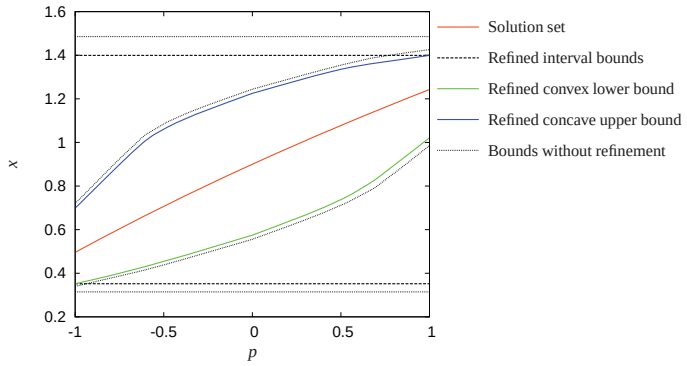

Figure 3.5: Refined interval bounds and convex/concave relaxations, with 10 reference points, for the state variable x at $t = 1$ in the scalar ODE problem (3.17); the dotted lines and relaxations correspond to the case without the tightening procedure.

obtained from differential inequalities [95] and bounds calculated by the two-phase interval ODE method in §2.4.1.2. A better performance is obtained with the latter on this problem, which can be attributed to its ability to mitigate the wrapping effect efficiently. Note in particular that the bounds obtained with differential inequalities blow up before $t = 2$. The convex and concave bounds calculated by the proposed discretize-then-relax approach for the variable value $p = 3$ are shown on the right plots. To reiterate, these bounds are guaranteed to be no looser than their interval counterpart by the properties of generalized McCormick relaxations.

Pointwise-in-time convex and concave relaxations of x_1 and x_2 on P at $t = 2$ are displayed in Fig. 3.7. These plots are generated by computing the convex and concave bounds $[x_1^{\mathrm{cv}}{}_j, x_1^{\mathrm{cc}}{}_j](p)$ and $[x_2^{\mathrm{cv}}{}_j, x_2^{\mathrm{cc}}{}_j](p)$ repeatedly at a large number of points $p \in [2.95, 3.05]$. Here again, the relaxations provide much tighter bounds than the underlying interval bounds. For comparison, the bounds obtained with the interval ODE solver VSPODE [49]—with Taylor models of order 4—are also reported in Fig. 3.7. While the VSPODE bounds clearly outperform simple interval bounds, convex/concave bounds remain competitive since they approximate the original solution set more closely on a large portion of the variable domain P. From these results, it is worthwhile to consider the combination of Taylor models with convex/concave relaxations in the next developments.

Finally, in order to illustrate the computation of convex/concave bounds in the

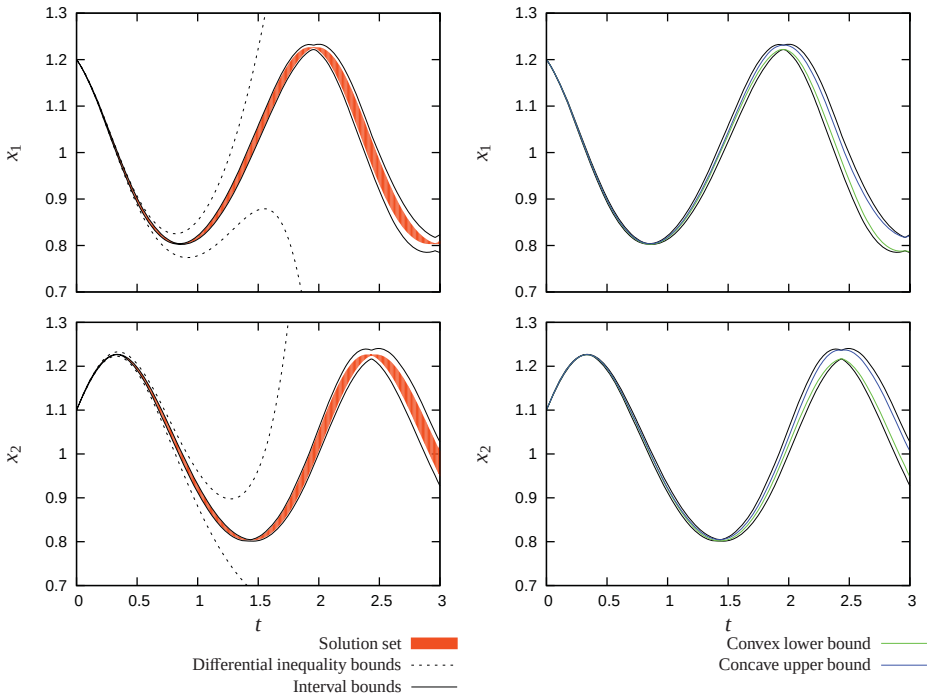

Figure 3.6: Interval bounds (left plot) and convex/concave bounds at $p = 3$ (right plot) for the state variables x_1 and x_2 in the Lotka-Volterra problem (3.18),(3.19).

presence of multiple variables, the following modified Lotka-Volterra problem is considered

$$\dot{x}_1(t) = p_1 x_1(t)\,[1 - x_2(t)]\,; \quad x_1(0) = 1.2, \tag{3.20}$$
$$\dot{x}_2(t) = p_2 x_2(t)\,[x_1(t) - 1]\,; \quad x_2(0) = 1.1, \tag{3.21}$$

with $t \geq 0$, and the variables $(p_1, p_2) \in \mathbf{P} := [2.98, 3.02] \times [0.98, 1.02]$.

The discretize-then-relax algorithm proceeds in the exact same way for multiple variables. Pointwise-in-time convex and concave relaxations of x_1 and x_2 on $\mathbf{P}$ at $t = 2$ are displayed in Fig. 3.8; the solution set of (3.20),(3.21) is not represented in this plot in the interest of clarity. The interval bounds for x_1 and x_2 at $t = 2$ are computed as $[0.8540, 0.8768]$ and $[0.8750, 0.8935]$, respectively, and therefore convex/concave bounds result in large improvements.

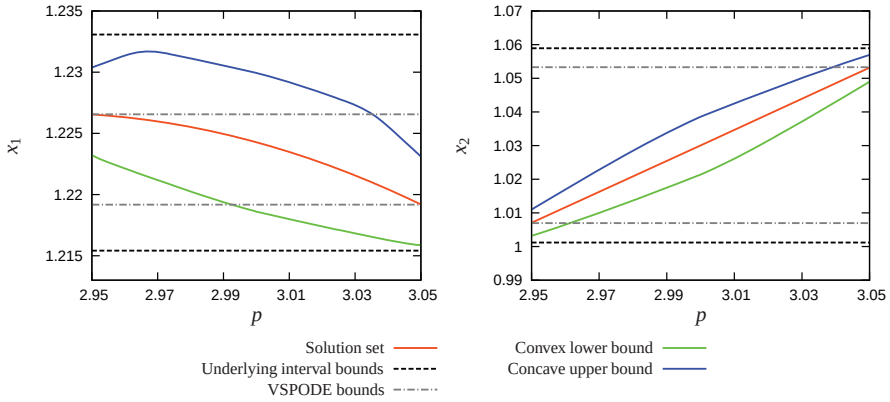

Figure 3.7: Interval and convex/concave bounds for the state variables x_1 (left plot) and x_2 (right plot) at $t = 2$ in the Lotka-Volterra problem (3.18),(3.19).

3.3 Conclusions

An algorithm that computes convex and concave bounds for the solutions of nonlinear parametric ODEs has been developed. This algorithm builds upon interval methods for ODEs and the McCormick relaxation technique. Unlike other existing ODE relaxation methods, it is rigorous in its accounting of truncation errors and has built-in capabilities to efficiently mitigate the wrapping effect of interval arithmetic. Also, a procedure that takes advantage of the subgradients of the state relaxations at preselected reference variable values to further refine the state bounds at each integration step has been described. In terms of computational effort, the time needed to compute convex/concave bounds is a fixed multiple factor of that of interval bounds, typically of the order of two to three. If the subgradient tightening procedure is used, this effort increases linearly with the number of reference points. Finally, numerical case studies demonstrated the effectiveness of the algorithm to generate tight convex/concave state relaxations that are considerably tighter than their underlying interval bounds. However, due to the dependency problem of interval analysis, the resulting convex/concave bounds can be still looser than the interval bounds obtained from Taylor model ODE methods [49]. The next chapter considers incorporation of Taylor models into the algorithm so as to improve the relaxation algorithm.

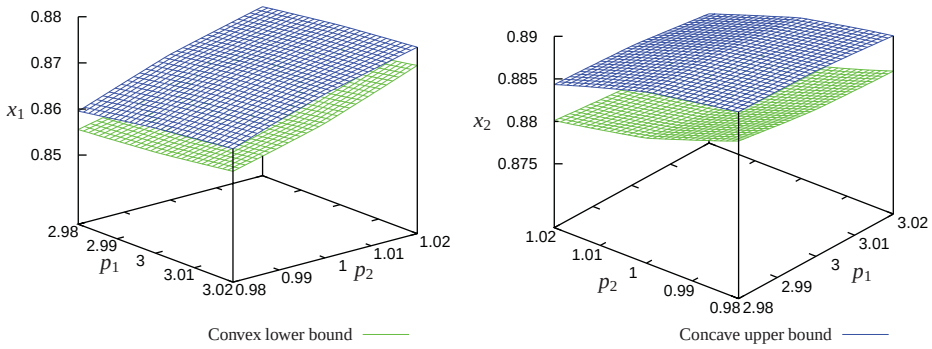

Figure 3.8: Convex/concave bounds for the state variables x_1 (left plot) and x_2 (right plot) at $t = 2$ in the modified Lotka-Volterra problem (3.20),(3.21).

Chapter4

Convex/Concave Relaxation of Dynamic Models using Taylor Model-based ODE Methods

As mentioned in Chapter 3, convex/concave relaxations typically yield tighter enclosures than simple interval bounds. However, the quality of the relaxations is directly affected by the quality of the interval bounds used in the relaxation algorithm. In the relaxation algorithm proposed in §3.1, the underlying interval bounds are obtained from interval analysis. Therefore, the same limitations of interval analysis also hold for the proposed algorithm, particularly the dependency problem. For this reason, it was seen that the convex/concave state relaxations could not always outperform the interval bounds obtained from Taylor models (see Fig. 3.7), which deal with the dependency problem systematically.

Taylor models have been used for GDO by Lin and Stadtherr [47, 48]. Their GDO algorithm offered superior convergence speeds compared to literature results [26, 96], which use previous relaxation techniques such as those based on differential inequalities. Provided the ability of Taylor models in computing tighter, faster converging enclosures, there seems to be much promise in combining Taylor models with convex/concave relaxations. This is the subject of this chapter.

The remainder of this chapter is organized as follows. In §4.1, different ways of combining McCormick relaxations with Taylor models are discussed. In particular, a new type of Taylor models, called McCormick-Taylor model, is introduced, and a procedure to extend McCormick-Taylor models to parametric ODEs is proposed. The relaxations of Taylor models are demonstrated through case studies in §4.2. Finally, the chapter is closed by concluding remarks in §4.3.

4.1 Taylor Model-based Relaxations

In this section, the combination of Taylor models with McCormick relaxations is considered in order to enable tighter convex/concave relaxations, and thus, a more efficient GDO algorithm.

A very simple form of the combination is by replacing interval analysis with Taylor models in McCormick relaxations. This way, Taylor models are propagated alongside McCormick relaxations in order to provide the underlying interval bounds required by the McCormick relaxation technique. Although straightforward, it has been observed that the convex/concave bounds obtained with this approach are often no better than the interval bounds from Taylor models, thereby providing no benefit over the Taylor model itself [81].

Another approach consists in direct relaxation of Taylor models using McCormick's technique. Like any factorable function, a Taylor model $\mathcal{T}_f = \mathcal{P}_f + R_f$ can be relaxed using McCormick's technique. This involves relaxation of the polynomial part $\mathcal{P}_f$ at a given $\underline{\mathbf{p}} \in \mathbf{P}$ to obtain convex/concave bounds $[\mathcal{P}_f^{\rm cv}(\underline{\mathbf{p}}), \mathcal{P}_f^{\rm cc}(\underline{\mathbf{p}})]$ (see page 83), and adding the result to the interval remainder bound. This way, convex/concave bounds at $\underline{\mathbf{p}}$ for a function f with a Taylor model $\mathcal{T}_f$ can be computed as

$$[f^{\rm cv}(\underline{\mathbf{p}}), f^{\rm cc}(\underline{\mathbf{p}})] = [\mathcal{P}_f^{\rm cv}(\underline{\mathbf{p}}) + r_f^{\rm L}, \mathcal{P}_f^{\rm cc}(\underline{\mathbf{p}}) + r_f^{\rm U}]. \tag{4.1}$$

Finally, a third form of the combination is by introducing a new type of Taylor models called McCormick-Taylor model [81, 83], as presented in the subsequent subsection.

4.1.1 Proposed McCormick-Taylor Models

A McCormick-Taylor model is a Taylor model where convex/concave bounds on the remainder term are computed in addition to the usual interval remainder bounds (see §2.3.2). Consider a function $f : \mathbf{P} \to \mathbb{R}$ defined on the set $\mathbf{P} \subset \mathbb{R}^{n_p}$, and let there be an n_p-variate polynomial $\mathcal{P}_f$ of order q and a pair of convex/concave functions on $\mathbf{P}$, r_f^{cv} and r_f^{cc}, such that:

$$f(\mathbf{p}) \in \mathcal{P}_f(\mathbf{p}) + [r_f^{\mathrm{cv}}, r_f^{\mathrm{cc}}](\mathbf{p}), \quad \text{for each } \mathbf{p} \in \mathbf{P}.$$

Based on this extension, a new type of Taylor models, called McCormick-Taylor model hereafter, can be defined for the function f at a point $\mathbf{p} \in \mathbf{P}$ as:

$$\mathcal{MT}_f(\mathbf{p}) := \mathcal{P}_f(\mathbf{p}) + \mathcal{M}_{R_f}(\mathbf{p}), \quad \text{for each } \mathbf{p} \in \mathbf{P},$$

where $\mathcal{M}_{R_f}(\mathbf{p}) = \left\{[r_f^{\mathrm{L}}, r_f^{\mathrm{U}}], [r_f^{\mathrm{cv}}, r_f^{\mathrm{cc}}](\mathbf{p})\right\}$ is a McCormick relaxation of the remainder term.

Like with any Taylor model $\mathcal{T}_f$, a McCormick-Taylor model $\mathcal{MT}_f$ encloses the function f between two hypersurfaces on $\mathbf{P}$. Although it should be clear that, in general, neither the lower-bounding function $\mathcal{P}_f + r_f^{\mathrm{cv}}$, nor the upper-bounding function $\mathcal{P}_f + r_f^{\mathrm{cc}}$ are, respectively, convex and concave on $\mathbf{P}$, since the polynomial function $\mathcal{P}_f$ may be neither convex nor concave on $\mathbf{P}$.

In order to distinguish those McCormick-Taylor models having a centered interval remainder bound, the notation $\mathcal{MT}_f^{\mathsf{C}}$ shall be used. In particular, a McCormick-Taylor model $\mathcal{MT}_f$ with a non-centered remainder bound can always be rewritten *equivalently* as the centered McCormick-Taylor model $\mathcal{MT}_f^{\mathsf{C}} = \mathcal{P}_f^{\mathsf{C}} + \mathcal{M}_{R_f}^{\mathsf{C}}$, with $\mathcal{P}_f^{\mathsf{C}} = \mathcal{P}_f + \mathrm{m}([r_f^{\mathrm{L}}, r_f^{\mathrm{U}}])$ and $\mathcal{M}_{R_f}^{\mathsf{C}} = \mathcal{M}_{R_f} - \mathrm{m}([r_f^{\mathrm{L}}, r_f^{\mathrm{U}}])$.

McCormick-Taylor Model Arithmetic. Similar to Taylor model arithmetic, the polynomial part in a McCormick-Taylor model is propagated by symbolic calculations wherever possible. On the other hand, the remainder term and all polynomial terms of order higher than q are now processed according to the rules of the generalized McCormick relaxation technique (see §2.3.3.1).
To illustrate it, consider two real-valued functions f_1 and f_2 defined on $\mathbf{P}$, and

suppose that $\mathcal{MT}_{f_1} = \mathcal{P}_{f_1} + \mathcal{M}_{f_1}$ and $\mathcal{MT}_{f_2} = \mathcal{P}_{f_2} + \mathcal{M}_{f_2}$ are qth-order McCormick-Taylor models of these functions on $\mathbf{P}$. Similar to Taylor model arithmetic, a qth-order McCormick-Taylor model of $f_1 + f_2$ on $\mathbf{P}$ is trivially obtained as $\mathcal{MT}_{f_1+f_2} = \mathcal{MT}_{f_1} + \mathcal{MT}_{f_2}$. Regarding the product $f_1 \times f_2$, one has:

$$\begin{aligned} f_1(\mathbf{p}) \times f_2(\mathbf{p}) \in\ & \mathcal{P}_{f_1}(\mathbf{p}) \times \mathcal{P}_{f_2}(\mathbf{p}) + \mathcal{M}_{R_{f_1}}(\mathbf{p}) \times \mathcal{M}_{\mathcal{P}_{f_2}}(\mathbf{p}) \\ & + \mathcal{M}_{\mathcal{P}_{f_1}}(\mathbf{p}) \times \mathcal{M}_{R_{f_2}}(\mathbf{p}) + \mathcal{M}_{R_{f_1}}(\mathbf{p}) \times \mathcal{M}_{R_{f_2}}(\mathbf{p}), \end{aligned}$$

for all $\mathbf{p} \in \mathbf{P}$, where $\mathcal{M}_{\mathcal{P}}$ stands for the McCormick relaxations of the qth-order polynomial $\mathcal{P}$ (see below). Because $\mathcal{P}_{f_1} \times \mathcal{P}_{f_2}$ is a polynomial of order $2q$, whereas a qth-order McCormick-Taylor model is sought, consider splitting this term as $\mathcal{P}_{f_1} \times \mathcal{P}_{f_2} = \mathcal{P}_{f_1 \times f_2} + \mathcal{H}_{f_1 \times f_2}$, where $\mathcal{P}_{f_1 \times f_2}$ contains all the terms of order q or less, and $\mathcal{H}_{f_1 \times f_2}$ contains the higher-order terms. A qth-order McCormick-Taylor model of $f_1 \times f_2$ on $\mathbf{P}$ is now given by $\mathcal{MT}_{f_1 \times f_2} = \mathcal{P}_{f_1 \times f_2} + \mathcal{M}_{R_{f_1 \times f_2}}$, with:

$$\begin{aligned} \mathcal{M}_{R_{f_1 \times f_2}}(\mathbf{p}) = & \mathcal{M}_{\mathcal{H}_{f_1 \times f_2}}(\mathbf{p}) + \mathcal{M}_{R_{f_1}}(\mathbf{p}) \times \mathcal{M}_{\mathcal{P}_{f_2}}(\mathbf{p}) \\ & + \mathcal{M}_{\mathcal{P}_{f_1}}(\mathbf{p}) \times \mathcal{M}_{R_{f_2}}(\mathbf{p}) + \mathcal{M}_{R_{f_1}}(\mathbf{p}) \times \mathcal{M}_{R_{f_2}}(\mathbf{p}). \end{aligned}$$

McCormick-Taylor models for the composition with a univariate intrinsic function (such as exponential, logarithm, inverse, square root, sine and cosine) can also be computed, provided that this intrinsic function is sufficiently many times continuously differentiable and convex/concave relaxations are known for it and its derivatives on the interval domain of its argument. In sum, these rules provide a way to compute qth-order McCormick-Taylor models for $(q+1)$ times continuously differentiable factorable functions.

Relaxation of McCormick-Taylor Models. A McCormick relaxation of a McCormick-Taylor model can be obtained by first computing a McCormick relaxation for its polynomial part, and then adding it to the McCormick relaxation of the remainder term,

$$\mathcal{M}_f(\mathbf{p}) = \mathcal{M}_{\mathcal{P}_f}(\mathbf{p}) + \mathcal{M}_{R_f}(\mathbf{p}).$$

Because constructing the convex/concave envelopes—i.e., the tightest possible convex/concave relaxations—for a multivariate polynomial turns out to be NP-hard,

some kind of over-relaxation approach needs to be considered. Unfortunately, direct use of the McCormick technique to relax $\mathcal{P}_f$ usually produces weak relaxations. Following the same ideas as [49] for range bounding of Taylor models (see §2.3.2), priority can be given to exact bounding of the first- and diagonal second-order terms, whereas the other terms are directly relaxed via the generalized McCormick technique. Let the polynomial $\mathcal{P}_f$ be written as $\mathcal{P}_f(\mathbf{p}) = \mathcal{Q}(\mathbf{p}) + \mathcal{R}(\mathbf{p})$, with $\mathcal{Q}(\mathbf{p})$ and $\mathcal{R}(\mathbf{p})$ given by (2.4). A McCormick relaxation of $\mathcal{P}_f$ on $\mathbf{P}$ is then obtained as:

$$\mathcal{M}_{\mathcal{P}_f}(\mathbf{p}) = \mathcal{M}_{\mathcal{Q}}(\mathbf{p}) + \mathcal{M}_{\mathcal{R}}(\mathbf{p}), \quad \text{for each } \mathbf{p} \in \mathbf{P},$$

where $\mathcal{M}_{\mathcal{Q}}$ and $\mathcal{M}_{\mathcal{R}}$ are McCormick relaxations of $\mathcal{Q}$ and $\mathcal{R}$, respectively. In order to prevent division by zero, the rearrangement in (2.4) is considered only when $|a_i| \geq \epsilon$, where ϵ is a small positive number; otherwise, a McCormick relaxation $\mathcal{M}_{\mathcal{P}_f}$ of $\mathcal{P}_f$ is computed without the rearrangement in (2.4). It should be noted that, since the convex/concave bounds in generalized McCormick relaxations are no looser than the supporting interval bounds [90], neither will the convex/concave bounds derived from the relaxation of a McCormick-Taylor model be any looser than their supporting interval bounds. Moreover, the interval bounds obtained from the relaxation of a McCormick-Taylor model are identical to the usual Taylor model-derived bounds.

Subgradients too can be computed for the McCormick relaxation of a McCormick-Taylor model. This requires that subgradients be first propagated during the computation of the remainder relaxation $\mathcal{M}_{R_f}(\mathbf{p})$, and then added to the subgradients propagated during the relaxation of the polynomial part $\mathcal{M}_{\mathcal{P}_f}(\mathbf{p})$.

Example. *Consider the function defined by* $f(p) = p\exp(-p^2)$*, for* $p \in P := [-0.5, 1]$*. On the left plot in Fig. 4.1, the* 4*th-order McCormick-Taylor model* $\mathcal{MT}_f$ *of* f *on* P *is represented in blue/green lines, along with the function* f *in solid red line. The solid green and blue lines depict, respectively, the functions* $\mathcal{P}_f + r_f^{\mathrm{cv}}$ *and* $\mathcal{P}_f + r_f^{\mathrm{cc}}$*, while the dashed green and blue lines depict, respectively,* $\mathcal{P}_f + r_f^{\mathrm{L}}$ *and* $\mathcal{P}_f + r_f^{\mathrm{U}}$*. Observe that each pair of these functions enclose* $f(p)$ *at each* $p \in P$*, and that those using the convex/concave remainder bounds yield tighter enclosures than the ones using the interval remainder bounds. Notice also that neither* $\mathcal{P}_f + r_f^{\mathrm{cv}}$ *nor* $\mathcal{P}_f + r_f^{\mathrm{cc}}$ *are, respectively, convex and concave on* P*, as a result of the nonconvexity/nonconcavity*

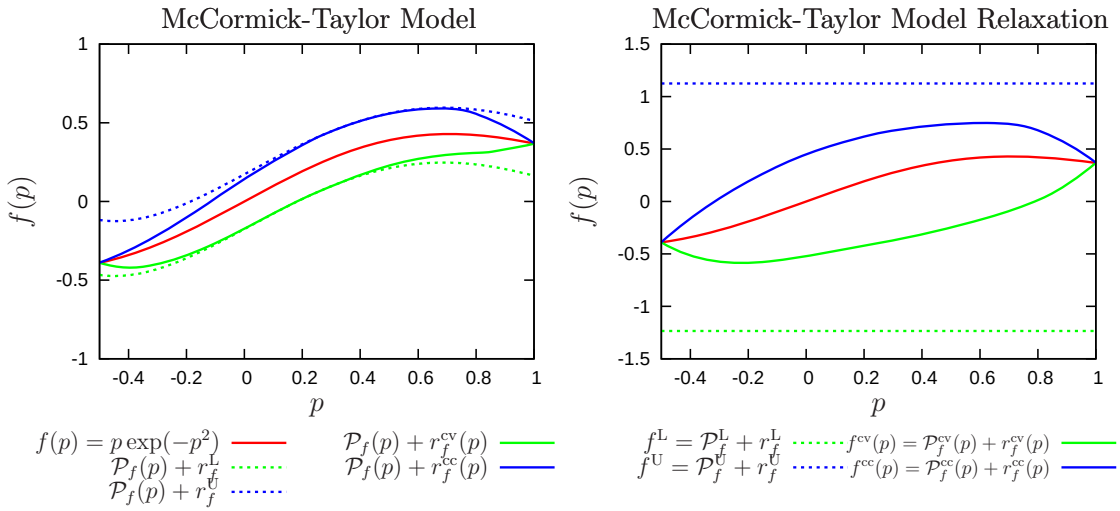

Figure 4.1: McCormick-Taylor model (left plot) and its McCormick relaxations (right plot) for $f(p) = p\exp(-p^2)$.

of the polynomial $\mathcal{P}_f$.

The McCormick relaxation $\mathcal{M}_f$ *of* $\mathcal{MT}_f$ *is shown on the right plot in Fig. 4.1, with the solid green and blue lines now representing the convex and concave bounds* $[f^{\rm cv}, f^{\rm cc}]$*, and the dashed green and blue lines representing the supporting interval bounds* $[f^{\rm L}, f^{\rm U}]$*. By construction, the former are guaranteed to enclose* $\mathcal{MT}_f(p)$ *for each* $p \in P$*—and therefore contain* $f(p)$ *itself—in addition to being convex or concave, whereas the latter are guaranteed to enclose the image of* f *on* P*,* $\mathcal{I}_f(P)$*.*

4.1.2 McCormick-Taylor Models for Ordinary Differential Equations

The attendant advantages of McCormick-Taylor models for computing convex/concave relaxations for the solutions of parametric ODEs are clear. Because McCormick-Taylor models inherit the high-order convergence properties of Taylor models, tighter

enclosures can be expected for the state variables as the variable set shrinks, a very desirable property for use in branch-and-bound algorithms. Moreover, the resulting state relaxations will be at least as good as the state interval enclosures derived from the Taylor model approach in §2.4.2. On the other hand, the use of (McCormick-)Taylor model-based methods comes with a few drawbacks. First and foremost, Taylor models are known to be much more computationally demanding than interval-based methods, especially when high-order Taylor models are considered or with a large number of variables, and McCormick-Taylor models add even more to this computational burden since they also propagate convex/concave bounds—typically a two-fold increase. Despite these important limitations, however, Lin and Stadtherr [48] have shown that the use of Taylor model-derived bounds in a global search procedure typically outperforms other bounding methods. There are therefore good reasons to believe that McCormick-Taylor models could provide even more improvement in this context.

The proposed discretize-then-relax algorithm for parametric ODEs builds upon the verified ODE approach described in §2.4.2 and extends it to encompass McCormick-Taylor models. For a specified variable value $\mathbf{p} \in \mathbf{P}$, the objective is to compute McCormick relaxations $\mathcal{M}_{\mathbf{x}_j}(\mathbf{p})$, at each integration step t_j, $j = 0, \ldots, N$. Unlike the discretize-then-relax scheme proposed in [80, 82], however, it is the McCormick-Taylor models $\mathcal{MT}_{\mathbf{x}_j}(\mathbf{p})$ of the functions $\mathbf{x}(t_j; \cdot)$ that are now propagated at each t_j, rather than the desired McCormick relaxations $\mathcal{M}_{\mathbf{x}_j}(\mathbf{p})$ themselves. The latter are obtained in a subsequent step, from the McCormick relaxation of $\mathcal{MT}_{\mathbf{x}_j}(\mathbf{p})$.

The prerequisites for applying this approach are the same as those for the Taylor model ODE method. The algorithm begins by constructing a McCormick-Taylor model $\mathcal{MT}_{\mathbf{x}_0}$ of $\mathbf{x}(t_0; \mathbf{p})$,

$$\mathcal{MT}_{\mathbf{x}_0}(\mathbf{p}) = \mathcal{MT}_{\mathbf{h}}(\mathbf{p}). \tag{4.2}$$

Next, the McCormick-Taylor model $\mathcal{MT}_{\mathbf{x}_j}(\mathbf{p})$ is propagated through each integration step t_j, $j = 0, \ldots, N$, based on a two-phase procedure similar to those described earlier in §2.4.1 and §2.4.2.

Phase I Given a McCormick-Taylor model $\mathcal{MT}_{\mathbf{x}_j}(\mathbf{p})$ of $\mathbf{x}(t_j; \mathbf{p})$ on $\mathbf{P}$, Phase I computes a stepsize h_j and a McCormick-Taylor model $\mathcal{MT}_{\widetilde{\mathbf{x}}_j}(\mathbf{p})$ that encloses $\{\mathbf{x}(t; \mathbf{p}) : t_j \leq t \leq t_j + h_j\}$. Such an a priori McCormick-Taylor model enclosure can

be obtained as follows:

$$\mathcal{MT}_{\widetilde{\mathbf{x}}_j}(\mathbf{p}) = \sum_{i=0}^{k-1}[0,h_j]^i \mathcal{MT}_{\mathbf{f}^{[i]}}(\mathcal{MT}_{\mathbf{x}_j}(\mathbf{p}),\mathbf{p}) + [0,h_j]^k \mathbf{f}^{[k]}(\widetilde{\mathbf{X}}_j^0,\mathbf{P}). \tag{4.3}$$

Provided that $\widetilde{\mathbf{X}}_j^0$ and $h_j = t_{j+1} - t_j$ satisfy the condition (2.34) in Corollary 2.1 for each $\mathbf{x}_j \in \mathbf{X}_j$, it is readily seen from the Taylor expansion (2.27) and the properties of McCormick-Taylor models that (4.3) yields a valid enclosure of $\mathbf{x}$ on $[t_j, t_{j+1}] \times \mathbf{P}$. This Phase I algorithm relies on the ability to compute McCormick-Taylor models for the Taylor coefficients, which are of the form $\mathbf{f}^{[i]}(\boldsymbol{\xi}(\mathbf{p}),\mathbf{p})$, $i = 1,\ldots,k$. In particular, the recursive technique described in §4.1.1 applies because the outer functions $\mathbf{f}^{[i]}$ as obtained from (2.26) are factorable, and a McCormick-Taylor model $\mathcal{MT}_{\boldsymbol{\xi}}(\mathbf{p})$ is known for the inner function $\boldsymbol{\xi}$ at $\mathbf{p}$.

Phase II Given a McCormick-Taylor model $\mathcal{MT}_{\mathbf{x}_j}(\mathbf{p})$ of $\mathbf{x}(t_j;\mathbf{p})$ on $\mathbf{P}$ and a priori McCormick-Taylor model enclosure $\mathcal{MT}_{\widetilde{\mathbf{x}}_j}(\mathbf{p})$ of $\mathbf{x}(t;\mathbf{p})$ on $\mathbf{P}$ for $t \in [t_j, t_{j+1}]$, Phase II computes a McCormick-Taylor model $\mathcal{MT}_{\mathbf{x}_{j+1}}(\mathbf{p})$ of $\mathbf{x}(t_{j+1};\mathbf{p})$ on $\mathbf{P}$. Using the high-order Taylor series expansion with the mean-value theorem given in (2.42), and choosing $\hat{\mathbf{p}} = \mathbf{p}$ and $\hat{\mathbf{x}}_j(\mathbf{p}) = \mathcal{P}^{\mathsf{C}}_{\mathbf{x}_j}(\mathbf{p})$ as the (variable) reference for the mean-value theorem (see §2.4.2 for a discussion), one has:

$$\begin{aligned}\mathcal{MT}_{\mathbf{x}_{j+1}}(\mathbf{p}) = &\underbrace{\sum_{i=0}^{k-1} h_j^i \mathcal{MT}_{\mathbf{f}^{[i]}}(\mathcal{P}^{\mathsf{C}}_{\mathbf{x}_j}(\mathbf{p}),\mathbf{p})}_{=:\,\mathcal{MT}_{\mathbf{V}_{j+1}}(\mathbf{p})} + \underbrace{h_j^k \mathcal{MT}_{\mathbf{f}^{[k]}}(\mathcal{MT}_{\widetilde{\mathbf{x}}_j}(\mathbf{p}),\mathbf{p})}_{=:\,\mathcal{MT}_{\mathbf{R}_{j+1}}(\mathbf{p})} \\ &+ \underbrace{\left(\sum_{i=0}^{k-1} h_j^i \mathcal{MT}_{\frac{\partial \mathbf{f}^{[i]}}{\partial \mathbf{x}}}(\mathcal{MT}_{\mathbf{x}_j}(\mathbf{p}),\mathbf{p})\right)}_{=:\,\mathcal{MT}_{\mathbf{J}_{j+1}}(\mathbf{p})} \times \mathcal{M}^{\mathsf{C}}_{R_{\mathbf{x}_j}}(\mathbf{p}).\end{aligned} \tag{4.4}$$

After rearranging the matrix-vector product operations in order to mitigate the wrapping effect, one finally gets:

$$\begin{aligned}\mathcal{MT}_{\mathbf{x}_{j+1}}(\mathbf{p}) &= \mathcal{MT}_{\mathbf{V}_{j+1}}(\mathbf{p}) + \mathcal{MT}_{\mathbf{R}_{j+1}}(\mathbf{p}) \\ &\quad + \left[\mathcal{MT}_{\mathbf{J}_{j+1}}(\mathbf{p})\,\mathbf{A}_j\right]\mathcal{MT}_{\mathbf{\Delta}_j}(\mathbf{p}) \qquad (4.5)\\ \mathcal{MT}_{\mathbf{\Delta}_{j+1}}(\mathbf{p}) &= \left[\mathbf{A}_{j+1}^{-1}\left(\mathcal{MT}_{\mathbf{J}_{j+1}}(\mathbf{p})\,\mathbf{A}_j\right)\right]\mathcal{MT}_{\mathbf{\Delta}_j}(\mathbf{p}) \\ &\quad + \mathbf{A}_{j+1}^{-1}\left[\mathcal{MT}_{\mathbf{R}_{j+1}}(\mathbf{p}) + \mathcal{MT}_{\mathbf{V}_{j+1}}(\mathbf{p}) - \mathcal{P}^{\mathsf{C}}_{\mathbf{x}_{j+1}}(\mathbf{p})\right], \qquad (4.6)\end{aligned}$$

The transformation matrix $\mathbf{A}_{j+1}$ can be obtained by first bounding the range of $\mathcal{MT}_{\mathbf{J}_{j+1}}\mathbf{A}_j$ on $\mathbf{P}$, say $\mathbf{J}_{j+1}\mathbf{A}_j$, and then taking the orthogonal matrix in the QR decomposition of $\mathrm{m}(\mathbf{J}_{j+1}\mathbf{A}_j)$. This algorithm is initialized with (4.2), $\mathcal{MT}_{\Delta_0}(\mathbf{p}) = \mathcal{M}^{\mathsf{C}}_{R_{\mathbf{x}_0}}(\mathbf{p})$ and $\mathbf{A}_0 = \mathbf{I}$. The terms $\mathcal{MT}_{\frac{\partial \mathbf{f}^{[i]}}{\partial \mathbf{x}}}$ in (4.4) are the McCormick-Taylor models of the Jacobian $\frac{\partial \mathbf{f}^{[i]}}{\partial \mathbf{x}}$ of the Taylor coefficients, for each $i = 0, \ldots, k-1$. Similar to McCormick-Taylor models for the Taylor coefficients $\mathbf{f}^{[i]}$, these terms can be computed on application of the recursive technique described in §4.1.1.

4.1.2.1 Simplifications for Improved Efficiency

The discretize-then-relax algorithm detailed in §4.1.2 relies heavily on McCormick-Taylor models for bounding the Taylor coefficients and their Jacobians. Similar to Taylor models, an important drawback of McCormick-Taylor model is their high computational burden, in particular for high-order models and/or large number of variables. To improve efficiency, this subsection formulates an algorithm variant that limits the use of McCormick-Taylor models as much as possible, while preserving tightness of the ODE bounds.

From inspection of (4.4), it is seen that McCormick-Taylor models are computed for all three terms in the right-hand side at each step. These terms can be simplified without much effect on the resulting state bounds.

The vector $\mathcal{MT}_{\mathbf{R}_{j+1}}(\mathbf{p})$ typically remains small during the integration, especially when the step size h_j is itself small or automatically adjusted [66]. Consequently, devoting much effort to first compute an a priori McCormick-Taylor model enclosure $\mathcal{MT}_{\widetilde{\mathbf{x}}_j}(\mathbf{p})$, and then compute a McCormick-Taylor model as $\mathcal{MT}_{\mathbf{f}^{[k]}}(\mathcal{MT}_{\widetilde{\mathbf{x}}_j}(\mathbf{p}), \mathbf{p})$

appears somewhat unnecessary. Following the Phase I algorithm in §3.1.1, a McCormick relaxation $\mathcal{M}_{\widetilde{\mathbf{x}}_j}(\mathbf{p})$ of $\mathbf{x}(t;\mathbf{p})$ on $\mathbf{P}$ for $t \in [t_j, t_{j+1}]$ is given by:

$$\mathcal{M}_{\widetilde{\mathbf{x}}_j}(\mathbf{p}) = \sum_{i=0}^{k-1} [0,h_j]^i \mathcal{M}_{\mathbf{f}^{[i]}}(\mathcal{M}_{\mathbf{x}_j}(\mathbf{p}),\mathbf{p}) + [0,h_j]^k \mathbf{f}^{[k]}(\widetilde{\mathbf{X}}_j^0,\mathbf{P}), \tag{4.7}$$

where $\mathcal{M}_{\mathbf{x}_j}(\mathbf{p})$ results from the McCormick relaxation of $\mathcal{MT}_{\mathbf{x}_j}(\mathbf{p})$. In turn, a McCormick relaxation of the remainder term $\mathbf{f}^{[k]}(\mathbf{x}(t_j+\tau),\mathbf{p})$ in (3.10) is obtained as:

$$\mathcal{M}_{\mathbf{R}_{j+1}}(\mathbf{p}) = h_j^k \mathcal{M}_{\mathbf{f}^{[k]}}(\mathcal{M}_{\widetilde{\mathbf{x}}_j}(\mathbf{p}),\mathbf{p}),$$

which can be used in lieu of $\mathcal{MT}_{\mathbf{R}_j}(\mathbf{p})$ in (4.3).

By far the most demanding task in Phase II is computing the McCormick-Taylor models $\mathcal{MT}_{\frac{\partial \mathbf{f}^{[i]}}{\partial \mathbf{x}}}(\mathcal{MT}_{\mathbf{x}_j}(\mathbf{p}),\mathbf{p})$, for each $i = 1,\ldots,k-1$. It is interesting to note that the result of the matrix-vector product $\mathcal{MT}_{\mathbf{J}_{j+1}}(\mathbf{p}) \times \mathcal{M}^{\mathsf{C}}_{R_{\mathbf{x}_j}}(\mathbf{p})$ are McCormick relaxations—not McCormick-Taylor models. Therefore, another way of obtaining a McCormick relaxation of the Jacobian term $\frac{\partial \mathbf{f}^{[i]}}{\partial \mathbf{x}}$ involves computing the McCormick relaxations of the Jacobian of the Taylor coefficients, which are much less demanding than their McCormick-Taylor model counterparts:

$$\mathcal{M}_{\mathbf{J}_{j+1}}(\mathbf{p}) = \sum_{i=0}^{k-1} h_j^i \mathcal{M}_{\frac{\partial \mathbf{f}^{[i]}}{\partial \mathbf{x}}}(\mathcal{M}_{\mathbf{x}_j}(\mathbf{p}),\mathbf{p}),$$

and then multiplying the resulting matrix $\mathcal{M}_{\mathbf{J}_{j+1}}(\mathbf{p})$ by $\mathcal{M}^{\mathsf{C}}_{R_{\mathbf{x}_j}}(\mathbf{p})$.

Finally, the reference term $\mathcal{MT}_{\mathbf{f}^{[i]}}(\mathcal{P}^{\mathsf{C}}_{\mathbf{x}_j}(\mathbf{p}),\mathbf{p})$ can be simplified by computing it using usual Taylor models as in (2.44).

Incorporating the foregoing simplifications, (4.4) becomes:

$$\mathcal{MT}_{\mathbf{x}_{j+1}}(\mathbf{p}) = \mathcal{T}_{\mathbf{V}_{j+1}}(\mathbf{p}) + \mathcal{M}_{\mathbf{R}_{j+1}}(\mathbf{p}) + \mathcal{M}_{\mathbf{J}_{j+1}}(\mathbf{p}) \times \mathcal{M}^{\mathsf{C}}_{R_{\mathbf{x}_j}}(\mathbf{p}). \tag{4.8}$$

Like previously, it is important to rearrange these matrix-vector product operations

to mitigate the wrapping effect. Noting from (4.8) that $\mathcal{P}_{\mathbf{x}_j} = \mathcal{P}_{\mathbf{V}_j}$ for each $j > 0$, and therefore $\mathcal{T}_{\mathbf{V}_j} - \mathcal{P}^{\mathsf{C}}_{\mathbf{x}_j} = R^{\mathsf{C}}_{\mathbf{V}_j}$, it is not hard to show that one such possible rearrangement is:

$$\mathcal{MT}_{\mathbf{x}_{j+1}}(\mathbf{p}) = \mathcal{T}_{\mathbf{V}_{j+1}}(\mathbf{p}) + \mathcal{M}_{\mathbf{R}_{j+1}}(\mathbf{p}) + \left[\mathcal{M}_{\mathbf{J}_{j+1}}(\mathbf{p})\,\mathbf{A}_j\right]\mathcal{M}_{\mathbf{\Delta}_j}(\mathbf{p}) \tag{4.9}$$

$$\begin{aligned}\mathcal{M}_{\mathbf{\Delta}_{j+1}}(\mathbf{p}) = {} & \left[\mathbf{A}_{j+1}^{-1}\left(\mathcal{M}_{\mathbf{J}_{j+1}}(\mathbf{p})\,\mathbf{A}_j\right)\right]\mathcal{M}_{\mathbf{\Delta}_j}(\mathbf{p}) \\ & + \mathbf{A}_{j+1}^{-1}\left[\mathcal{M}_{\mathbf{R}_{j+1}}(\mathbf{p}) + R^{\mathsf{C}}_{\mathbf{V}_{j+1}}(\mathbf{p})\right],\end{aligned} \tag{4.10}$$

with initialization from (4.2), $\mathbf{A}_0 = \mathbf{I}$, and $\mathcal{M}_{\Delta_0}(\mathbf{p}) = \mathcal{M}^{\mathsf{C}}_{R_{\mathbf{x}_0}}(\mathbf{p})$. The need for computing McCormick-Taylor models is thus reduced in this algorithm variant.

4.2 Numerical Case Studies

In this section, the Taylor model based relaxation techniques presented in §4.1 are examined via numerical experiments. In particular, the bounding quality of the Taylor model and McCormick-Taylor model ODE methods are compared, with and without the simplifications described in §4.1.2.1. Moreover, the computational burden of these algorithms is studied. Finally, convergence properties of Taylor model and McCormick-Taylor model enclosures are investigated.

Unless otherwise noted, the simplified algorithm presented in §4.1.2.1 is considered throughout the case studies. Stepsize control is used in all the case studies, based on the LEPUS approach discussed in [66], where the initial stepsize is set to $h_0 = 0.01$ and the absolute/relative tolerances to 10^{-6}. Moreover, the order of the Taylor expansion in the HOE method is taken as $k = 10$, and Taylor models of order $q = 4$ are considered, unless otherwise noted. The computations are performed on a PC with Intel Core 2 Duo 2.4GHz CPU, 4GB RAM, and running GCC version 4.3.2 (Linux).

In the next subsection, the enclosure quality and computational aspects of the methods presented in this chapter are studied. In §4.2.2, the convergence properties of these methods are investigated.

4.2.1 Enclosure Quality and Computational Aspects

4.2.1.1 Scalar ODE Problem

Consider the following scalar IVP in ODEs,

$$\dot{x}(t;p) = -0.1(x(t;p)-p)^2; \quad x(0;p) = p^2 - 0.5, \tag{4.11}$$

with $t \in [0,2]$, and where the single variable $p \in P := [-2,0]$ appears both in the ODE and in the initial condition.

Since the McCormick-Taylor model algorithm presented in §4.1.2 builds upon Taylor model methods for verified ODE integration, the interval bounds derived from this algorithm are identical to those from the Taylor model ODE method. Figure 4.2 shows these bounds as obtained from the Phase I and Phase II of the aforementioned algorithms at each integration step along with the actual solution set of Problem (4.11). It is seen that these bounds are able to follow the contracting solution set—as enabled by the mean-value theorem in Phase II—, and that the overestimation inherent to verified ODE integration remains quite small in this simple problem. Observe the effect of the stepsize control strategy, which reaches the final time $t = 2$ in 7 steps only, despite the specified initial step size of $h_0 = 0.01$. Note that the tightness of these bounds has a direct influence on the tightness of the resulting convex and concave bounds.

The Taylor model and McCormick-Taylor model of $x(2;\cdot)$ on P are depicted in Fig. 4.3. It is seen that both pairs of lower/upper bounding functions in Fig. 4.3 validly enclose the ODE solutions for each $p \in P$. Note that the McCormick-Taylor model $\mathcal{MT}_{x_2(t_\mathrm{f})}$ obtained from each round of integration is valid for a specific $\underline{p} \in P$ only because its convex/concave remainder bound is obtained at one $\underline{p}$ each time. Therefore, to generate the plot of a McCormick-Taylor model over P, the integration must be repeated for a large number of points in P. On the other hand, only one integration is needed to generate the plot of the Taylor model $\mathcal{T}_{x_2(t_\mathrm{f})}$ because its interval remainder bound is valid for all $p \in P$. Therefore, its plot can be generated by simply evaluating the computed $\mathcal{T}_{x_2(t_\mathrm{f})}$ at a large number of points in P.

The left plot in Fig. 4.3 shows the results of the simplified Taylor model algorithm and the simplified McCormick-Taylor model algorithm that minimize the use of Taylor/McCormick-Taylor models. On the other hand, the right plot presents

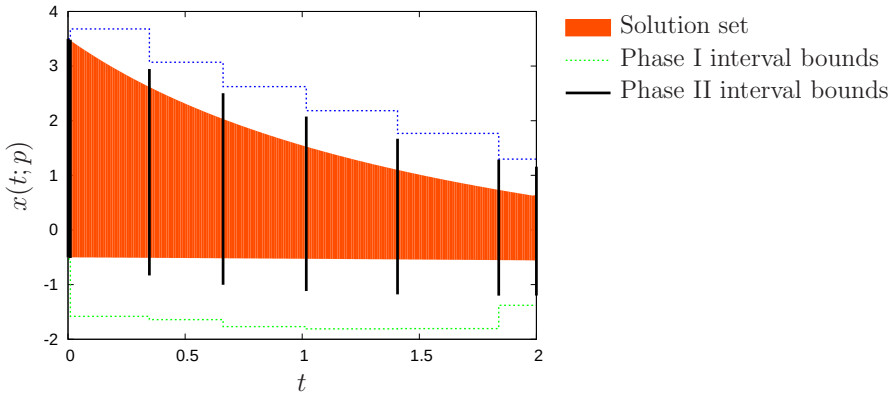

Figure 4.2: Interval bounds after Phase I and Phase II in Problem (4.11).

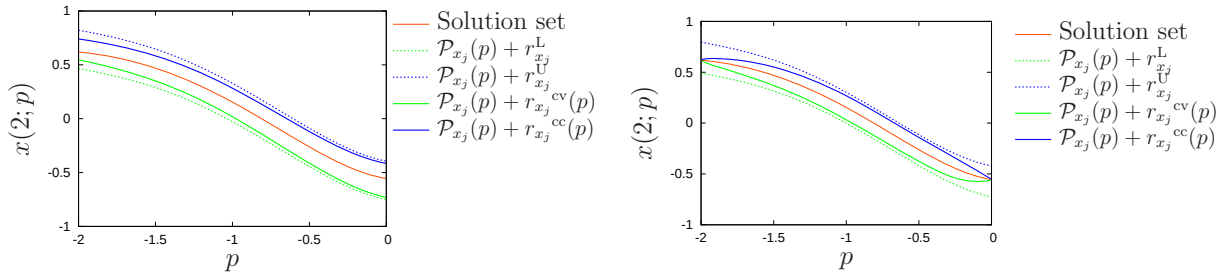

Figure 4.3: Taylor and McCormick-Taylor models at $t = 2$ in Problem (4.11); left plot: simplified algorithms, right plot: full algorithms.

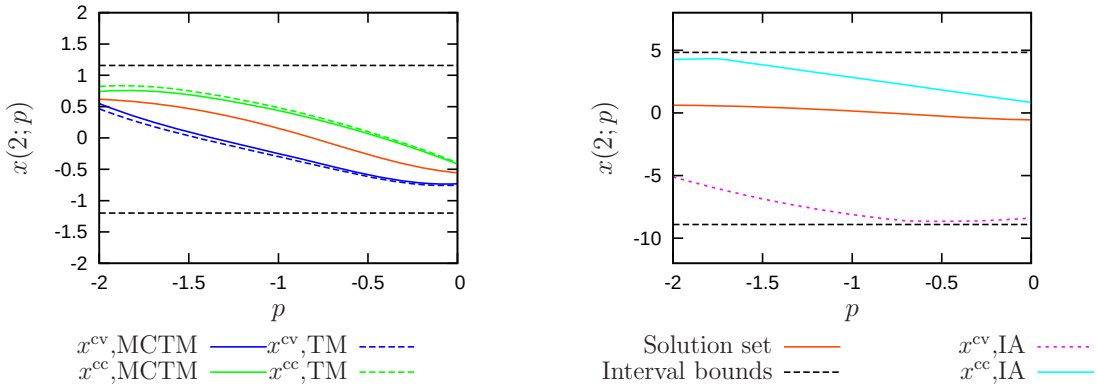

Figure 4.4: Interval and convex/concave bounds at $t = 2$ in Problem (4.11). Left plot: with Taylor and McCormick-Taylor models; right plot: with interval analysis [82]. IA, TM, and MCTM stand for interval analysis, Taylor models, and McCormick-Taylor models, respectively.

the results of the unsimplified algorithms that make full use of Taylor/McCormick-Taylor models. As expected, the full algorithms yield tighter enclosures although the extra tightness is not significant here. Also, the use of convex/concave remainder bounds in McCormick-Taylor models has provided little improvement over usual Taylor models; this improvement is more apparent in the right plot where the full algorithms are used.

Although tight, the enclosures obtained from Fig. 4.3 cannot be readily used in a SBB procedure because they are not convex nor concave yet. In turn, interval and convex/concave bounds for the solutions of Problem (4.11) are obtained from the McCormick relaxations of $\mathcal{T}_{x_2(t_\mathrm{f})}$ and $\mathcal{MT}_{x_2(t_\mathrm{f})}$ for each $p \in P$. The resulting bounds and relaxations at $t = 2$ are shown in the left plot of Fig. 4.4. Here again, there is only a subtle difference between the two Taylor model-based relaxation approaches as the relaxations of $\mathcal{T}_{x_2(t_\mathrm{f})}$ and $\mathcal{MT}_{x_2(t_\mathrm{f})}$ are quite similar. These relaxations are not only enclosed by their supporting interval bounds—as guaranteed by the theory—, but these also provide large improvement. In the right plot, the results of the discretize-then-relax approach without Taylor models (Chapter 3) are shown for comparison. It is found that the use of Taylor/McCormick-Taylor models significantly tightens the interval bounds and the convex/concave relaxations.

In terms of computational effort, the propagation of 4th-order McCormick-Taylor models and 4th-order Taylor models take about 1.4 ms and 1.0 ms, respectively. The

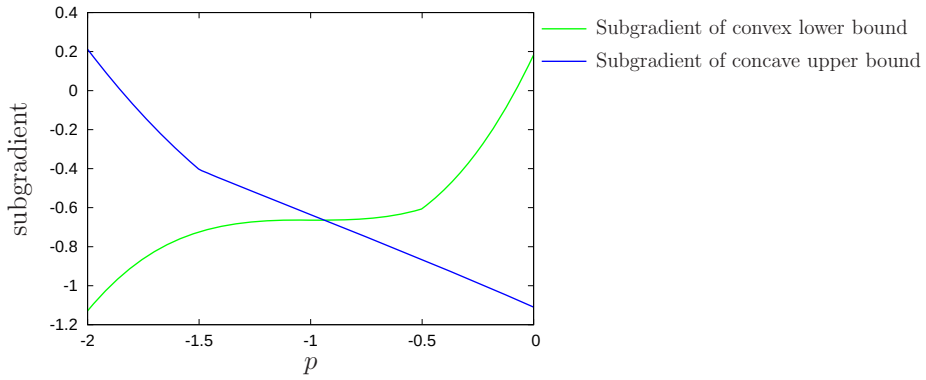

Figure 4.5: Subgradients of the convex and concave relaxations derived from the McCormick-Taylor model of x at $t = 2$ in Problem (4.11).

propagation of convex/concave bounds along with interval bounds in McCormick-Taylor models thus causes 40% overhead. These times are also to be compared with the direct propagation of McCormick relaxations and Interval bounds as presented in Chapter 3, which takes about 0.9 ms and 0.5 ms, respectively. For this simple problem, the propagation of Taylor models therefore results in a 1.5-2 fold increase in terms of computational effort. Note that, for this problem, the order of the Taylor and McCormick-Taylor models was found to have a small effect on the computational times (results not reported), probably due to the presence of a single variable. Note also that the time required for subgradient calculations are not included in the above CPU times.

Subgradients can be propagated along with the McCormick relaxations. The subgradient values computed from relaxation of McCormick-Taylor models at various points in P are shown in Fig. 4.5. Convexity and concavity of the McCormick relaxations are confirmed by the fact that the corresponding subgradients are, respectively, nondecreasing and nonincreasing in (the interior of) P. A possible use of the subgradients is in the construction of supporting affine relaxations that are more manageable than general, potentially nonsmooth, convex/concave relaxations. Computational issues regarding subgradients are discussed in Problems (4.12-4.17) and the subsequent chapter.

4.2.1.2 Lotka-Volterra System

Revisit the Lotka-Volterra problem studied in §3.2.2:

$$\dot{x}_1(t) = px_1(t)\left[1 - x_2(t)\right]; \quad x_1(0) = 1.2$$
$$\dot{x}_2(t) = px_2(t)\left[x_1(t) - 1\right]; \quad x_2(0) = 1.1,$$

for $t \geq 0$, with the variable $p \in P := [2.95, 3.05]$.

Bounding trajectories for x_1 and x_2 obtained identically from Taylor and McCormick-Taylor models are displayed in the left plots on Fig. 4.6. In addition, other bounds computed with the HOE interval method (see §2.4.1) and differential inequalities (Müller's theorem, see [103, 95]) are shown for comparison. The early breakdown of the differential inequality approach around $t = 2$ appears to be caused by inefficient treatment of the dependency problem and the wrapping effect. Also, a similar situation for the HOE interval approach around $t = 4$ is attributed to the lack of treatment of the dependency problem.

The convex and concave bounds derived from the relaxation of McCormick-Taylor models at each time step are shown in the right plots on Fig. 4.6, for the variable value $p = 3$. To reiterate, these bounds are guaranteed to be no weaker than their interval analogs by construction.

The performance of the Taylor model and McCormick-Taylor model ODE methods (both full and simplified variants) against various (McCormick-)Taylor model orders q are reported in Table 4.1. Note that the results in Table 4.1 exclude the time required for subgradient calculations. Quite expectedly, an increase of the order delays the breakdown time of the solvers, yet this is at the price of higher computational burden. A comparison of the computational times for the integration between $t = 0$ to $t = 8$ also indicates that propagating McCormick-Taylor models causes a 50-100% increase with respect to Taylor models in the full algorithms. This overhead is reduced to 20-40% in the simplified algorithms. Also, the simplified algorithms are 5-10 fold more efficient than their full counterparts. Although this huge improvement comes at the price of an earlier breakdown time of the verified solvers due to the minimal use of (McCormick-)Taylor models.

The next experiment investigates the computational efforts of Taylor and McCormick-Taylor model ODE methods for various Taylor series expansion orders k, in the absence of subgradient calculations. As reported in Table 4.2, it is seen that the

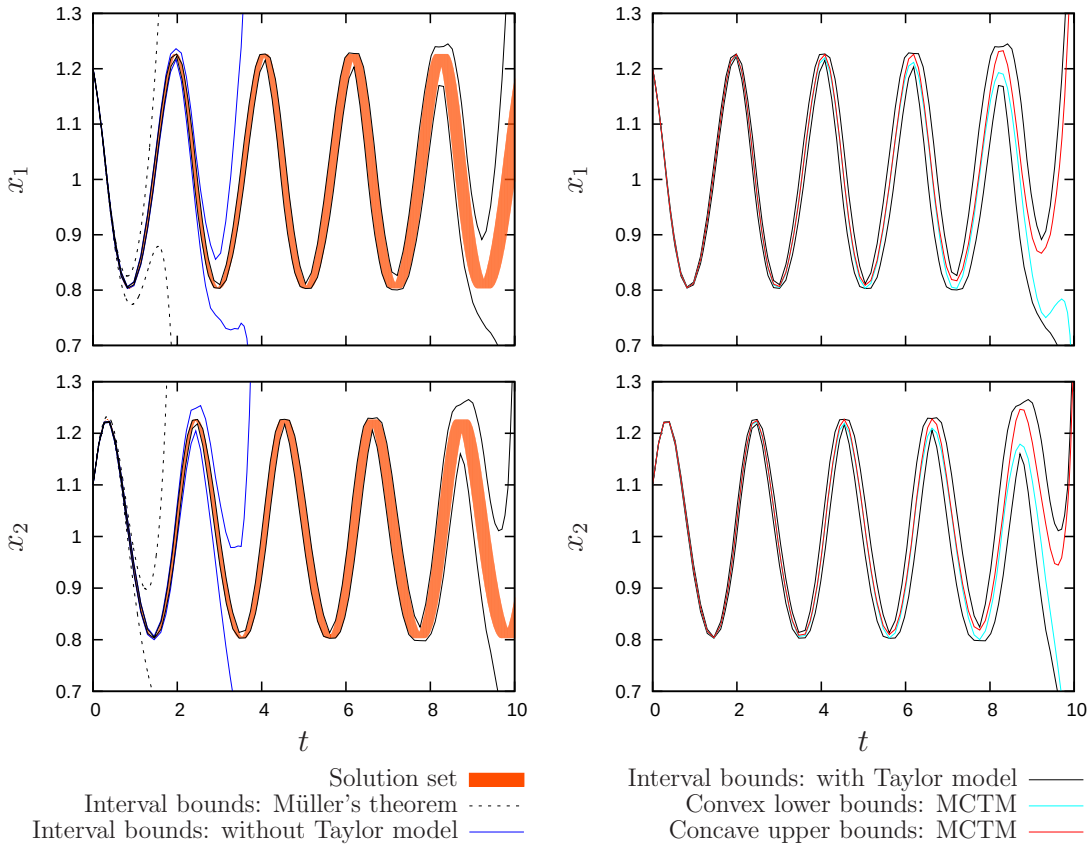

Figure 4.6: Interval bounds (left plot) and convex/concave bounds at $p = 3$ (right plot) for x_1 and x_2 in Problem (3.18,3.19). MCTM stands for McCormick-Taylor model.

Table 4.1: Comparison of Taylor model-based methods for interval bounds and McCormick relaxations in Problem (3.18,3.19); TM and MCTM stand for the Taylor model and McCormick-Taylor model methods, respectively.

		Order q	2	3	4	6	8
Full	Algorithms	TM, $0 \le t \le 8$	147 ms	151 ms	156 ms	201 ms	246 ms
		MCTM, $0 \le t \le 8$	283 ms	289 ms	337 ms	421 ms	520 ms
		Steps Taken, $0 \le t \le 8$	71	63	63	63	63
		Breakdown Time	$t \approx 8.15$	$t \approx 10.71$	$t \approx 12.50$	$t \approx 15.89$	$t \approx 17.95$
Simplified	Algorithms	TM, $0 \le t \le 8$	–	30 ms	36 ms	39 ms	42 ms
		MCTM, $0 \le t \le 8$	–	42 ms	44 ms	51 ms	55 ms
		Steps Taken, $0 \le t \le 8$	–	64	63	63	63
		Breakdown Time	$t \approx 6.92$	$t \approx 8.80$	$t \approx 10.17$	$t \approx 11.83$	$t \approx 11.98$

number of integration steps decreases as higher-order Taylor series expansions are used, thereby reducing the computational effort from $k = 5$ up to $k = 10$. However, the computational effort increases again for $k = 20$. This is due to the overhead for computing high-order Taylor coefficients and their Jacobians that eventually offsets the improvement by a reduced number of integration steps.

Table 4.2: Effect of Taylor series expansion order on the performance of Taylor model-based methods for interval bounds and McCormick relaxations in Problem (3.18,3.19).

Taylor Series Expansion Order	5	7	10	20
Taylor models, $0 \le t \le 8$	82 ms	38 ms	31 ms	49 ms
McCormick-Taylor models, $0 \le t \le 8$	111 ms	55 ms	43 ms	69 ms
Steps Taken, $0 \le t \le 8$	330	120	63	35

Interval bounds and convex/concave relaxations derived from Taylor models and McCormick-Taylor models of x_1 and x_2 at $t = 8$, are displayed in Fig. 4.7. Here again, the convex/concave relaxations are seen to provide a much tighter enclosure than the supporting interval bounds. However, the relaxations from McCormick-

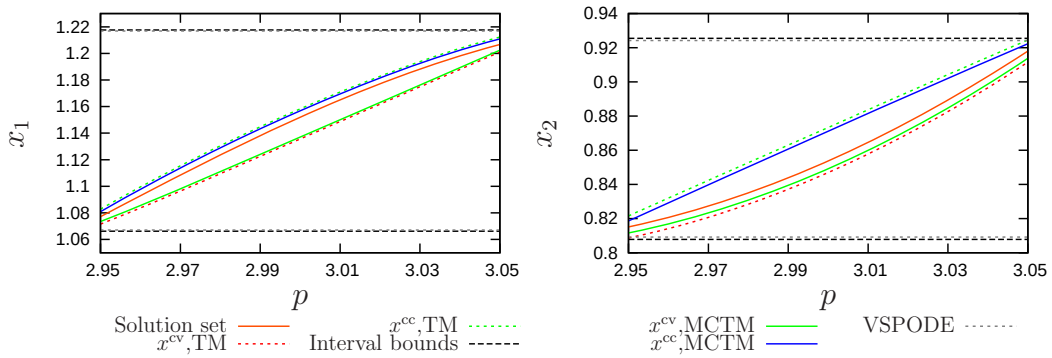

Figure 4.7: Interval and convex/concave bounds for x_1 (left plot) and x_2 (right plot) at $t = 8$ in the problem (3.18,3.19).

Taylor models offer no noticeable improvement over those from Taylor models, as also observed in Problem (4.11).

For validation, the bounds obtained with the solver VSPODE [49] using similar settings, are also shown on Fig. 4.7. It is seen that these bounds are almost the same as those obtained with the procedure described in §2.4.2 despite the fact that the two procedures employ different techniques to deal with the wrapping effect.

4.2.1.3 Irreversible Series Reaction Problem

Consider the following ODEs that represent a first-order irreversible series reaction [36, 70]:

$$\dot{x}_1 = -p_1 x_1; \qquad x_1(0) = 1 \tag{4.12}$$

$$\dot{x}_2 = p_1 x_1 - p_2 x_2; \quad x_2(0) = 0, \tag{4.13}$$

with $t \in [0, 1]$, and $(p_1, p_2) \in \mathbf{P} := [0, 1] \times [0, 1]$.

The convex and concave relaxations derived from the McCormick-Taylor models of x_1 and x_2 on $\mathbf{P}$ at $t = 1$ are displayed in Fig. 4.8; the relaxations from the Taylor model counterparts are quite the same, and not shown in this figure. Also, neither the solution set of (4.12,4.13) nor the supporting interval bounds are represented for the sake of clarity. It is noted, however, that the supporting interval bounds for x_1

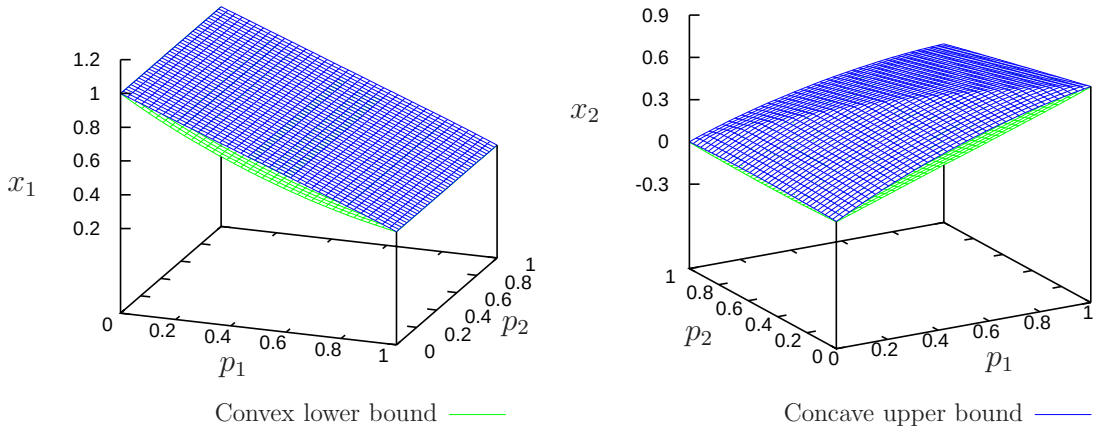

Figure 4.8: Convex and concave bounds of x_1 (left plot) and x_2 (right plot) at $t = 1$ in Problem (4.12,4.13).

and x_2 at $t = 1$ are $[0.37, 1]$ and $[-0.16, 0.66]$, respectively. These interval bounds are again much tighter than the ones computed without Taylor models (see §2.4.1), which are $[-0.16, 1.37]$ and $[-1.14, 1.53]$ at $t = 1$.

In the next experiment, the effects of number of variables and subgradient calculations on the performance of the Taylor/McCormick-Taylor model ODE methods are studied. To this end, Problem (4.12, 4.13) is modified as a one-variable and a four-variable problem given by:

$$\dot{x}_1 = -px_1; \qquad x_1(0) = 1 \tag{4.14}$$
$$\dot{x}_2 = px_1 - 0.5x_2; \quad x_2(0) = 0, \tag{4.15}$$

with $t \in [0, 1]$, and $p \in P := [0, 1]$, and

$$\dot{x}_1 = -p_1x_1; \qquad x_1(0) = p_3 \tag{4.16}$$
$$\dot{x}_2 = p_1x_1 - p_2x_2; \quad x_2(0) = p_4, \tag{4.17}$$

with $t \in [0, 1]$, and $(p_1, p_2, p_3, p_4) \in \mathbf{P} := [0, 1] \times [0, 1] \times [0.98, 1.02] \times [0, 0.02]$. A fixed (McCormick-)Taylor order $q = 6$ is considered for all the cases. The performance of the simplified algorithms is given in Table 4.3, for the various number of variables. In the McCormick-Taylor model case, the CPU times are reported with and without

the propagation of subgradients of the convex/concave remainder bounds. In is seen that increasing the number of variables from 1 to 4 causes a 6-fold increase in the computational time of both Taylor and McCormick-Taylor models. This significant increase is due to the propagation of multivariate polynomials that becomes more expensive when more variables are involved. A column-wise comparison shows that the computation of convex/concave remainder bounds in McCormick-Taylor models imposes 20-30% overhead compared to Taylor models. Nonetheless, the convex/concave bounds without subgradients may not be useful for global optimization because they are generally nonsmooth, and often have to be linearized using subgradients. The computation of subgradients adds an additional overhead to McCormick-Taylor models; 60-70% compared to the case without subgradients. Consequently, McCormick-Taylor models with subgradients are seen to be over 100% more expensive than usual Taylor models. Recall from §2.3.3 that subgradient calculations in this work are hanlded by Mitsos et al.'s procedure [61], which follows a forward automatic differentiation scheme [38, 39].

Table 4.3: Effect of number of variables n_p on performance of Taylor/McCormick-Taylor models of order $q = 6$ (with and without subgradient calculations) in Problems (4.12-4.17).

n_p	1	2	4
Steps Taken	3	3	3
Taylor models	1.3 ms	2.2 ms	7.8 ms
McCormick-Taylor models (without subgradients)	1.7 ms	2.9 ms	9.6 ms
McCormick-Taylor models (with subgradients)	2.8 ms	5.0 ms	16.4 ms

4.2.2 Convergence Order of Taylor Model-based Bounds/Relaxations for ODEs

Besides the tightness, the convergence rate of the Taylor/McCormick-Taylor model enclosures is important to the efficiency of SBB. For factorable functions, Bompadre et al. [20] analyzed the convergence of Taylor/McCormick-Taylor models, with focus

on the remainder term. They showed that, in general, the convex/concave remainder bounds in McCormick-Taylor models do not yield a higher convergence order compared to interval remainder bounds in Taylor models. That is, the inequality $\gamma, \beta \geq q+1$ also holds for a qth-order McCormick-Taylor model. Furthermore, the bounds on pointwise and Hausdorff convergence of a qth-order McCormick-Taylor model are obtained from the same expression used for Taylor models (see (2.14,2.15)).

The numerical studies in this subsection aim to investigate convergence orders of Taylor/McCormick-Taylor models for ODE systems. The discussions herein rely on the results presented in §2.3.4. The application of those results to dynamic systems is made possible by the ODE methods presented in §2.4.2 and §4.1.2. The Lotka-Volterra system used in §3.2.2 and §4.2.1.2 is considered below.

4.2.2.1 Lotka-Volterra System Revisited

Consider the Lotka-Volterra problem (3.18,3.19) with $t \in [0,4]$ and the variable $p \in P = [-0.5, 0.5]$. In this problem, the Hausdorff and pointwise convergence orders of the polynomial, remainder bound, and the whole Taylor/McCormick-Taylor models at $p = 0$ are desired. To this end, the variable set is written as $P = [-\frac{\epsilon}{2}, \frac{\epsilon}{2}]$ with $\epsilon \in [0,1]$ so that reducing $\epsilon = \mathrm{w}(P)$ shrinks P toward $p = 0$. The integrations are performed with the Taylor series expansion order $k = 10$, a variable stepsize strategy enabled by the LEPUS approach [64], and the initial stepsize $h_0 = 0.01$.

As a first experiment, the Lotka-Volterra problem is solved using Taylor models. Figure 4.9 shows the convergence results for the Taylor model of x_2 at $t_\mathrm{f} = 4$, $\mathcal{T}_{x_2(t_\mathrm{f})}$, corresponding to different Taylor model orders q. For this single-variable problem, a brute force optimization approach is utilized to obtain the infimum and supremum values as well as the function enclosure widths required in the calculations. In the left plots on Fig. 4.9, the Hausdorff and pointwise metrics for the polynomial estimator $[\mathcal{P}^\mathrm{L}_{x_2}, \mathcal{P}^\mathrm{U}_{x_2}]$ on P are depicted. This estimator is computed by the range bounder approach presented in §2.3.2. The pointwise convergence order of the polynomial estimator is seen to be $\gamma_\mathcal{P} = 1$ regardless of the Taylor model order. Regarding the Hausdorff convergence, the order $\beta_\mathcal{P} = 3$ is observed for $q > 2$. For $q = 2$, no plot can be seen in the log scale because the Hausdorff convergence is infinite. This is due to the fact that the polynomial range bounder results in the exact enclosure (i.e. image) for a univariate second-order polynomial. Therefore, the approximation error is zero in the Hausdorff metric.

The convergence of the remainder estimator $[r^{\mathrm{L}}_{x_2}, r^{\mathrm{U}}_{x_2}]$ is given in the center plots on Fig. 4.9. In these plots, the remainder function $r_{x_2(t_\mathrm{f})}(p)$ is computed as $r_{x_2(t_\mathrm{f})}(p) = x_2(t_\mathrm{f}; p) - \mathcal{P}_{x_2(t_\mathrm{f})}(p)$ for each $p \in P$ [20]. The pointwise convergence orders γ_r for $q = 2, 3, 4$ are 3, 4, and 5, respectively. This shows that the lower limit $\gamma_r \geq q + 1$ is sharp is these cases. However, for the Taylor model orders $q = 5$ and $q = 6$, the pointwise convergence orders γ_r are slightly higher than their lower limits 6 and 7, respectively. The Hausdorff convergence plots of the remainder estimator verify $\beta \geq \gamma$; also they show that different Taylor model orders can lead to the same convergence.

The convergence results of the whole Taylor model are illustrated in the right plots on Fig. 4.9. The inclusion functions associated with the Taylor models are obtained by (2.11). The pointwise convergence of Taylor models is found to be $\gamma_\mathcal{T} = 1$ irrespective of the Taylor model order. This is in fact in compliance with the theoretical result that $\gamma_\mathcal{T} = \min\{\gamma_\mathcal{P}, \gamma_r\}$ provided $\gamma_p = 1$ for all q orders. Also, the Hausdorff convergence orders are all $\beta_\mathcal{T} = 3$, which comply with the lower limit $\beta_\mathcal{T} \geq \min\{\max\{q + 1, \alpha_\mathcal{P}\}, \beta_\mathcal{P}, \beta_r\}$ given by (2.14). Provided the range order of all the polynomials is $\alpha_\mathcal{P} = 1$ (see Fig. 4.10), one has

$$\beta_\mathcal{T} \geq \min\{\max\{q + 1, 1\}, 3, \beta_r\}.$$

Since $q \geq 2$ in these experiments, $\max\{q + 1, 1\} = q + 1$, which gives:

$$\beta_\mathcal{T} \geq \min\{q + 1, 3, \beta_r\}.$$

Also, since $q + 1 \geq 3$, one has $\beta_r \geq q + 1 \geq 3$. Therefore, the bound on the Hausdorff convergence of the whole Taylor model is $\beta_\mathcal{T} \geq 3$ in this example, which turns out to be independent of the Taylor model order q.

In the second experiment, McCormick-Taylor models are used in solving the Lotka-Volterra problem. Convergence results of the McCormick-Taylor model of x_2 at t_f, $\mathcal{MT}_{x_2(t_\mathrm{f})}$, are plotted in Fig. 4.11. Unlike Fig. 4.9 where the inclusion functions were obtained from interval bounds, the inclusion functions here are computed from convex/concave relaxations of the corresponding terms. The left plots show the convergence of the polynomial estimators $[\mathcal{P}^{\mathrm{cv}}_{x_2}, \mathcal{P}^{\mathrm{cc}}_{x_2}]$ in the pointwise and Hausdorff metric. In the pointwise sense, a quadratic convergence is observed for the polynomial estimators for all Taylor model orders. This is to be compared with the linear convergence of the polynomial estimators $[\mathcal{P}^{L}_{x_2}, \mathcal{P}^{U}_{x_2}]$ in Fig. 4.9. More interestingly,

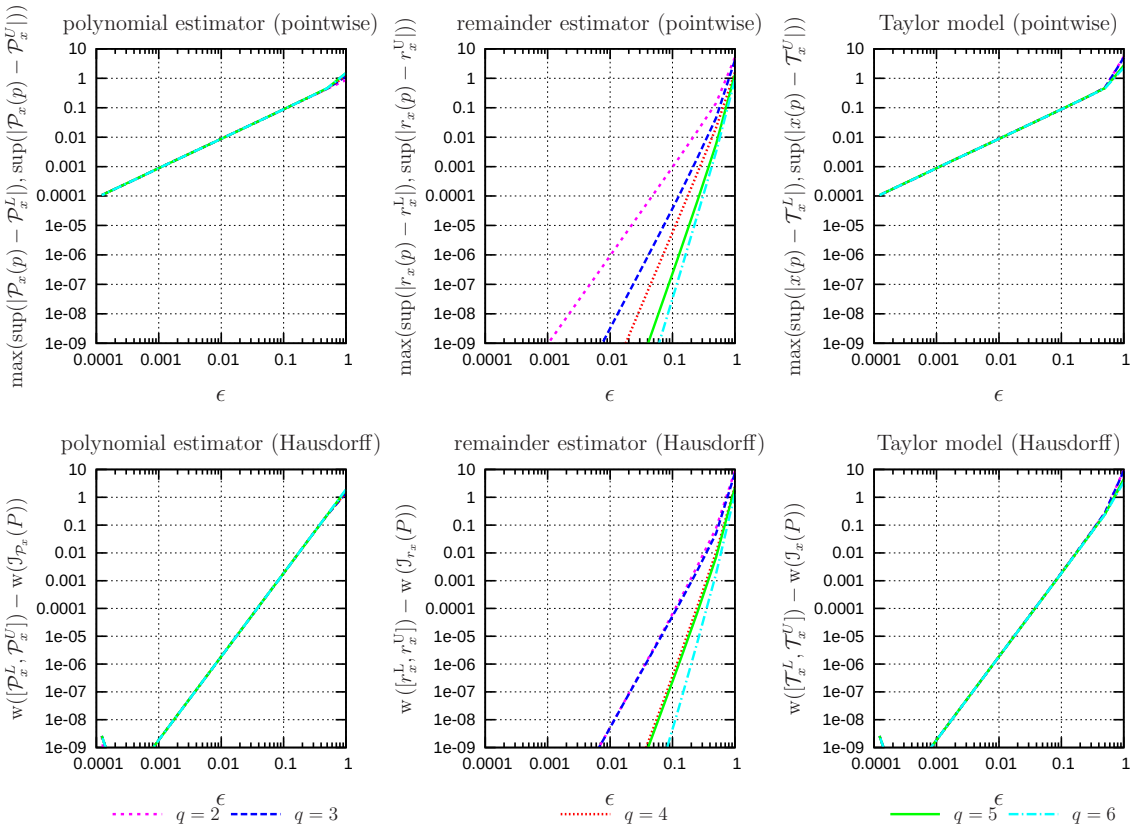

Figure 4.9: Pointwise and Hausdorff convergence of Taylor model $\mathcal{T}_{x_2(t_\mathrm{f})}(p)$ with interval bounding of polynomial from the solution of the Lotka-Volterra system. The index subscript of x_2 is omitted for notational simplicity.

the relaxations $[\mathcal{P}^{\mathrm{cv}}_{x_2}, \mathcal{P}^{\mathrm{cc}}_{x_2}]$ show a very high (in fact infinite) convergence order in the Hausdorff sense, for all the McCormick-Taylor model orders. For $q = 2$, the infinite convergence is expected since the supporting interval bounds give the exact enclosure (i.e. image), and the polynomial relaxation is guaranteed to be no looser than it. Therefore, the Hausdorff approximation error is zero. For the higher orders, the infinite convergence is probably due to monotonic relaxations that touch the polynomial function at the variable bounds (see Fig. 4.12). Therefore, again there is no approximation error in the Hausdorff sense. It is needless to say that relaxation of the polynomial term in Taylor models would give exactly the same results as in McCormick-Taylor models because both share the same polynomial.

The convergence results of the convex/concave remainder estimators $[r^{\mathrm{cv}}_{x_2}, r^{\mathrm{cc}}_{x_2}]$, as

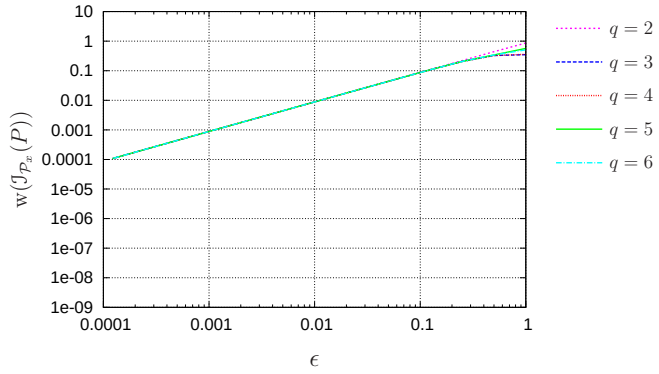

Figure 4.10: The range of the polynomial $\mathcal{P}_{x_2}(p)$ versus the width of the variable set.

shown in the center plots on Fig. 4.11, are the same as those of the interval remainder estimators given in Fig. 4.9. This suggests that the convex/concave remainder estimators in McCormick-Taylor models do not increase the convergence order practically over interval remainder estimators in Taylor models [20]. The convergence of the entire McCormick-Taylor model is illustrated in the right plots on Fig. 4.11. The inclusion functions associated with the McCormick-Taylor models are obtained by

$$\mathcal{F}_{x_2}(P) := [\inf_p \mathcal{MT}_x^{cv}(p), \sup_p \mathcal{MT}_x^{cc}(p)] = [\inf_p (\mathcal{P}_{x_2}^{\mathrm{cv}}(p) + r_{x_2}^{\mathrm{cv}}(p)), \sup_p (\mathcal{P}_{x_2}^{\mathrm{cc}}(p) + r_{x_2}^{\mathrm{cc}}(p))].$$

As expected, the pointwise convergence orders comply with $\gamma_{\mathcal{MT}} = \min\{\gamma_{\mathcal{P}}, \gamma_r\} = \min\{2, \gamma_r\} = 2$ provided $\gamma_r \geq q + 1 \geq 3$ for all q orders here. Also, regarding the Hausdorff convergence, it is easy to check that $\beta_{\mathcal{MT}} \geq \min\{\max\{q + 1, \alpha_{\mathcal{P}}\}, \beta_{\mathcal{P}}, \beta_r\}$ for each of the q orders (see similar arguments for Fig. 4.9).

4.3 Conclusions

In this chapter, the combination of McCormick's relaxations with Taylor models has been proposed in order to produce tight convex/concave relaxations of state variables. The combination was motivated by Taylor models outperforming relaxations derived from interval analysis (see Chapter 3), and is achieved in two forms.

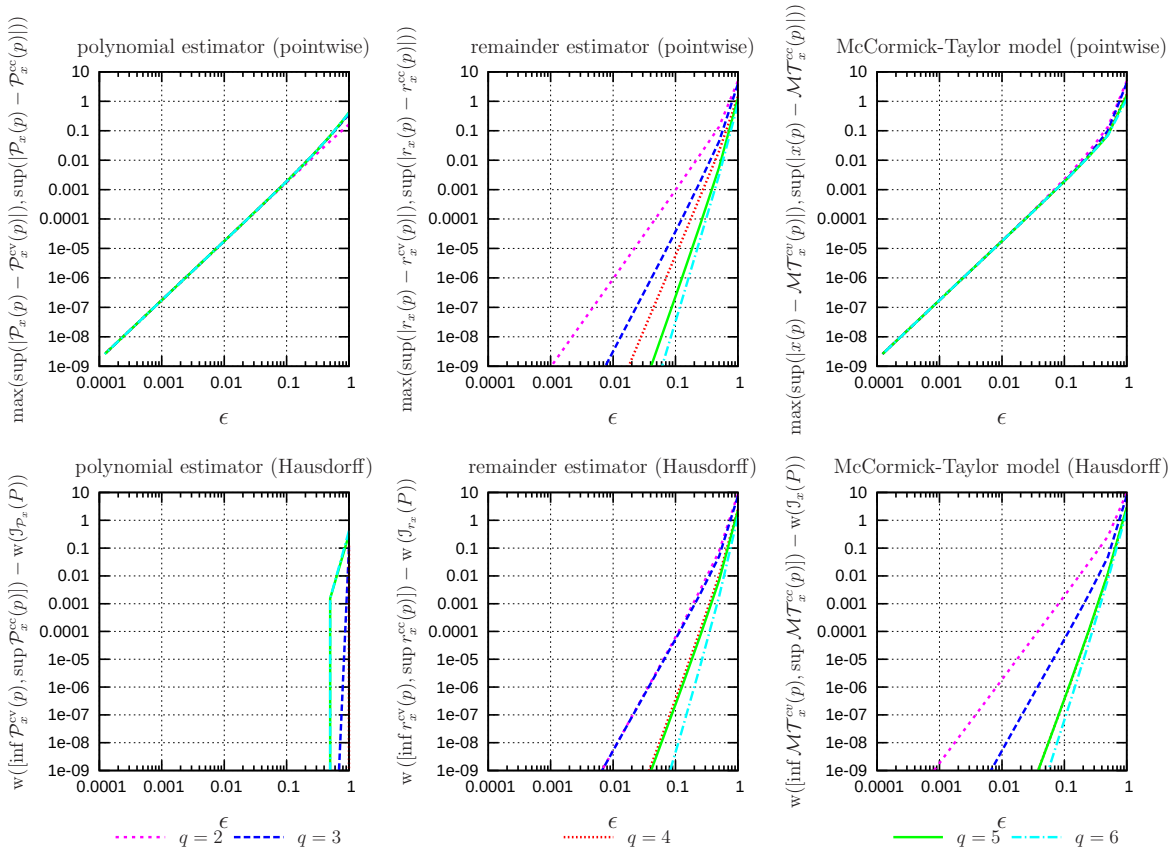

Figure 4.11: Pointwise and Hausdorff convergence of McCormick-Taylor model $\mathcal{MT}_{x_2(t_f)}(p)$ with relaxation of polynomial from the solution of the Lotka-Volterra system. The index subscript of x_2 is omitted for notational simplicity.

The first form considers McCormick relaxation of Taylor models, which are obtained with the Taylor model ODE method developed in [49]. The second form proposes McCormick relaxation of McCormick-Taylor models: a new type of Taylor models where convex/concave bounds of the remainder term are computed in addition to the interval remainder bounds. A two-phase procedure, similar to the verified procedure proposed for Taylor models [49], has also been developed in order to compute McCormick-Taylor models of state variables.

Numerical experiments demonstrate the effectiveness of Taylor model based relaxations in generating tight convex/concave approximation of nonlinear dynamic models. In theory, such relaxations are no looser than the interval bounds from

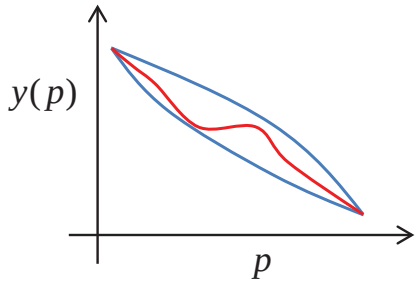

Figure 4.12: Example of monotonic convex/concave relaxations that also reach the function at the variable bounds.

Taylor models, but also are seen to be much tighter in practice. In terms of computational effort, the first form of combination of relaxations with Taylor models comes with no additional cost (practically) compared to usual bounding of Taylor models. However, the McCormick-Taylor models used in the second form impose a considerable overhead while usually making negligible improvement over the first form, where usual Taylor models are used.

Convergence properties of Taylor/McCormick-Taylor models were also studied in this chapter. The case studies verified the theoretical results in [20] for the remainder term, and showed that convex/concave remainder bounds in McCormick-Taylor models do not provide a higher convergence order than interval remainder bounds in general. Therefore, relaxation of the remainder term may not contribute to increased convergence for the whole McCormick-Taylor model. Rather, relaxation of the polynomial term can have a positive impact on the convergence of the McCormick-Taylor model, especially in the Hausdorff sense. Interestingly enough, the polynomial relaxations can be done in Taylor models too. Therefore, one can use Taylor models and obtain similar convergence to McCormick-Taylor models, but with less computational demand.

In conclusion, relaxations from Taylor models show much promise as an effective lower bounding strategy for GDO. However, such relaxations are nonlinear and (potentially) nonsmooth, which would cause difficulties in their reliable and efficient solution by conventional NLP techniques. This issue will be addressed in the next chapter, where a polyhedral relaxation approach is developed.

Chapter5

Global Dynamic Optimization using Polyhedral Relaxations

As mentioned in §2.5.3, each node in SBB involves computing feasible lower/upper bounds on the objective function of Problem (2.3) over a partition of the variable space. An upper bound can be obtained by any local search method, e.g. gradient-based optimization techniques. A lower bound can be obtained from interval bounding or convex relaxation techniques. It is known that relaxations often yield faster convergence compared to interval bounds by providing tighter enclosures and higher convergence rates [98, 42] (see also Figs. 4.9-4.11 and related discussions).

In this chapter, the developments presented in the previous chapter are embedded in a SBB procedure. The relaxations derived in Chapter 4 would provide tight enclosures in SBB, which is desired for an efficient GDO algorithm. Nonetheless, the direct use of such relaxations is not suitable due to the following reasons.

I. The relaxations are nonlinear, and thus require NLP solvers, which are computationally expensive for large-scale dynamic problems.

II. Robustness issues associated with NLP solvers increase the likelihood of solution failure or unreliable results. For example, a feasible problem may appear to an NLP solver as locally infeasible during the numerical solution. If this occurs

in a subspace where a global solution exists, the algorithm may incorrectly discard that global solution. This is obviously not desirable in a rigorous global optimization algorithm[1].

III. The nonsmoothness of McCormick relaxations makes them unsuitable for conventional NLP solvers that require differentiability of the optimization model.

A simple remedy to the nonlinearity/nonsmoothness of McCormick relaxations is their linearization at some arbitrary point, provided subgradient values are propagated along with the McCormick relaxations (see §2.3.3.1). However, the tightness of the resulting affine relaxations depends on the linearization point as well as the subgradient value. As shown in Fig. 5.1, the affine relaxation obtained at the point p_1 brings no improvement to the underlying interval bound. Unfortunately, there is no prior knowledge as to what point gives the tightest affine relaxation.

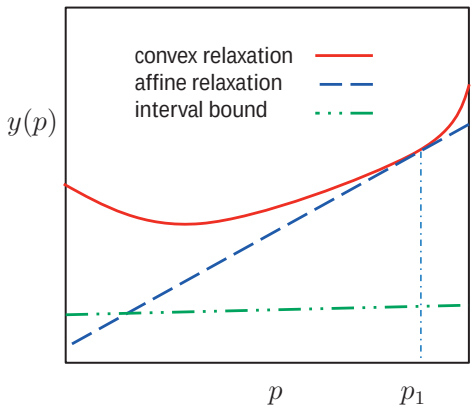

Figure 5.1: Naive linearization of a convex relaxation at a given point p_1.

To overcome the aforementioned issues, this work considers polyhedral relaxations [97, 99] of Taylor/McCormick-Taylor models to enable tight linear outer-approximations of nonconvex problems. This approach will be used in a SBB procedure to form a GDO algorithm. For performance comparisons, the GDO algorithm in [47, 48], which is solely based on Taylor model-derived interval bounds, is also used within a similar SBB procedure. For clarity of subsequent discussions, this algorithm is first outlined in §5.1. Then, the proposed algorithm is presented in §5.2.

[1]In principle, robustness issues should not occur for convex NLP problems since gradient-based methods have global convergence properties for such problems. However, a solution failure could happen due to numerical issues (e.g. ill-conditioning).

The SBB procedure utilizing the proposed relaxation algorithm is outlined in §5.3. The software implementation of the entire GDO algorithm is described in §5.4. The effectiveness of the proposed algorithm is demonstrated via benchmark problems in §5.5. Finally, the chapter is closed by some concluding remarks in §5.6.

5.1 GDO Algorithm using Taylor Model-derived Interval Bounds

The Taylor model based GDO algorithm in [47, 48] involves Taylor model integration of the parametric ODEs (2.1) as discussed in §2.4.2 (simplified variant). The Taylor models of state variables at the desired time[2] (e.g. final time t_f), $\mathcal{T}_{\mathbf{x}_{t_\mathrm{f}}}(\mathbf{p})$, are used to compute Taylor models of the objective and constraint functions in (2.3). Then, the Taylor model of the objective function, $\mathcal{T}_{\mathcal{J}(\mathbf{p})} = \phi(\mathcal{T}_{\mathbf{x}(t_\mathrm{f})}, \mathbf{p})$, is bounded to obtain a lower bound on Problem (2.3) at each node during the branch-and-bound procedure.

The problem constraints are handled using a domain reduction technique tailored to Taylor models. Applying this technique to the constraints can shrink the variable bounds, thereby accelerating the branch-and-bound procedure. More details are presented in the following subsection.

5.1.1 Constraint Handling and Taylor Model-based Domain Reduction

Lin and Stadtherr [47] developed a type of constraint propagation procedure by exploiting the parametric expressions from Taylor models. Given a Taylor model $\mathcal{T}_g(\mathbf{p})$ for the constraint $g(\mathbf{p}) \leq 0$ at a node characterized by the domain $\mathbf{P}'$, the constraint propagation on Taylor models (CPT) proceeds as follows. First, $\mathcal{T}_g(\mathbf{p})$ is bounded over $\mathbf{P}'$ to give $[g^\mathrm{L}, g^\mathrm{U}]$. Then, one of the following cases occurs:

- $g^\mathrm{L} > 0$, which means no $\mathbf{p} \in \mathbf{P}'$ can satisfy the constraint. The node $\mathbf{P}'$ is then

[2] Without loss of generality, the ensuing discussions assume that all the desired times t_i in (2.3) are t_f.

safely discarded from the SBB search.

- $g^{\mathrm{U}} \leq 0$, which means every $\mathbf{p} \in \mathbf{P}'$ satisfies the constraint. The CPT will not be helpful in this case, and the node $\mathbf{P}'$ is retained intact.

- Neither of the above situations occurs, in which case, it cannot be decided readily if any parts of $\mathbf{P}'$ violate the constraint. If there are any infeasible parts in $\mathbf{P}'$, they may be potentially identified and excluded by applying the CPT procedure to $\mathcal{T}_g(\mathbf{p}) \leq 0$. The (potentially) tightened $\mathbf{P}'$ is then retained for further partitioning by SBB. This in turn enables tighter lower bounds in the subsequent nodes, thereby speeding up the convergence. If the CPT reveals that $\mathbf{P}'$ is infeasible, the node $\mathbf{P}'$ is entirely removed from the SBB search. Mathematical details of the CPT procedure can be found in [47, 48].

In addition to the original problem constrains, an auxiliary constraint can be appended at each node to help further shrink the search space. This constraint reads

$$C_{\mathcal{J}}(\mathbf{p}) := \mathcal{J}(\mathbf{p}) - \mathcal{U} \leq 0, \tag{5.1}$$

with $\mathcal{U}$ the incumbent in the branch-and-bound procedure (see §2.5.3). The violation of this constraint for any subdomain $\mathbf{P}'' \subseteq \mathbf{P}'$ means that the optimization problem in that subdomain cannot contain an improving solution over the incumbent $\mathcal{U}$. Thus, $\mathbf{P}''$ can be removed from the current partition. To potentially discard non-improving subdomains in $\mathbf{P}'$, the CPT is performed on the Taylor model form of $C_{\mathcal{J}}(\mathbf{p})$ as $\mathcal{T}_{C_{\mathcal{J}}}(\mathbf{p}) = \mathcal{T}_{\mathcal{J}}(\mathbf{p}) - \mathcal{U} \leq 0$.

In addition to inequality constraints, equality constraints such as $\mathbf{s}(\mathbf{p}) = \mathbf{0}$ can be handled by this procedure. To do so, $\mathbf{s}(\mathbf{p})$ is split into two sets of inequality constraints $\mathbf{s}(\mathbf{p}) \leq \mathbf{0}$ and $-\mathbf{s}(\mathbf{p}) \leq \mathbf{0}$, and the CPT is performed on each set separately.

5.2 GDO Algorithm using Polyhedral Relaxations from Taylor Models and McCormick-Taylor Models

An improved algorithm for relaxation of nonlinear dynamic problems as (2.3) is presented here. It builds upon the Taylor/McCormick-Taylor model ODE methods and polyhedral relaxations. The choice of using Taylor models or McCormick-Taylor models leads to two algorithm variants, whose procedures are quite similar. To avoid repetition, the Taylor model variant is presented below; the McCormick-Taylor model variant is noted in brackets, and discussed explicitly only where necessary.

The relaxation algorithm proceeds in the following steps.

Step I. Compute Taylor [McCormick-Taylor] models for state variables. Apply the algorithm presented in §2.4.2 [§4.1.2] to (2.1)-(2.2) to compute Taylor [McCormick-Taylor] models of the state variables, i.e. $\mathcal{T}_{\mathbf{x}(t_\mathrm{f})}(\mathbf{p})$ [$\mathcal{MT}_{\mathbf{x}(t_\mathrm{f})}(\mathbf{p})$].

Step II. Compute Taylor [McCormick-Taylor] models for objective/constraint functions. Compute Taylor [McCormick-Taylor] models for the objective function and constraints as $\mathcal{T}_\mathcal{J}(\mathbf{p})$, $\mathcal{T}_\mathbf{g}(\mathbf{p})$, and $\mathcal{T}_\mathbf{s}(\mathbf{p})$ [$\mathcal{MT}_\mathcal{J}(\mathbf{p})$, $\mathcal{MT}_\mathbf{g}(\mathbf{p})$, and $\mathcal{MT}_\mathbf{s}(\mathbf{p})$] based on the results from Step I.

Step III. Construct a non-convex relaxation problem. Construct a *non-convex* relaxation of (2.3) using the Taylor [McCormick-Taylor] models obtained from Step II. Recall that a Taylor [McCormick-Taylor] model $\mathcal{T}_f(\mathbf{p}) = \mathcal{P}_f(\mathbf{p}) + R_f$ [$\mathcal{MT}_f(\mathbf{p}) = \mathcal{P}_f(\mathbf{p}) + \mathcal{M}_{R_f}(\mathbf{p})$] is generally a *non-convex* relaxation of its corresponding function $f(\mathbf{p})$ (see e.g. Figs. 4.1,4.3). Such a relaxation can be expressed as $\{\mathcal{P}_f(\mathbf{p}) + r_f \,|\, r_f \in R_f\} \ni f(\mathbf{p})$ [$\{\mathcal{P}_f(\mathbf{p}) + r_f \,|\, r_f \in [r_f^{\mathrm{cv}}, r_f^{\mathrm{cc}}](\mathbf{p})\} \ni f(\mathbf{p})$]. As a result, a *non-convex* relaxation of (2.3) on $\mathbf{P}'$ for the Taylor model variant can be

formed as:

$$\begin{aligned}\min_{\mathbf{p},r_{\mathcal{J}},r_{\mathbf{g}},r_{\mathbf{s}}} \quad & \mathcal{P}_{\mathcal{J}}(\mathbf{p}) + r_{\mathcal{J}} \\ \text{s.t.} \quad & \mathcal{P}_{\mathbf{g}}(\mathbf{p}) + r_{\mathbf{g}} \leq 0 \\ & \mathcal{P}_{\mathbf{s}}(\mathbf{p}) + r_{\mathbf{s}} = 0 \\ & r_{\mathcal{J}} \in R_{\mathcal{J}}, r_{\mathbf{g}} \in R_{\mathbf{g}}, r_{\mathbf{s}} \in R_{\mathbf{s}} \\ & \mathbf{p} \in \mathbf{P'},\end{aligned} \tag{5.2}$$

where r are the relaxation variables that are bounded by their corresponding interval remainder bounds R.

For the McCormick-Taylor model variant, the bound constraints $r \in R$ in (5.2) can be augmented by the constraints $r^{\text{cv}}(\mathbf{p}) \leq r \leq r^{\text{cc}}(\mathbf{p})$. However, the functions $r^{\text{cv}}(\mathbf{p})$ and $r^{\text{cc}}(\mathbf{p})$ are not known in explicit form, and obtaining them for each $\mathbf{p} \in \mathbf{P'}$ requires repeating the integration of McCormick-Taylor models for that $\mathbf{p}$. Since an LP problem is eventually sought, the constraints $r^{\text{cv}}(\mathbf{p}) \leq r \leq r^{\text{cc}}(\mathbf{p})$ can be linearized using affine relaxations. This will yield the following non-convex relaxation of (2.3) on $\mathbf{P'}$ for the McCormick-Taylor model variant:

$$\begin{aligned}\min_{\mathbf{p},r_{\mathcal{J}},r_{\mathbf{g}},r_{\mathbf{s}}} \quad & \mathcal{P}_{\mathcal{J}}(\mathbf{p}) + r_{\mathcal{J}} \\ \text{s.t.} \quad & \mathcal{P}_{\mathbf{g}}(\mathbf{p}) + r_{\mathbf{g}} \leq 0 \\ & \mathcal{P}_{\mathbf{s}}(\mathbf{p}) + r_{\mathbf{s}} = 0 \\ & r_{\mathcal{J}} \geq r^{\text{cv}}_{\mathcal{J}}(\underline{\mathbf{p}}) + \boldsymbol{\sigma}^{\text{cv}}_{\mathcal{J},\underline{\mathbf{p}}}(\mathbf{p} - \underline{\mathbf{p}}) \\ & r_{\mathcal{J}} \leq r^{\text{cc}}_{\mathcal{J}}(\underline{\mathbf{p}}) + \boldsymbol{\sigma}^{\text{cc}}_{\mathcal{J},\underline{\mathbf{p}}}(\mathbf{p} - \underline{\mathbf{p}}) \\ & r_{\mathbf{g}} \geq r^{\text{cv}}_{\mathbf{g}}(\underline{\mathbf{p}}) + \boldsymbol{\sigma}^{\text{cv}}_{\mathbf{g},\underline{\mathbf{p}}}(\mathbf{p} - \underline{\mathbf{p}}) \\ & r_{\mathbf{g}} \leq r^{\text{cc}}_{\mathbf{g}}(\underline{\mathbf{p}}) + \boldsymbol{\sigma}^{\text{cc}}_{\mathbf{g},\underline{\mathbf{p}}}(\mathbf{p} - \underline{\mathbf{p}}) \\ & r_{\mathbf{s}} \geq r^{\text{cv}}_{\mathbf{s}}(\underline{\mathbf{p}}) + \boldsymbol{\sigma}^{\text{cv}}_{\mathbf{s},\underline{\mathbf{p}}}(\mathbf{p} - \underline{\mathbf{p}}) \\ & r_{\mathbf{s}} \leq r^{\text{cc}}_{\mathbf{s}}(\underline{\mathbf{p}}) + \boldsymbol{\sigma}^{\text{cc}}_{\mathbf{s},\underline{\mathbf{p}}}(\mathbf{p} - \underline{\mathbf{p}}) \\ & r_{\mathcal{J}} \in R_{\mathcal{J}}, r_{\mathbf{g}} \in R_{\mathbf{g}}, r_{\mathbf{s}} \in R_{\mathbf{s}} \\ & \mathbf{p} \in \mathbf{P'},\end{aligned} \tag{5.3}$$

where $\underline{\mathbf{p}} \in \text{int}(\mathbf{P'})$ is a pre-specified linearization point typically chosen as $\text{m}(\mathbf{P'})$; $\boldsymbol{\sigma}^{\text{cv}}_{\mathcal{J},\underline{\mathbf{p}}}$

and $\boldsymbol{\sigma}^{\text{cc}}_{\mathcal{J},\underline{\mathbf{p}}}$ are the subgradients of convex and concave remainder bounds of $\mathcal{MT}_{\mathcal{J}}(\mathbf{p})$ at $\underline{\mathbf{p}}$, respectively. The subgradients of convex/concave remainder bounds of $\mathcal{MT}_{\mathbf{g}}(\mathbf{p})$ and $\mathcal{MT}_{\mathbf{s}}(\mathbf{p})$ in (5.3) follow similar notations.

Step IV. Construct polyhedral relaxations. Apply polyhedral relaxations to the nonconvex problem (5.2) [(5.3)]. The resulting LP will have the same form as (2.18).

Step V. Solve polyhedral relaxations. Solve the LP problem obtained from Step IV in order to compute a feasible lower bound on the relaxed objective function in (5.2) [(5.3)], which is in turn a lower bound on the actual objective function $\mathcal{J}(\mathbf{p})$ in (2.3), for $\mathbf{p} \in \mathbf{P}'$.

Remark 5.1. *As a result of the additional constraints by the affine relaxations, Problem* (5.3) *can potentially provide a tighter lower bound for the objective function than Problem* (5.2)*. Nevertheless, the potentially extra tightness comes at the price of a higher computational burden for the McCormick-Taylor model variant. The computational aspects of the two algorithm variants in SBB are further discussed in §5.5.*

The proposed GDO algorithm employs an optimization-based domain reduction technique as discussed in the next subsection.

5.2.1 LP-based Domain Reduction

Suppose the polyhedral outer-approximation of (5.2) or (5.3) has led to the LP (2.18). At this point, the bounds on the variables $\mathbf{v}$ can be potentially tightened by solving auxiliary optimization problems. Specifically, two auxiliary LP problems are constructed for each variable in an attempt to obtain a tighter lower bound and upper bound for that variable. Given the variable $v_i \in [v_i^{\text{L}}, v_i^{\text{U}}]$, a lower bound tighter

than v_i^{L} may be obtained from solving the following LP:

$$\begin{aligned} \min_{\mathbf{v}} \; & v_i \\ \text{s.t. } & v_{\mathrm{obj}} - \mathcal{U} \leq 0 \\ & \mathbf{G}'\,\mathbf{v} \leq \mathbf{b}' \\ & \mathbf{H}\,\mathbf{v} = \mathbf{c} \\ & \mathbf{v}^{\mathrm{L}} \leq \mathbf{v} \leq \mathbf{v}^{\mathrm{U}}, \end{aligned} \tag{5.4}$$

where $\mathcal{U}$ is the incumbent in the SBB procedure. If the above problem is feasible, the lower bound v_i^{L} is updated as $v_i^{L,*}$, where $v_i^{L,*}$ is the solution of (5.4) ($v_i^{L,*} \geq v_i^{\mathrm{L}}$). The above minimization problem enforces the constraints of the relaxed problem (2.18). This way, if a variable set containing v_i^{L} as $\{[v_i^{\mathrm{L}}, v_i^{U,\mathrm{inf}}) \,|\, v_i^{\mathrm{L}} < v_i^{U,\mathrm{inf}} \leq v_i^{\mathrm{U}}\}$ is infeasible for (2.18), that set will be excluded by (5.4). As a result, $v_i^{L,*}$ will be at least as tight as $v_i^{U,\mathrm{inf}}$. Similarly, the constraint $v_{\mathrm{obj}} - \mathcal{U} \leq 0$ in (5.4) removes possible subsets of $[v_i^{\mathrm{L}}, v_i^{\mathrm{U}}]$, in which the relaxed problem (2.18) is guaranteed not to contain a solution better than the incumbent. In particular, if a subset containing v_i^{L} as $\{[v_i^{\mathrm{L}}, v_i^{U,\mathrm{nopt}}) \,|\, v_i^{\mathrm{L}} < v_i^{U,\mathrm{nopt}} \leq v_i^{\mathrm{U}}\}$ does not satisfy $v_{\mathrm{obj}} - \mathcal{U} \leq 0$, then it will be removed, and $v_i^{L,*}$ will be at least as tight as $v_i^{U,\mathrm{nopt}}$.

Another LP is solved for tightening the upper bound v_i^{U}. It is the same as (5.4), except that the objective v_i is maximized:

$$\begin{aligned} \max_{\mathbf{v}} \; & v_i \\ \text{s.t. } & v_{\mathrm{obj}} - \mathcal{U} \leq 0 \\ & \mathbf{G}'\,\mathbf{v} \leq \mathbf{b}' \\ & \mathbf{H}\,\mathbf{v} = \mathbf{c} \\ & \mathbf{v}^{\mathrm{L}} \leq \mathbf{v} \leq \mathbf{v}^{\mathrm{U}} \end{aligned} \tag{5.5}$$

If this problem is feasible, the upper bound v_i^{U} is updated as $v_i^{U,*}$, where $v_i^{U,*}$ is the solution of (5.5) ($v_i^{U,*} \leq v_i^{\mathrm{U}}$). Note that if either (5.4) or (5.5) turns out to be infeasible, the node will be deleted from the SBB search.

5.3 Branch-and-Bound Procedure

The branch-and-bound procedure that accommodates the polyhedral relaxation of Taylor/McCormick-Taylor models is presented in this section. For the sake of meaningful comparisons, the previous work [48] that makes sole use of Taylor models is also implemented within a similar procedure. The following acronyms are used to refer to the above algorithms: IBTM denotes the previous work which is based on interval bounds from Taylor models [48]; PRTM stands for the algorithm variant in this work that is based on polyhedral relaxations of usual Taylor models; PRMCTM refers to the algorithm variant in this work that is based on polyhedral relaxations of McCormick-Taylor models. A dynamic optimization problem given by (2.3) is solved to global optimality through the following SBB procedure.

Step 0: Initialization.

- Set an absolute tolerance ϵ_{abs} and a relative tolerance ϵ_{rel} for branch-and-bound.
- Define the root node with the variable space $\mathbf{P}^0 = \mathbf{P}$.
- Initialize a list of active nodes with the root node. This list contains all subsequent subdomains of $\mathbf{P}$ that must be explored before a global solution is confirmed.
- Set the index of the node $K = 0$.
- Set the incumbent $\mathcal{U} = +\infty$ and its location $\mathbf{p}^* = \emptyset$ (empty set).
- Choose the relaxation algorithm from PRTM or PRMCTM.
- Set Max_Repeat which indicates how many times the domain reduction (Step 3) can be repeated for a given node.
- Set β, a threshold for the domain reduction percentage.
- Set DR_Repeat = 0, where DR_Repeat is a counter indicating how many times the domain reduction has been repeated at a given node. Skip to Step 3.

Step 1: Termination.

- If the list of active nodes is empty, but $\mathbf{p}^* \neq \emptyset$, ϵ-convergence is achieved. Stop, and report $\mathcal{U}$ as the approximate global optimum value and $\mathbf{p}^*$ as its location.
- If both list of active nodes and $\mathbf{p}^*$ are empty, then the problem is infeasible. Stop, and report infeasibility.
- If none of the above cases occurs, continue to Step 2.

Step 2: Node Selection.

- Set $K \leftarrow K + 1$
- Select a node from the list of active nodes, and denote it by $\mathbf{P}^K$ ($\mathbf{P}^K \subset \mathbf{P}$).
- Set DR_Repeat $= 0$.
- Continue to Step 3.

Remark 5.2. *This work considers the best-first approach as the node selection strategy. See §2.5.3.2 for more information.*

Step 3: Domain Reduction.

- Obtain a polyhedral outer-approximation of (2.3) as described in §5.2.
- Construct and solve the bound tightening problems (5.4) and (5.5) for each variable in $\mathbf{P}^K$.
- If (5.4) and (5.5) do not have a feasible solution for at least one of the variables, delete Node K from the search. Return to Step 1.
- Update $\mathbf{P}^K$ with the potentially tightened variable bounds.
- If the width of any element of $\mathbf{P}^K$ has been reduced by β percent or more and DR_Repeat $<$ Max_Repeat, then set DR_Repeat $\leftarrow$ DR_Repeat $+ 1$ and repeat Step 3 with the tightened bounds $\mathbf{P}^K$. Otherwise, continue to Step 4.

Step 4: Lower Bounding Problem.

- Replace v_i with v_{obj} in (5.4), and solve the resulting LP in order to obtain a lower bound L^K for Node K.
- Delete Node K from the search if
  - no feasible lower bound L^K can be obtained; or
  - $L^K \geq \mathcal{U} - \epsilon_{\text{abs}}$ or $L^K \geq \mathcal{U} - |\mathcal{U}|\epsilon_{\text{rel}}$.
- If Node K is deleted, return to Step 1. Otherwise, continue to Step 5.

Remark 5.3. *By satisfying the inequality $L^K \geq \mathcal{U} - \epsilon_{abs}$, it is ensured that the lower bound for the current node is no smaller than the incumbent according to the preset absolute tolerance. In such a case, the node will need no further branching and can be deleted. A similar explanation holds for the inequality $L^K \geq \mathcal{U} - |\mathcal{U}|\epsilon_{rel}$, which is defined in a relative sense and according to the preset relative tolerance.*

Step 5: Upper Bounding Problem.

- Solve Problem (2.3) with $\mathbf{p} \in \mathbf{P}^K$ locally, for instance, using the sequential approach of dynamic optimization, in order to obtain an upper bound U^K for Node K.
- If the local solution of (2.3) is successful and $U^K < \mathcal{U}$, update $\mathcal{U} = U^K$, and set $\mathbf{p}^*$ as the location of the local optimum.

Step 6: Branching. Branch the current node $\mathbf{P}^K$ using a branching strategy (see §2.5.3.2), and add the newly created nodes to the list of active nodes. Go to Step 2.

Remark 5.4. *The IBTM algorithm has also been accommodated into a very similar SBB procedure. The difference is that the domain reduction step, which applies the CPT technique in* (5.1)*, is performed after the lower-bounding problem. This is because the domain reduction step requires the Taylor model of the objective function,*

and upon computing this Taylor model, the lower-bounding problem is also solved readily.

Remark 5.5. *It should be emphasized that the SBB procedure will not converge if the bounds on the global solution value do not shrink to the desired tolerance by branching on the variable bounds. This situation could particularly happen in GDO if truncation errors are bounded along with the ODE solution. Bounds on the truncation errors will have a non-zero width as long as a finite-order Taylor series expansion is used (which is always the case in practice). However, they can be shrunk to the desired precision by choosing smaller integration stepsizes and a higher-order k for the Taylor series expansion, although this remedy will increase the cost of integration. Also, the SBB may not converge if rounding errors are bounded in the computer implementation, and are significant probably due to the use of a low precision arithmetic.*

5.4 Software Implementation of GDO Solver

The GDO algorithm presented in this thesis has been implemented in the MC++ software package [24]. MC++ is a suite of C++ libraries by Dr. B. Chachuat that are developed for local/global optimization of steady-state and dynamic systems. It is a successor of libMC, which was developed in 2006 for automated computation of McCormick relaxations and their subgradients for factorable functions. During the course of this research, MC++ has become a versatile collection of libraries that perform different tasks required for global and local optimization. This includes C++ classes for validated solution of ODEs, Taylor model computations, polyhedral relaxations, SBB, and interfaces for link to third-party software. The object-oriented design of MC++ allows for easy management and expansion of the software. MC++ may be obtained by contacting Dr. B. Chachuat[3].

[3]Email: b.chachuat@imperial.ac.uk. Centre for Process Systems Engineering, Department of Chemical Engineering, Imperial College London, South Kensington Campus, London SW7 2AZ, United Kingdom.

In parallel with the development of MC++, another GDO software, called GDOPT, was written in C++ by Ali Sahlodin. GDOPT still relies on MC++'s libraries to perform the lower-bounding problem, and employs the same third-party software as with MC++. Nonetheless, the two packages have different implementations in using the libraries and external software. Also, unlike MC++ that has a built-in class for SBB, GDOPT employs the third-party library libBandB by Singer [93].

Since MC++ is constantly being maintained and expanded, it is likely that it will be a future choice of solver for other researchers in the area of global optimization. To provide a basis of future comparisons, almost all numerical results in this thesis are obtained using MC++. The only exception is that the results of the previous algorithm (GDO with IBTM, see §5.1) are still obtained with GDOPT because this algorithm is not implemented in MC++. Despite the different implementations, the performance evaluation of the proposed and previous algorithms in this thesis are valid due to the following.

I. To a great extent, both packages share the same libraries (of MC++) and third-party software.

II. The same solver settings were used in both packages, where possible.

III. The performance of the two (proposed and previous) algorithms were also evaluated by GDOPT only, where similar performance results (not reported) to those given in the thesis were obtained.

In the following, the GDO solver of MC++ is briefly described. Note that the description is not meant to be a technical presentation of MC++'s object-oriented design, which can be found in its user's manual. Rather, it provides an overview of the major tasks involved in GDO and related MC++'s components. In addition to the descriptions below, the interface of the GDO solver is illustrated via an example in Appendix A.

Figure 5.2 illustrates the three major tasks of the GDO solver[4], namely SBB procedure, lower-bounding solver (LBS), and upper-bounding solver (UBS). The upper- and lower-bounding solvers involve a number of sub-tasks. Each of these tasks are handled in MC++ by its built-in classes or classes that act as an interface to third-party libraries. More information on each task are provided in the following subsections.

[4]The explanations assume a minimization problem.

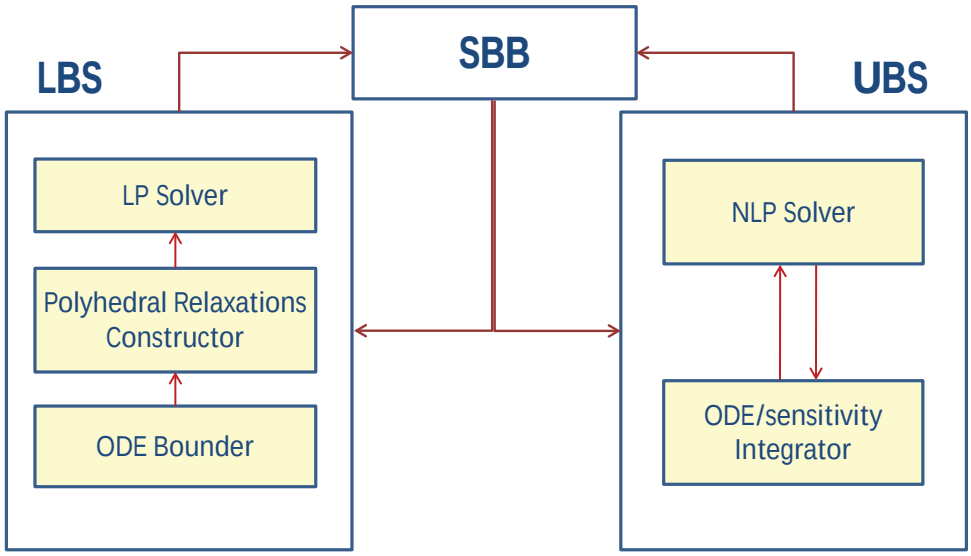

Figure 5.2: Illustration of the GDO solver. See text for information on the blocks' inputs/outputs.

5.4.1 SBB

The SBB procedure is implemented in a virtual base class named `SBB`. It contains a virtual method, called `subproblems`, that invokes the methods for LBS and UBS at each node. The virtual definition allows a derived class to have its own implementation of `subproblems`. For GDO, an implementation of `subproblems` as well as methods for LBS and UBS are contained in the derived class `DOSBB`. The inputs to `subproblems` are the variables bounds and initial guesses at the current node as provided by SBB. Also, its outputs are the solution points and values of lower/upper bounds on the current node given the solution has been successful. These outputs are provided by the methods for LBS/UBS. The return value of `subproblems` describes the solution status, and can be any of NORMAL (feasible solution found), INFEASIBILE (no feasible solution found), and FAILURE (solution failed due to other reasons). The value of the solution status is determined from the underlying methods for LBS/UBS. The outputs from `subproblems` along with its return value are sent back to SBB in order to decide on branching or deleting the current node.

5.4.2 Lower-bounding Solver

The LBS provides a lower bound on the current node by solving an underestimation of the dynamic optimization problem. The method for LBS has the same return values as `subproblems`, A NORMAL return means that a feasible lower bound has been obtained. An INFEASIBLE return indicates that the underestimation has no feasible solution, which in turn proves that the actual problem is also infeasible. Therefore, passing the solution status INFEASIBLE to `subproblems` and in turn to `SBB` will result in deleting the current node. On the other hand, a FAILURE return provides no useful information; thus, the node is retained by `SBB` for further branching. The LBS is comprised of three sub-tasks, as shown in Fig. 5.2, and described subsequently.

5.4.2.1 ODE Bounding and Relaxation

The first step in LBS is bounding the solutions of the embedded ODE system. This is handled by the class `ODEBND`. All the verified ODE techniques presented in Chapters 2, 3, and 4 can be performed by `ODEBND`. A uniform implementation of these techniques has been made possible by extensive use of class templates, function overloading, and operator overloading. These features allow reusability of methods and their arithmetic operations for any data-type for which they have been overloaded. In this work, these data-types correspond to interval bounds, McCormick relaxations, Taylor models, and McCormick-Taylor models that are defined by built-in MC++ libraries. An alternative to operator-overloading is source code transformation which, although more efficient, would be more difficult to program.

Note that the MC++ libraries for interval analysis and McCormick relaxations are not validated in the sense of round-off errors. Therefore, the results from `ODEBND` are guaranteed only against truncation errors in general. To obtain interval bounds accounting for round-off errors, third-party interval libraries are required. The validated interval packages PROFIL [43] and FILIB++ [46] are currently supported by MC++ (see §5.4.4), and can be used instead of the built-in interval library. It should be mentioned that, in practice, the difference between the results of a validated and a non-validated interval library may not be noticeable up to many decimal digits.

Two other external packages are used by `ODEBND`, namely the automatic differentiation package FADBAD++ [11] for computation of Taylor coefficients (see (2.26))

and their derivatives, and LAPACK/BLAS [3] for linear algebra including the QR transformation. See §5.4.4 for more information.

The ODE bounding methods in ODEBND have three return values as follows.

- NORMAL: when the integration is completed normally, and bounds are obtained up to the final integration time. In this case, the state bounds obtained by ODEBND are sent to Polyhedral Relaxations Constructor.
- FAILURE: when the integration is aborted prematurely because the state bounds have exploded before reaching the final time. In this case, the LBS is terminated with a FAILURE status.
- FATAL: when the integration is aborted prematurely because of other reasons such as numerical inconsistencies (e.g. bounds from two different methods do not intersect.) or incorrect setup of the ODE system by the user. Here again, the LBS is terminated with a FAILURE status.

5.4.2.2 Polyhedral Relaxations and LP Solver

The construction of polyhedral relaxations is implemented in the base class FPRELAX. Several classes are also responsible for specific tasks including definition of a new data-type for variables in the decomposed form of a factorable program; operations between these variables to generate linear relaxations; and linear outer-approximation of nonlinear relaxations. An important prerequisite to the automation of polyhedral relaxations is the ability to detect the structure of the factorable program as required for the decomposition step. This is facilitated by the class STRUCTURE of MC++. For a given factorable function and a set of participating variables, the sparsity pattern and linearity of the function with respect to the variables are determined by STRUCTURE.

The result from Polyhedral Relaxations Constructor is an LP, which is then solved to give a lower bound on the current node. Also, the domain reduction technique presented in §5.2.1 leads to an LP. The LP solutions can be handled by any conventional LP solver. MC++ currently supports CPLEX [2] and Gurobi [1], which are interfaced through the classes FPRCplex and FPRGurobi, respectively.

5.4.3 Upper-bounding Solver

The UBS provides an upper bound on the current node by performing a local optimization based on the sequential approach. The method for UBS has the same return type as `subproblems`. A NORMAL return means that a feasible upper bound has been obtained. A FAILURE return is used when the UBS is terminated due to unexpected reasons or because no feasible solution is found. A node with a FAILURE return from the UBS will be retained by `SBB` for further branching. It is important to emphasize that, unlike the LBS, the INFEASIBLE return value is not used in the UBS even if no feasible solution is found. This is to account for the fact that NLP solvers are prone to numerical issues and missing a feasible solution when it actually exists. By sending an INFEASIBLE flag, the SBB would delete the node even if it actually contained a feasible solution, possibly a global one. As shown in Fig. 5.2, the UBS performs an iterative procedure between an integrator and an NLP solver, which is described subsequently.

5.4.3.1 ODE/Sensitivity Integrator

This tasks involves integration of the embedded ODE system and their sensitivity equations. The variable values are provided from the NLP solver (see below). The outputs from the Integrator are the values of state variables and their sensitivity information that are sent back to the NLP solver. If the integration fails, the UBS is terminated with a FAILURE return. The ODE/sensitivity integration is performed by the external package CVODES [91], which is interfaced via the class `CVODES`. Also, the derivatives of the ODE right-hand sides are obtained using FADBAD++ [11].

5.4.3.2 NLP Solver

The local optimization is performed by the external NLP solver IPOPT, which is interfaced via the class `IPOPT`. Also, FADBAD++ is employed for convenient provision of exact gradient values to the NLP solver. As mentioned earlier, if the NLP solver finds a feasible solution, then the method for UBS returns a NORMAL status. Otherwise, a FAILURE status will be returned.

5.4.4 Third-party Software

All the third-party packages used in the GDO code are currently free of charge (at least) for academic use. These are listed below.

CVODES [91], is an ODE solver package with sensitivity analysis capabilities. Both forward and backward (adjoint) sensitivities [25, 22] are supported by CVODES.

IPOPT [102], is a C++ package for solving large-scale NLPs. It also has a C interface, and provides a wrapper for a Fortran interface. The C++ interface is used by MC++.

PROFIL/BIAS [43], is a C++ library for validated interval computations with the account of round-off errors.

FILIB++ [46], is another C++ library for validated interval computations. According to case studies conducted in this work, FILIB++ turns out to be (slightly) faster than PROFIL.

FADBAD++ [11], is a C++ implementation of automatic differentiation based on operator overloading. It supports both forward and backward modes of automatic differentiation. Also, high-order Taylor coefficients of functions and solutions of ODEs can be calculated easily using FADBAD++. The use of C++ templates enables differentiation of functions with arbitrary data-types such as those used in this work.

CPLEX [2], is an efficient software package for solving large-scale LPs.

Gurobi [1], is yet another powerful solver for large-scale LPs.

LAPACK/BLAS [3], is a collection of Fortran libraries for numerical linear algebra such as solving systems of linear equations and matrix factorizations.

5.5 Numerical Case Studies

Several case studies are conducted in this section to investigate the performance of the Taylor/McCormick-Taylor model based GDO algorithms presented in this chapter. Unless otherwise specified, the solver settings are as follows. Both relative and absolute tolerances for branch-and-bound are set to $\epsilon_{abs} = \epsilon_{rel} = 10^{-3}$. Taylor series expansions of order $k = 10$ and Taylor/McCormick-Taylor models of order $q = 4$ are used in the lower bounding problem. The settings of the domain reduction technique are chosen as $\beta = 20\%$ and Max_Repeat=4, respectively. For the proposed polyhedral relaxation-based approaches, a maximum of five cuts are considered in the sandwich algortihm (see §2.3.5). Although allowing more cuts can yield tighter lower bouns, it will increase the computational cost by increasing the number of constraints that must be hanlded by the LP solver. The computations are run on a PC with 4 GB RAM and Intel Core 2 Quad 2.4GHz CPU. However, all the CPU cores are not fully used in runtime due to the absence of a parallelization scheme.

5.5.1 Singular Control Problem

The first example is a nonlinear optimal control problem that has been studied by [52], and later by [34, 26, 96, 48]. It is formulated as

$$\begin{aligned}
\min_{p(t)\in P} \mathcal{J}(p(t)) &= \int_0^1 \left[x_1^2(t) + x_2^2(t) + 0.0005(x_2(t) + 16t - 8 - 0.1x_3(t)p^2(t))^2\right] dt \\
\dot{x}_1(t) &= x_2(t) \\
\dot{x}_2(t) &= -x_3(t)p(t) + 16t - 8 \\
\dot{x}_3(t) &= p(t) \\
\mathbf{x}(0) &= (0, -1, -\sqrt{5}) \\
p(t) &\in P = [-4, 10], \quad t \in [0, 1].
\end{aligned}$$

The above non-autonomous ODEs are written in the autonomous form (2.1) by introducing a new state variable $x_4(t) := t$, with the corresponding ODE $\dot{x}_4(t) = 1$, $x_4(0) = 0$. Moreover, in order to pose this problem in the form of (2.3), the integral objective function is replaced by an additional state variable x_5, with its

derivative equal to the integrand. These changes give the following form for the singular control problem.

$$\begin{aligned}
&\min_{p(t)\in P} \mathcal{J}(p(t)) = x_5(1) \qquad (5.6)\\
&\quad \dot{x}_1(t) = x_2(t)\\
&\quad \dot{x}_2(t) = -x_3(t)p(t) + 16x_4(t) - 8\\
&\quad \dot{x}_3(t) = p(t)\\
&\quad \dot{x}_4(t) = 1\\
&\quad \dot{x}_5(t) = x_1^2(t) + x_2^2(t) + 0.0005(x_2(t) + 16x_4(t) - 8 - 0.1x_3(t)p^2(t))^2\\
&\quad \mathbf{x}(0) = (0, -1, -\sqrt{5}, 0, 0)\\
&\quad p(t) \in P = [-4, 10], \quad t \in [0, 1].
\end{aligned}$$

The infinite dimensional control variable $p(t)$ is approximated by a piecewise constant parameterization into N equally spaced time stages. Since the original problem has one control variable, the parameterization will create $n_p = 1 \times N$ optimization variables.

As a first experiment, Problem (5.6) is solved using the IBTM, PRTM, and PRMCTM algorithms in the absence of domain reduction techniques. The results including the global solution and the number of nodes taken to solve the problem for up to 4 time stages are reported in Table 5.1. As expected, the global minimum value decreases with increasing the number of stages N. However, the improved solution comes at the price of a rapid increase in the branch-and-bound nodes for all the three algorithms. The node results show that the PRTM and PRMCTM algorithms have been able to considerably reduce the required nodes compared to the IBTM algorithm. This improvement is due to the use of polyhedral relaxations in Taylor/McCormick-Taylor models, which enables tighter over-approximations than simple interval bounds. Observe that the node counts for both PRTM and PRMCTM algorithms are coincidentally the same, suggesting that PRMCTM does not improve the bounds compared to PRTM in this problem. Note also that the node reduction by PRTM and PRMCTM becomes more prominent as the problem dimension n_p increases. It goes from about 10% for $n_p = 1$, to about 88% for $n_p = 4$.

The CPU times for the three algorithms are given in Table 5.2. As expected from the node results, the computational times are considerably reduced by the PRTM and PRMCTM algorithms. However, PRMCTM provides less improvement

	Solution		Nodes			Reduction vs. IBTM (%)	
n_p	$\mathcal{J}(\mathbf{p}^*)$	$\mathbf{p}^*$	IBTM	PRTM	PRMCTM	PRTM	PRMCTM
1	0.49654	(4.07089)	19	17	17	**10.5**	**10.5**
2	0.27711	(5.57479, -4)	193	75	75	**61.1**	**61.1**
3	0.14748	(8.00149, -1.94385, 6.04201)	4,947	1,081	1,081	**78.1**	**78.1**
4	0.12375	(9.78913, -1.19995, 1.25668, 6.35575)	88,605	11,917	11,917	**86.6**	**86.6**

Table 5.1: Global solution and node counts for Problem (5.6) without domain reduction techniques.

	Time (s)			Time reduction vs. IBTM (%)	
n_p	IBTM	PRTM	PRMCTM	PRTM	PRMCTM
1	0.17	0.12	0.17	**29.4**	**0.0**
2	3.74	1.41	1.85	**62.3**	**50.5**
3	143	35.8	45.9	**74.9**	**67.9**
4	4,166	774	987	**81**	**76**

Table 5.2: CPU time results (seconds) for Problem (5.6) without domain reduction techniques.

than PRTM, and particularly no improvement for $n_p = 1$. This is due to the extra overhead caused by the propagation of McCormick-Taylor models. Consequently, it takes more time to solve each node with the PRMCTM algorithm than with the PRTM algorithm. Therefore, branch-and-bound with PRMCTM is computationally more demanding than PRTM unless it results in a sufficient node reduction. On the other hand, the times needed to solve a node with IBTM and PRTM algorithms are comparable. This is because both use exactly the same method for the ODE integration, which is usually the dominant step in PRTM computationally compared to constructing and solving the polyhedral relaxations.

In the next experiment, Problem (5.6) is solved while employing applicable domain reduction techniques (see §5.1.1 and §5.2.1). The node and CPU time results are given in Tables 5.3 and 5.4, respectively. A comparison between these tables and Tables 5.1 and 5.2 shows that the domain reduction techniques effectively reduce the computational demand in all the three algorithms. This enables solving the problem for $n_p = 5$ in a reasonable time. Here again, the IBTM algorithm requires significantly more nodes than the other two. The node difference reaches an order of magnitude for $n_p = 5$. Within the polyhedral-based algorithms, PRMCTM converges in less nodes than PRTM for $n_p = 3, 4$. Nonetheless, the opposite result is observed for $n_p = 5$. The latter result may be unexpected at the first sight because the PRMCTM algorithm is guaranteed to produce tighter bounds than or at least as

tight bounds as the PRTM algorithm. However, it should be noted that bounding quality is not the only factor influencing the branch-and-bound performance. For example, the choice of the branching variable at each node can impact the sequence of the nodes being solved, and thus, the total number of nodes required. The selection of the branching variable at each node may not be the same for PRTM and PRMCTM because such a selection depends on the node solution that is generally different for the two algorithms. Consequently, it is not possible to make a prior judgment as to which of these algorithms would require fewer nodes to converge.

	Solution		Nodes			Reduction vs. IBTM (%)	
n_p	$\mathcal{J}(\mathbf{p}^*)$	$\mathbf{p}^*$	IBTM	PRTM	PRMCTM	PRTM	PRMCTM
1	0.4965	(4.0709)	5	3	3	**40.0**	**40.0**
2	0.2771	(5.5748, -4)	55	27	27	**50.9**	**50.9**
3	0.1475	(8.0015, -1.9439, 6.0420)	1,367	445	443	**67.4**	**67.6**
4	0.1238	(9.7891, -1.2000, 1.2567, 6.3558)	28,809	5,159	5,137	**82.1**	**82.2**
5	0.1236	(10, 1.4937, -0.8144, 3.3540, 6.1514)	504,827	54,617	55,107	**89.2**	**89.1**

Table 5.3: Global solution and node counts for Problem (5.6) with applicable domain reduction techniques.

Finally, it is interesting to compare the trends of the node and CPU time reductions in Tables 5.3 and 5.4. Despite the fact that the percentages of node reductions in Table 5.3 are monotonically increasing with n_p, the corresponding trend for CPU time reductions show a decrease for $n_p = 5$ (see Table 5.4). This decrease is due to increased cost of solving each node in the PRTM and PRMCTM cases. The increased cost is too high to be caused by the polyhedral relaxation procedure. Rather, it is attributed to repeated use of the domain reduction procedure (Step 3 in §5.3) at a large number of nodes. The computational effects of domain reduction techniques will be discussed further in §5.5.5.

	Time (s)			Time reduction vs. IBTM (%)	
n_p	IBTM	PRTM	PRMCTM	PRTM	PRMCTM
1	0.10	0.05	0.07	**50.0**	**30.0**
2	2.00	0.75	0.98	**62.5**	**51.0**
3	70.4	26.8	34.0	**62.0**	**51.8**
4	2,214	739	863	**67**	**61**
5	39,105	18,548	21,516	**53**	**45**

Table 5.4: CPU time results (seconds) for Problem (5.6) with applicable domain reduction techniques.

5.5.2 Van der Pol Problem

Consider the constrained van der Pol problem [18] given as

$$\begin{aligned}
\min_{p(t)\in P} \mathcal{J}(p(t)) &= \int_0^5 \left[x_1^2(t) + x_2^2(t) + p^2(t)\right] dt \\
&\dot{x}_1(t) = x_2(t) \\
&\dot{x}_2(t) = -x_1(t) + (1 - x_1^2(t))x_2(t) + p(t) \\
&x_1(5) - x_2(5) + 1 = 0 \\
&\mathbf{x}(0) = (1, 0) \\
&p(t) \in P = [-1, 1], \quad t \in [0, 5].
\end{aligned} \tag{5.7}$$

The problem is written in the form (2.3) by replacing the integral objective function with a new state variable as in (5.6). It is solved for up to 4 variables $n_p = N = 4$, using piecewise constant parameterization of $p(t)$, in the presence of applicable domain reduction techniques. The global solutions and node results are reported in Table 5.5. It is seen that a finer parameterization of p improves the global solution at the price of dramatical increase in the number of nodes. Furthermore, the polyhedral approaches PRTM and PRMCTM are able to significantly reduce the node counts. The reduction reaches 96% for $n_p = 4$, for which the IBTM needs over 1.3 million nodes to converge; that is, two orders of magnitude more than the polyhedral-based algorithms.

The CPU time results for Problem (5.7) are given in Table 5.6, where the PRTM algorithm is shown to decrease the CPU time up to an order of magnitude for $n_p = 4$.

	Solution		Nodes			Reduction vs. IBTM (%)	
n_p	$\mathcal{J}(\mathbf{p}^*)$	$\mathbf{p}^*$	IBTM	PRTM	PRMCTM	PRTM	PRMCTM
1	2.76	0.689	33	25	25	**24.2**	**24.2**
2	2.19	(7.20e-01, 4.92e-01)	337	305	305	**9.5**	**9.5**
3	2.06	(0.655, 0.678, 0.163)	10,891	3,669	3,657	**66.3**	**66.4**
4	1.89	(0.527, 0.915, 0.178, 0.226)	1,309,099	48,463	48,097	**96.3**	**96.3**

Table 5.5: Global solution and node counts for Problem (5.7) with applicable domain reduction techniques.

	Time (s)			Time reduction vs. IBTM (%)	
n_p	IBTM	PRTM	PRMCTM	PRTM	PRMCTM
1	2.96	1.43	2.48	**51.7**	**16.2**
2	23.3	17.4	32.9	**25.1**	**-41.5**
3	665	308	545	**54**	**18**
4	92,468	6,351	11,030	**93**	**88**

Table 5.6: CPU time results (seconds) for Problem (5.7) with applicable domain reduction techniques.

However, unlike the node results in Table 5.5, the CPU seconds for the PRTM and PRMCTM are not comparable, and in turn, PRMCTM offers less improvement than PRTM. In particular, notice the case $n_p = 2$ in which PRMCTM performs even worse than IBTM by 40 percent. As explained before, the overhead of repeatedly computing convex/concave bounds and subgradients in McCormick-Taylor models is responsible for the inferior performance.

5.5.3 Denbigh Problem

The Denbigh problem models a series reaction $A \xrightarrow{k_1} B \xrightarrow{k_2} C$, where the concentration of the middle component B is to be maximized [32],

$$\begin{aligned}
&\max_{\theta(t)\in\Theta} \mathcal{J}(\theta(t)) = C_B(10) \\
&\quad \dot{C}_A(t) = -k_1 C_A^2(t) \\
&\quad \dot{C}_B(t) = k_1 C_A^2(t) - k_2 C_B(t) \\
&\quad k_i = a_i exp(\frac{-b_i}{R\theta}), \quad i = 1,2 \\
&\quad (C_A(0), C_B(0)) = (1,0), \quad \theta(t) \in \Theta = [298, 423], \quad t \in [0,10],
\end{aligned}$$

with $\theta(t)$ being the reaction temperature and the constants $a_1 = 4000$, $a_2 = 620,000$, $\frac{b_1}{R} = 2500$, and $\frac{b_2}{R} = 5000$. To avoid the costly inverse operation on Taylor/McCormick-Taylor models [48], the optimization variable θ is replaced by a new variable p, with $p = \frac{298}{\theta}$. Replacing the notations C_A and C_B with x_1 and x_2, respectively, and writing the problem as a minimization one will give the following form.

$$\begin{aligned}
&\min_{p(t)\in P} \mathcal{J}(p(t)) = -x_2(t_f) \\
&\quad \dot{x}_1(t) = -k_1 x_1^2(t) \\
&\quad \dot{x}_2(t) = k_1 x_1^2(t) - k_2 x_2(t) \\
&\quad k_i = a_i exp(\frac{-b_i p}{R}), \quad i = 1,2 \\
&\quad \mathbf{x}(t_0) = (1,0), \quad t \in [t_0, t_\mathrm{f}] := [0,10], \quad p(t) \in P = [298/423, 1].
\end{aligned} \tag{5.8}$$

By using the same piecewise constant parameterization as before, the above problem is solved for up to $n_p = 6$ variables in the presence of applicable domain reduction techniques. Table 5.7 gives the global solutions and node results for the three algorithms. It is seen that there is no node reduction by PRTM and PRMCTM for $n_p = 1$, which implies that the bounds from IBTM are equally good due to the small size of the problem. Nonetheless, the polyhedral-based algorithms are shown to constantly

	Solution		Nodes			Reduction vs. IBTM (%)	
n_p	$\mathcal{J}(\mathbf{p}^*)$	$\mathbf{p}^*$	IBTM	PRTM	PRMCTM	PRTM	PRMCTM
1	-0.88112	0.9772	7	7	7	**0**	**0**
2	-0.88130	(0.9692, 0.9854)	31	27	27	**12.9**	**12.9**
3	-0.88139	(0.9633, 0.9828, 0.9863)	111	93	93	**16.2**	**16.2**
4	-0.88146	(0.95840, 0.9803, 0.98461, 0.98664)	411	303	295	**26.3**	**28.2**
5	-0.88151	(0.9542, 0.97808, 0.98306, 0.98545, 0.98687)	1,455	1,001	997	**31.2**	**31.5**
6	-0.88155	(0.9506, 0.9760, 0.9816, 0.9843, 0.9859, 0.9870)	5,521	3,421	3,409	**38.0**	**38.3**

Table 5.7: Global solution and node counts for Problem (5.8) with applicable domain reduction techniques.

	Time (s)			Time reduction vs. IBTM (%)	
n_p	IBTM	PRTM	PRMCTM	PRTM	PRMCTM
1	0.51	0.52	0.96	**-2.0**	**-88.2**
2	4.27	2.85	4.95	**33.3**	**-15.9**
3	16.9	15.5	25.1	**8.3**	**-48.5**
4	99.1	86.5	127	**12.7**	**-27.7**
5	518	484	727	**7**	**-40**
6	2,966	2,693	3,777	**9**	**-27**

Table 5.8: CPU time results (seconds) for Problem (5.8) with applicable domain reduction techniques.

improve the node results as n_p is increased from 2 to 6. Also, the PRMCTM algorithm converges in slightly fewer nodes than the PRTM algorithm for $n_p \geq 4$. This small improvement is attributed to McCormick-Taylor models that can contribute to tighter bounds in polyhedral relaxations (see §5.2). The convergence times tabulated in Table 5.8 show that the PRTM algorithm again outperforms the IBTM except for $n_p = 1$. This exception is explained by the extra overhead of polyhedral relaxations that cannot be offset without any node reduction. Fortunately, this overhead is minimal compared to the Taylor model ODE integration that are performed equally in both IBTM and PRTM algorithms. Therefore, slower convergence by the PRTM algorithm would not be significant in the worst case (only 2% in this case). On the other hand, the PRMCTM is seen to have the slowest convergence in spite of taking the least nodes. This clrealy shows that the trade-off between bouding quality and computational expense will determine the overall performenace in global optimization.

Another experiment is performed to study the impact of Taylor/McCormick-Taylor model order q on the GDO performance. Problem (5.8) with $n_p = 3$ is solved

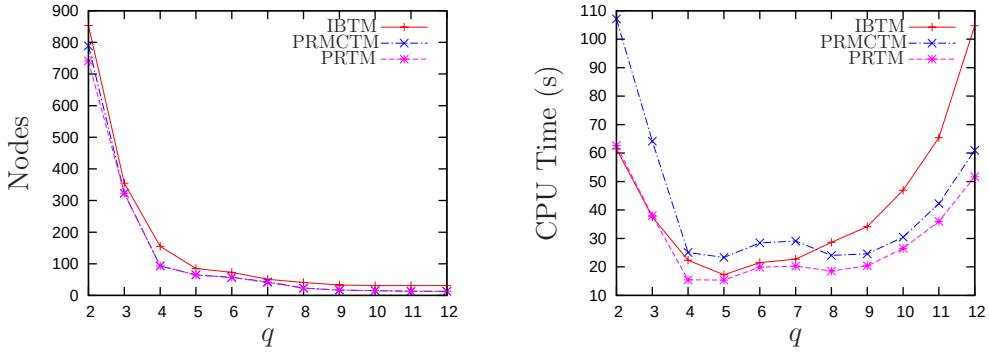

Figure 5.3: Effect of order of Taylor/McCormick-Taylor models in global optimization of Problem (5.8) using the three algorithms ($n_p = 3$).

using the three algorithms, and in each case, the solution is repeated by incrementing the order q from 2 to 12. The nodes and CPU times required for convergence in each case are plotted versus q in Fig. 5.3. It is seen that, in all the cases, the nodes required for convergence are decreased as q is increased. This is a direct result of tighter bounds produced by higher-order Taylor/McCormick-Taylor models. However, the effect of q on the nodes becomes less pronounced for higher q values. In particular, there is no node reduction by any of the algorithms by increasing q from 11 to 12. This can be an indication of sufficiently tight bounds in the $q = 11$ case, which leaves no improvement opportunity the case of $q = 12$.

On the other hand, a different trend is observed for the convergence times. By increasing the order q, the convergence times initially decrease, but start to increase after some q in each case. The initial decrease with higher orders q is due to tighter bounds, and hence, reduction in the number of nodes. The increasing trend, however, is explained by the computational cost of higher-order Taylor/McCormick-Taylor models, especially when the cost is no longer compensated for by significant node reduction. The trade-off between computing tight bounds and the effort needed to generate such tight bounds gives an optimum order q_{opt} for each case. Unfortunately, these optimum orders are not known a priori. Interestingly enough, the PRTM algorithm continues to preserve its superiority over the other two, even when they are run at their optimum orders q_{opt}.

5.5.4 Flow Control Problem

This example is adapted from [105], and considers product maximization in an isothermal semi-batch reactor. The reactions are given by

$$A + B \xrightarrow{k_1} C,$$
$$2B \xrightarrow{k_2} D.$$

The objective is to maximize the production of C, while keeping the terminal concentrations of B and D no higher than given thresholds. To achieve this, a dynamic optimization problem is formulated as follows:

$$\begin{aligned}
&\max_{p(t)} \mathcal{J}(p(t)) := C_C(t_\mathrm{f}) \qquad (5.9)\\
&\dot{C}_A(t) = -k_1 C_A(t) C_B(t) - \frac{p(t)}{V(t)} C_A(t)\\
&\dot{C}_B(t) = -k_1 C_A(t) C_B(t) - 2k_2 C_B(t)^2 + \frac{p(t)}{V(t)}(C_{B,in} - C_B(t))\\
&\dot{V}(t) = p(t)\\
&C_B(t_\mathrm{f}) \leq 0.02\\
&C_D(t_\mathrm{f}) \leq 0.005\\
&C_C(t_\mathrm{f}) = \frac{1}{V(t_\mathrm{f})}(C_{A0} V_0 - C_A(t_\mathrm{f}) V(t_\mathrm{f}))\\
&C_D(t_\mathrm{f}) = \frac{1}{2}(C_A(t_\mathrm{f}) + C_{B,in} - C_B(t_\mathrm{f})) - \frac{1}{2}(C_{A0} + C_{B,in} - C_{B0})\frac{V_0}{V(t_\mathrm{f})}\\
&(C_A(0), C_B(0), V(0)) = (C_{A0}, C_{B0}, V_0),\\
&t \in [t_0, t_\mathrm{f}] := [0, 50], \quad p(t) \in P = [0, 0.001],
\end{aligned}$$

where C_A, C_B, C_C and C_D are the concentrations of the species A, B, C, and D, respectively; V is the reaction volume; p is the volumetric flowrate of a pure B stream with the concentration of $C_{B,in} = 5$. The remaining model parameters are set as $k_1 = 0.053$, $k_2 = 0.128$, $C_{A0} = 0.72$, $C_{B0} = 0.05$, and $V_0 = 1$.

Problem (5.9) is solved using piecewise constant parameterization of $p(t)$, for up to $n_p = 11$ variables, in the presence of applicable domain reduction techniques. The

	Solution		Nodes			Reduction vs. IBTM (%)	
n_p	$\mathcal{J}(\mathbf{p}^*)$	$\mathbf{p}^*$	IBTM	PRTM	PRMCTM	PRTM	PRMCTM
1	4.857e-2	1.0147e-4	5	1	1	**80.0**	**80.0**
3	4.966e-2	(0, 2.4240e-4, 1.1934e-4)	215	3	3	**98.6**	**98.6**
5	4.967e-2	(0, 5.4307e-5, 2.3838e-4, 2.2272e-4, 8.7623e-5)	65,043	27	27	**100.0**	**100.0**
7	4.970e-2	(0, 0, 1.4419e-4 2.2187e-4, 1.8698e-4 2.3975e-4, 5.2425e-5)	$\gg$2.5e5	179	73	**100.0**	**100.0**
9	4.971e-2	(0, 0, 0, 2.3417e-4 1.9493e-4, 2.0789e-4 1.8731e-4, 2.4928e-4 1.3434e-5)	$\gg$2.0e6	1,007	209	**100.0**	**100.0**
10	4.971e-2	(0, 0, 0, 1.5142e-4 2.1783e-4, 1.9749e-4 2.0568e-4, 1.8623e-4 2.4921e-4, 0)	$\gg$1.7e6	2,427	281	-	-
11	4.971e-2	(0, 0, 0, 7.0290e-05 2.4064e-04, 1.9390e-04 2.0452e-04, 2.0232e-04 1.9420e-04 , 2.2273e-04, 0)	N/A	6,653	553	-	-

Table 5.9: Global solution and node counts for Problem (5.9) with applicable domain reduction techniques.

global solutions and node results are reported in Table 5.9. Interestingly, the PRTM and PRMCTM algorithms are shown to outperform IBTM very significantly; from 80% improvement for $n_p = 1$ to nearly 100% (after rounding the decimal digits) for $n_p = 5, 7, 9$. For $n_p \geq 7$, the IBTM algorithm did not even show a converging trend after so many nodes, and thus, was stopped by the user. For this reason, the node reduction percentages are not available for $n_p = 10$. Also, the IBTM algorithm was not attempted for $n_p = 11$ because, from its results for $n_p \geq 7$, it was not expected to converge in a reasonable time for this case. By comparing PRTM with PRMCTM, it is seen that the latter converges in much fewer nodes than the former for $n_p \geq 7$. This shows that McCormick-Taylor models in this example yield much tighter enclosures than Taylor models, unlike the previous examples where similar node results were observed in both variants.

The convergence times are given in Table 5.10. As expected from Table 5.9, the proposed algorithm converges remarkably faster than IBTM. The time reduction becomes more prominent as the problem size increases, and reaches nearly 100% (after rounding the demical digits) for $n_p = 7$. Within the two algorithm variants, PRMCTM is seen to be more expensive than PRTM for $n_p \leq 5$. This is explained by the observation from Table 5.9 that both variants required the same number of nodes for these cases. Thus, the extra cost of McCormick-Taylor models in PRMCTM is not compensated by any node reduction. However, a significant node reduction is

	Time (s)			Time reduction vs. IBTM (%)	
n_p	IBTM	PRTM	PRMCTM	PRTM	PRMCTM
1	0.10	0.04	0.07	**58.3**	**25.0**
3	7.08	0.32	0.46	**95.5**	**93.5**
5	4,886	9.64	10.6	**99.8**	**99.8**
7	$\gg$100,000	340	101	**100**	**100**
9	$\gg$1,500,000	9,655	1,397	-	-
10	$\gg$1,500,000	50,868	3,715	-	-
11	N/A	245,071	15,633	-	-

Table 5.10: CPU time results (seconds) for Problem (5.9) with applicable domain reduction techniques.

obtained by PRMCTM for $n_p \geq 7$, which makes it much cheaper than PRTM in terms of the overall SBB convergence.

5.5.5 Reversible Reactions Problem

This example is a parameter estimation problem for the following reversible series reaction system [36]:

$$A \underset{k_2}{\overset{k_1}{\rightleftharpoons}} B \underset{k_4}{\overset{k_3}{\rightleftharpoons}} C,$$

where k_i are the kinetic parameters. Assuming a first-order kinetic model, the following ODEs describe the dynamics of the system:

$$\begin{aligned}
&\dot{C}_A(t) = -k_1 C_A(t) + k_2 C_B(t) \\
&\dot{C}_B(t) = k_1 C_A(t) - (k_2 + k_3) C_B(t) + k_4 C_C(t) \\
&\dot{C}_C(t) = -k_4 C_C(t) + k_3 C_B(t) \\
&C_A(0) = C_{A,0},\ C_B(0) = C_{B,0},\ C_C(0) = C_{C,0},
\end{aligned}$$

where C_A, C_B, and C_C are the concentrations of the species A, B, and C, respectively. Specifying $t \in [0, 1]$, $k_1, k_2 \in [0, 10]$, and $k_3, k_4 \in [10, 50]$ [36], the following dynamic optimization problem is formulated to find optimal values of the kinetics parameters

t	0.05	0.1	0.15	0.2	0.25	0.3	0.35	0.4	0.45	0.5
x_1	0.8551	0.6748	0.5747	0.4867	0.4166	0.3608	0.3164	0.281	0.2529	0.2304
x_2	0.0937	0.1345	0.1654	0.1899	0.2094	0.2249	0.2373	0.2472	0.255	0.2613
x_3	0.0821	0.1802	0.2598	0.3233	0.3738	0.4141	0.4461	0.4717	0.492	0.5082
t	0.55	0.6	0.65	0.7	0.75	0.8	0.85	0.9	0.95	1
x_1	0.2126	0.1984	0.187	0.178	0.1709	0.1651	0.1606	0.157	0.1541	0.1518
x_2	0.2662	0.2702	0.2733	0.2759	0.2779	0.2794	0.2807	0.2817	0.2825	0.2832
x_3	0.521	0.5313	0.5395	0.546	0.5511	0.5553	0.5585	0.5612	0.5632	0.5649

Table 5.11: Concentration data used in the parameter estimation problem (5.10).

that minimize the error between the model predictions and reaction data.

$$
\begin{aligned}
\min_{\mathbf{p}} \mathcal{J}(\mathbf{p}) = & \sum_{i=1}^{3}\sum_{j=1}^{m}(x_{i,j}-\hat{x}_{i,j})^2 \\
& \dot{x}_1(t) = -p_1 x_1(t) + p_2 x_2(t) \\
& \dot{x}_2(t) = p_1 x_1(t) - (p_2+p_3) x_2(t) + p_4 x_3(t) \\
& \dot{x}_3(t) = -p_4 x_3(t) + p_3 x_2(t) \\
& t \in [0,1], \quad \mathbf{x}(0) = (1,0,0) \\
& p_1, p_2 \in [0,10], \quad p_3, p_4 \in [10,50],
\end{aligned}
\tag{5.10}
$$

where $p_i := k_i$; $(x_1, x_2, x_3) := (C_A, C_B, C_C)$; m is the number of time points at which the data are sampled; $\hat{x}_{i,j}$ are the reaction data. These data are given, with minor adaptation from the original one in [36], in Table 5.11.

It is interesting to note that the structure of (5.10) implies a lower bound of zero for the objective function. This information is exploited in the GDO solver by setting the lower bound of each node to $L^K \leftarrow \max(L^K, 0)$, which potentially allows for tighter bounds and faster convergence of the SBB procedure. The global optimization of (5.10) results in the minimum objective value $\mathcal{J}^* = 1.061523 \times 10^{-3}$, which is attained at $\mathbf{p}^* = (3.985491, 1.982305, 4.045275 \times 10^1, 2.023206e \times 10^1)$.

The nodes and CPU times required by the IBTM, PRTM, and PRMCTM algorithms are reported in Table 5.12. In terms of the nodes, there is a significant improvement by PRTM and PRMCTM compared to IBTM. The PRMCTM algorithm requires the fewest nodes, which is attributed to the ability of McCormick-

	Bounding method			Reduction vs. IBTM (%)	
	IBTM	PRTM	PRMCTM	PRTM	PRMCTM
Nodes	207	39	31	**81.2**	**85.0**
CPU Time	98.6	113	49.4	***-15.0***	**49.9**

Table 5.12: Nodes and CPU times for global solution of Problem (5.10) with default solver settings.

	Bounding method			Reduction vs. IBTM (%)	
	IBTM	PRTM	PRMCTM	PRTM	PRMCTM
Nodes	207	37	31	**82.1**	**85.0**
CPU Time	97.7	43.0	51.0	**56.0**	**47.8**

Table 5.13: Nodes and CPU times for global solution of Problem (5.10) with Max_Repeat=0.

Taylor models to produce tighter bounds. In terms of the CPU times, the PRMCTM algorithm results in about 50% reduction. However, it is surprising to see that the PRTM algorithm is inferior to IBTM in spite of the huge node reduction. This means that (at least) some nodes in the PRTM algorithm have been very costly to solve compared to the nodes in IBTM. This cost cannot be attributed to the overhead of polyhedral relaxations as it was found to be small from the previous examples. Rather, an inspection of the individual nodes showed that multiple execution of the domain reduction procedure in some nodes is responsible for the big computational burden. With the default solver settings, the domain reduction can be repeated up to Max_Repeat=4 times for each node. Recall from §5.3 that repeating the domain reduction involves re-integration of the ODEs, which is usually the dominant computation in the lower-bounding problem.

To verify the above arguments, Problem (5.10) was resolved with Max_Repeat=0, i.e. performing the domain reduction only once at each node. The corresponding node and CPU time results are given in Table 5.13. It seen that disabling multiple execution of the domain reduction has led to almost no change in the number of nodes required in this problem. On the other hand, the CPU time of the PRTM algorithm is now 56% less than that of the IBTM algorithm, which is quite expected given the node reduction. This experiment suggests that although the domain reduction technique is very helpful in reducing the computational time, it can adversely impact the efficiency if performed multiple times at each node.

5.5.6 Scaling Problem

Unlike the previous problems which had only few state and optimization variables, real-life problems may have many state and optimization variables. Therefore, it is important to know how the GDO would perform in problems with a larger number of optimization and state variables. To this end, a nonlinear dynamic problem is considered here, whose special design allows simultaneous increase in its number of state and optimization variables while retaining its structure. The problem, which is adapted from its linear dynamic version given by [94], is expressed as below.

$$\begin{aligned} \min_{\mathbf{p}\in\mathbf{P}} J(\mathbf{p}) &= \int_0^1 [(x_{n_p-1}^2(t) + x_{n_p}(t) - 11)^2 + (x_{n_p-1}(t) + x_{n_p}^2(t) - 7)^2]dt \qquad (5.11)\\ \dot{\mathbf{x}}(t) &= A\mathbf{x}(t) + B\mathbf{p}\\ \mathbf{x}(0) &= (0,\ldots,0)_{n_p} \quad t\in[0,1], \quad p_i \in P_i = [1,2] \quad \text{for} \quad i = 1,..,n_p \end{aligned}$$

where A is an $n_p \times n_p$ matrix given as

$$A_{n_p,n_p} = \begin{pmatrix} -1 & 0 & 0 & \cdots \\ 1 & -1 & 0 & \cdots \\ 0 & 1 & -1 & \cdots \\ \vdots & & & \ddots \end{pmatrix}$$

and B is a matrix of size $n_p \times n_p$ with its diagonal entries being $b_{i,i} = p_i$ and the off-diagonal entries being zero. Note that choosing n_p will determine both number of states and optimization variables. Here again, a new state variable will replace the integral objective function so it conforms with (2.3). In the first experiment, the problem is solved to global optimality for $n_p = 2,...,6$. The applicable domain reduction techniques are used to accelerate convergence. Figure 5.4 shows trends of the node counts and the CPU times versus n_p in semi-logarithmic scales. The left plot in Fig. 5.4 shows a sharp increase in the number of nodes required by the IBTM algorithm. This is compared to the PRTM and PRMCTM algorithms which follow a better trend, and result in a node reduction of two orders of magnitude for $n_p = 6$. Although it may not be clear from Fig. 5.4, both PRTM and PRMCTM algorithms converge in the same number of nodes, except for $n_p = 5$ and 6, where the PRMCTM

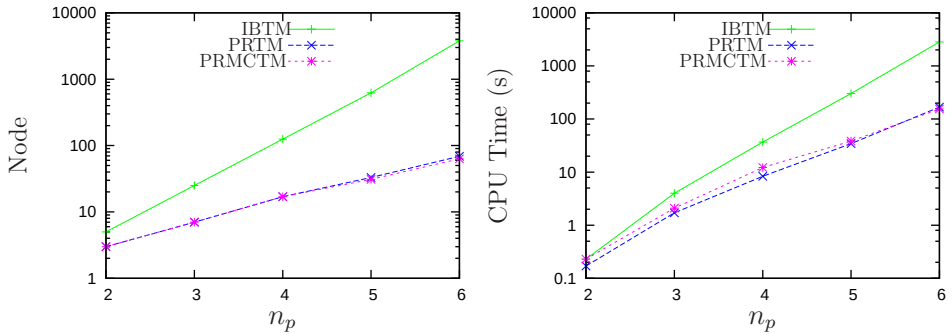

Figure 5.4: Scaling of nodes and CPU time in global optimization of Problem (5.11) with increasing the number of both states and optimization variables.

algorithm converges in respectively 6% and 9% fewer nodes than PRTM. Regarding the CPU times (the right plot of Fig. 5.4), similar trends as that of the nodes can be seen. That is, the polyhedral-based algorithms converge much faster than the IBTM algorithm, and again, this becomes more significant as the size of the problem grows. Furthermore, the PRMCTM algorithm is noticeably slower than the PRTM algorithm, a situation that was also seen in the previous examples.

Note that, although the state variables are increased simultaneously in this problem, the exponential trend in Fig. 5.4 is mainly due to the increased optimization variables. This exponential complexity is inherent to branch-and-bound procedures in general, as it was also noticed in the preceding examples with fixed number of state variables.

Due to the fast increase in the CPU time, it is not possible to solve Problem (5.11) for $n_p \geq 6$ in a reasonable time. However, an alternative experiment should be undertaken in order to predict the performance for larger problems. In particular, it is interesting to see how the underlying lower-bounding methods, which all build upon Taylor models, scale in computational complexity with the number of variables and ODEs. To enable such investigations for higher n_p, the next experiment considers solving Problem (5.11) at the root node only. Particularly, the computational time of the lower-bounding problem is considered as it has been the focus of this research. Moreover, the upper-bounding problem will remain unchanged regardless of the lower-bounding method used. The plots in Fig. 5.5 show the times taken by each algorithm to solve the lower-bounding problem at the root node conditions. The left plot shows the results for Taylor/McCormick-Taylor model order $q = 4$, while the right plot shows the same results for $q = 1$. The lower-bounding problem here

involves only a one-time ODE integration followed by the computation of a lower bound for the objective function. Therefore, it does not involve the domain reduction techniques that would impose extra arbitrary overheads on each algorithm, and adversely affect meaningful comparisons.

As seen in the left plot on Fig. 5.5, from $n_p = 2$ to $n_p = 16$, the computational demand of solving the lower-bounding problem increases by three orders of magnitude for the IBTM and PRTM algorithms and four orders of magnitude for the PRMCTM algorithm. The rapid increase is due to the complexity of multivariate Taylor polynomials propagated by Taylor/McCormick-Taylor models. Also, it is seen that the IBTM algorithm requires the least CPU time for all problem sizes. Regarding the PRTM algorithm, the time it requires is higher than the IBTM algorithm. However, the difference remains relatively small and unaffected by increasing n_p. In fact, both IBTM and PRTM algorithms scale at almost the same rate with the number of optimization variables. The reason is that both use the same Taylor model method for the ODE integration. On the other hand, the CPU times taken by the PRMCTM algorithm are the highest for all the problem sizes. Even worse, the PRMCTM's trend makes a sharp divergence from the other two for $n_p \geq 14$. The divergence is so large that for $n_p = 16$, the time required by PRMCTM is more than 20-fold the time required by the other two algorithms. As mentioned earlier, the computation of convex/concave bounds by McCormick-Taylor models leads to some extra overhead. Nonetheless, such overhead is only a fixed multiple factor of the time spent on computing interval bounds [82], and would not scale by increasing the optimization variables. Thus, the divergence of the PRMCTM trend with increased variables cannot be explained by the overhead of the McCormick relaxations themselves. Rather, the accompanying subgradient calculations are responsible for the dramatic increase in the computational burden. It should be noted that, however, this is not an inherent limitation of the PRMCTM algorithm, and can be alleviated by adopting alternative ways of subgradient calculations. More specifically, the current subgradient calculation implements the forward differentiation approach [39], which is known to be inefficient for problems with many variables [61]. For these problems, an efficient alternative is to use the reverse differentiation approach [38]. Otherwise, the PRMCTM algorithm will turn out to be prohibitively expensive for higher dimensional problems.

Similar trends as discussed above are seen in the right plot on Fig. 5.5. Compared to the results in the left plot, the use of lower-order Taylor/McCormick-Taylor models in the right plot has contributed to a slower growth in the computational times. Consequently, the scaling of n_p was continued to over 30 variables so that

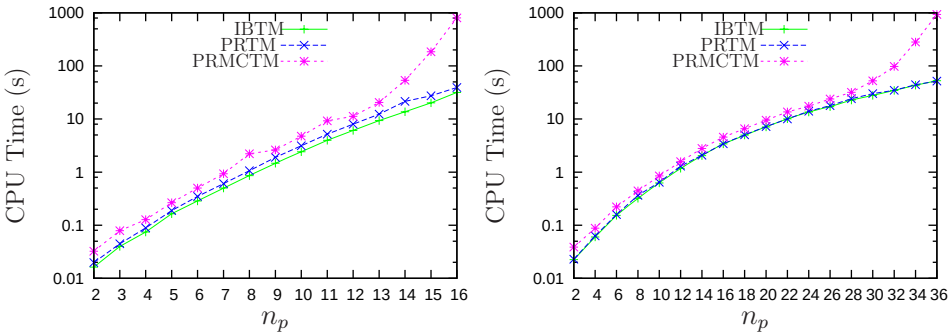

Figure 5.5: Scaling of time for solving the lower-bounding problem in Problem (5.11) using the Taylor model methods; left plot: $q = 4$; right plot: $q = 1$.

the deviation of the PRMCTM method can be noticed. Another point here is that the difference between the computational times of the IBTM and PRTM methods are less pronounced in the right plot than in the left plot. This is attributed to the fact that an LP obtained from the polyhedral relaxation of first-order Taylor/McCormick-Taylor models has fewer variables and constraints than one obtained from a fourth-order Taylor/McCormick-Taylor model. Therefore, solving the former imposes smaller overhead to the PRTM method.

5.6 Conclusions

In this chapter, a GDO algorithm was developed based on a new approach for bounding the underlying dynamic optimization model. This approach builds upon the (McCormick-)Taylor model ODE methods presented in Chapter 4. Instead of direct relaxation using McCormick's technique, it considers polyhedral relaxations of the (McCormick-)Taylor models. Polyhedral relaxations avoid the issues regarding the nonlinearity and nonsmoothness of McCormick relaxations, while enabling tight enclosures for the dynamic optimization model. Also, a special type of domain reduction technique is employed to accelerate the convergence of SBB.

The proposed algorithm has two variants, namely PRTM that stands for polyhedral relaxation of Taylor models, and PRMCTM that stands for polyhedral relaxation of McCormick-Taylor models. Due to the use of relaxations in their procedure, both variants are capable of reducing the SBB nodes significantly compared to the

algorithm in [47, 48] that relies solely on interval bounds from Taylor models (i.e. IBTM). In terms of solution times, solving each node with the PRTM variant has minimal computational overhead compared to the IBTM procedure because of cheap computations associated with polyhedral relaxations. As a result, the PRTM variant of the proposed GDO algorithm offers significant reduction in the GDO convergence time. Even in the worst case with no node reduction, the efficiency of the PRTM and IBTM should be comparable. On the other hand, solving each node with the PRMCTM variant is computationally expensive due to the propagation of convex/concave remainder bounds and their subgradients, particularly for large-scale problems. Also, the possible extra tightness by the PRMCTM variant is often negligible, and cannot offset the added computational burden. Consequently, the PRMCTM variant is usually less efficient than the PRTM variant, and can even be inferior to the IBTM algorithm. Based on these discussions, the proposed GDO algorithm with the PRTM variant seems to be promising for faster solution of dynamic optimization problems to global optimality compared to existing GDO algorithms.

Chapter6

Conclusions and Future Work

In this research, the global solution of dynamic optimization problems was addressed. The focus was on complete search methods (specifically SBB), where convergence to a certain neighborhood of the global solution value is possible in finite runtime, and can be proved mathematically. To decrease the computational complexities of SBB procedures, it is important to be able to provide tight bounds for the objective/-constraint functions participating in the optimization problem. This requirement becomes even more important in case of GDO due to the complexities introduced by the embedded nonlinear ODEs. The sequential approach of dynamic optimization was used because it preserves the original model, and in effect its global solution. However, computing valid, tight bounds for the ODE solutions in this approach has been a major challenge due to the absence of an explicit solution to nonlinear ODEs in general. As a result, practical GDO problems cannot be solved in a reasonable time with existing methods.

This research was aimed at addressing the above limitations of GDO. Specifically, the goal has been to improve computational efficiency of GDO by developing new algorithms for computing tight, yet relatively cheap, bounds on the dynamic optimization problem. The algorithms developed in this work build upon validated ODE methods that compute rigorous interval bounds on parametric ODE solutions by accounting for truncation errors. In addition, as the main contribution of this research, convex and concave relaxations of the dynamic model are enabled within the validated methods. This is to take various advantages of relaxations over simple

interval bounds including:

1. Tightness: Convex/concave relaxations are guaranteed not to be looser than their underlying interval bounds, and are usually tighter.

2. Convergence rate: Convex/concave relaxations typically converge faster to the actual solution set than interval bounds.

3. Domain reduction techniques: The use of convex/concave relaxations in SBB allows some types of domain reduction techniques that are not applicable with interval bounds. Note that the opposite is not true; that is, any domain reduction techniques designed for interval bounds remain applicable with convex/concave relaxations because interval bounds are also propagated with the relaxations.

All these benefits make the use of relaxations in SBB very appealing.

In Chapter 3 of the thesis, a validated ODE method based on interval analysis was extended to compute relaxations of state variables along with interval state bounds. Here, the computation of convex/concave bounds is handled by McCormick's relaxation technique. Due to the systematic treatment of the wrapping effect in the ODE method, tighter convex/concave bounds than those based on differential inequalities can be achieved. However, the convex/concave bounds are still inferior to interval bounds computed with a Taylor model ODE method. The reason is that this approach is still subject to large overestimation due to the dependency problem of interval analysis. On the other hand, Taylor models are able to deal with this problem quite effectively.

Based on the above observations, Chapter 4 of the thesis considered building convex/concave relaxations on Taylor models. This way, advantages of Taylor models and convex/concave bounds in producing tighter enclosures and having higher convergence rates are combined. Here, this combination is achieved in two forms, namely by McCormick relaxation of usual Taylor models and McCormick relaxation of McCormick-Taylor models that are introduced in this work. While the former still uses the same Taylor model ODE method presented in the literature, the latter required development of a new procedure to propagate McCormick-Taylor models of ODE solutions. It is shown through numerical studies that the obtained relaxations result in large improvement over interval bounds that have been considered previously. Nonetheless, while the latter form is potentially able to yield tighter enclosures than the former, it was observed that the extra tightness was often small

in practice. Therefore, the computational overhead of McCormick-Taylor models is not justified when the relaxation of usual Taylor models result in quite the same enclosure quality. The same arguments hold for the convergence orders of Taylor models and McCormick-Taylor models as investigated in §4.2.2: McCormick-Taylor models do not generally provide a higher convergence order than Taylor models.

Chapter 5 of the thesis consisted in developing a GDO algorithm by accommodating the (McCormick-)Taylor model based relaxations into SBB. Despite their ability to tightly enclose the solution of dynamic models, the nonlinearity and nonsmoothness of McCormick relaxations make their use in SBB difficult. Thus, polyhedral relaxations of (McCormick-)Taylor models were considered as an alternative to their McCormick relaxations. Although polyhedral relaxations may not be as tight as the nonlinear relaxations, they are much faster and more reliable to solve.

Similar to the Taylor model based relaxations in Chapter 4, the GDO algorithm has two variants, where in the first one, polyhedral relaxations of Taylor models, and in the second one, polyhedral relaxations of McCormick-Taylor models are considered. Compared to previous algorithms which did not incorporate relaxations into Taylor models, the developed GDO algorithm (both variants) is able to reduce the node-wise complexity of SBB for GDO. However, the two variants do not perform equally well in terms of the computational time. Here again, the variant with McCormick-Taylor models is found to be more costly, especially for problems with a large number of variables. Furthermore, it usually does not lead to major node reductions compared to the first variant, as expected from the results in Chapter 4. Consequently, it is usually slower than the first variant, and even than the previous algorithm in the worst case. On the other hand, the variant with Taylor models imposes a minimal computational overhead compared to the previous algorithm, and is seen to reduce the convergence times to a greater extent. It is interesting to mention that in the worst case with no node reduction, this variant is expected to converge in a comparable time compared to the previous algorithm in [48].

In spite of the promise shown by the proposed GDO algorithm (the first variant), it can be still very expensive to solve dynamic optimization problems with more than 10 decision variables. This greatly encourages further advancements in effective bounding of dynamic models in order to enable efficient global optimization of practical dynamic processes.

6.1 Future Work

In the following, a number of topics for future research are suggested.

Extension of the relaxation algorithm to DAEs. While this thesis addressed dynamic systems with embedded parametric ODEs only, there are many other systems that are characterized by differential-algebraic equations (DAEs). In general, a parametric DAE system takes the following form:

$$\begin{aligned} &\mathbf{f}(\dot{\mathbf{x}}(t), \mathbf{x}(t), \mathbf{p}) = 0 \\ &\quad \mathbf{x}(t) \in D \subseteq \mathbb{R}^{n_x}, \quad \mathbf{p} \in \mathbf{P} \subset \mathbb{R}^{n_p}, \quad \mathbf{f} : D \times D \times \mathbf{P} \to \mathbb{R}^{n_x}, \end{aligned} \tag{6.1}$$

with D an open connected set. Note that following the convention in this thesis, the DAE system is written in the autonomous form. In case (6.1) can be rewritten as (2.1), then it is called a system of *implicit* ODEs. A characteristic of DAE systems is that the state variables are subject to algebraic constraints. In a chemical process, these constraints can be hydraulic and thermodynamic relationships [84]. A DAE system is semi-explicit if the algebraic constraints appear explicitly as [33]:

$$\dot{\mathbf{x}}_d(t) = \mathbf{f}_d(\mathbf{x}_d(t), \mathbf{x}_a(t), \mathbf{p}) \tag{6.2}$$

$$0 = \mathbf{f}_a(\mathbf{x}_d(t), \mathbf{x}_a(t), \mathbf{p}), \tag{6.3}$$

where $\mathbf{x}_d$ are differential state variables as their derivatives occur in the system, and $\mathbf{x}_a$ are algebraic state variables as their derivatives do not appear in the system. The level of complexity of DAEs can be specified by their differential index, which is the number of differentiations of the DAE equations with respect to t that are needed to reveal the derivatives of all state variables [33].

Besides the complexities associated with the numerical solution of DAEs, there are difficulties bounding and relaxing their solutions for GDO. In particular, the presence of implicit functions in DAEs is a major barrier to propagating Taylor models and their relaxations to the state variables. To the best of the author's knowledge, there is currently no method for propagating Taylor models to a variable that occurs implicitly in an expression, i.e. a variable that does not have a factorable representation (e.g. the algebraic state variables in (6.3)). Therefore, the Taylor

model methods described in this thesis cannot be readily applied to DAE systems.

For the algebraic state variables, interval bounds can be computed using nonlinear interval techniques including the interval extension of the fixed-point iteration and interval Newton methods [62]. Building upon these methods and the generalization of McCormick relaxations in [90], Scott and Barton [86] proposed an iterative procedure for computing convex/concave relaxations of the algebraic state variables in semi-explicit index-one DAEs. Therefore, it is possible to extend the developments in this thesis to semi-explicit index-one DAE systems by incorporating Scott and Barton's procedure into the Taylor model methods. In doing so, the differential state variables expressed by $\mathbf{f}_d$ would be bounded by the Taylor model methods. However, the algebraic state variables would be bounded using interval analysis-based methods. As a clear disadvantage of this approach, the lack a Taylor model representation for the algebraic state variables will lead to large overestimation during integration of the DAEs and subsequent solution of the relaxed optimization problem. Thus, some modifications to this approach or more effective approaches for extending the Taylor model relaxations to DAEs are required.

Exploiting model structure for better performance. Apart from some continuity and differentiability assumptions, this work imposes no restrictions on the dynamic behavior and structure of the ODEs involved in the dynamic optimization problem. As a result, a wide range of nonlinear parametric ODEs can be tackled within the proposed relaxation algorithm. Nevertheless, a downside here is that the algorithm disregards model-specific information that may be extractable from the structure of the ODEs, and could help obtain refined enclosures for the dynamic model. Consequently, the computed relaxations can be overly conservative for some classes of problems. Therefore, a great potential lies in enabling the algorithm to exploit the structure and solution of the dynamic model in a way that yields tighter relaxations. In this regard, related ideas from other researchers could be incorporated into the algorithm. For example, Singer and Barton [94] presented a technique that computes exact bounds for systems with linear dynamics. Also, Ramdani et al. [72] proposed a hybrid scheme for bounding ODEs, which considers a multiple mode dynamics and selects a bounding strategy tailored to each mode during the integration.

Relaxation of multivariate Taylor polynomials. The polyhedral approximation of Taylor/McCormick-Taylor models was shown to significantly improve the

enclosure quality compared to bounding them. However, the over-approximation by polyhedral relaxations can be significant for high-order Taylor polynomials having many variables. The over-approximation is partly attributed to the fact that polyhedral relaxations are meant to tackle general factorable programs, and may ignore some useful properties of multivariate polynomials during their relaxation. This motivates further research into new relaxation procedures tailored to multivariate Taylor polynomials.

Combination of differential inequalities with Taylor model relaxations. Concurrent with the developments on applications of Taylor models and their relaxations for GDO, significant research has been carried out on the use of differential inequalities for GDO (see §1.1). Therefore, it is worthwhile to consider possible synergy of these two approaches for producing tighter and more efficient enclosures. Particularly, Chachuat and Villanueva [27] recently proposed a combination of Taylor models with differential inequalities for bounding solutions of parametric ODEs. In this combination, differential inequalities are used to compute interval remainder bounds in the Taylor model integration. Other combination forms and their extension to convex/concave bounds appear to be interesting topics for future research.

References

[1] Gurobi Optimizer. http://www.gurobi.com/. 122, 124

[2] IBM ILOG CPLEX Optimizer. http://www-01.ibm.com/software/integration/optimization/cplex-optimizer/. 122, 124

[3] Linear Algebra PACKage. http://www.netlib.org/lapack. 122, 124

[4] T. Achterberg, T. Koch, and A. Martin. Branching rules revisited. *OPERATIONS RESEARCH LETTERS*, 33:42–54, 2004. 53, 56

[5] C. Adjiman, I. Androulakis, and A. Floudas. A global optimization method, αBB, for general twice-differentiable constrained NLPs-II. implementation and computational results. *Computers and Chemical Engineering*, 22(9):1159 – 79, 1998. 4

[6] C. S. Adjiman, S. Dallwig, C. A. Floudas, and A. Neumaier. A global optimization method, αBB, for general twice-differentiable constrained NLPs - I. Theoretical advances. *Computers and Chemical Engineering*, 22(9):1137–1158, 1998. 4, 21

[7] F. AI-Khayyal and J. Falk. Jointly constrained biconvex programming. *Mathematics of Operations Research*, 8(2):273 – 286, 1983. 31

[8] G. Alefeld and G. Mayer. Interval analysis: Theory and applications. *Journal of Computational and Applied Mathematics*, 121:421–464, 2000. 15, 29

[9] E. Balas and P. Toth. Branch and bound methods for the traveling salesman problem. Technical Report MSRR 488, Carnegie-Mellon University, Pittsburgh, Pennsylvania, March 1983. 55, 56

[10] J. R. Banga and W. D. Seider. Global optimization of chemical processes using stochastic algorithms. In C. Floudas and P. Pardalos, editors, *State of the Art in Global Optimization: Computational Methods and Applications*, pages 563–583. Kluwer Academic Pub, 1996. 2

[11] C. Bendtsen and O. Stauning. FADBAD++. http://www.fadbad.com/fadbad.html. 121, 123, 124

[12] M. Berz. From taylor series to taylor models. In *Nonlinear Problems in Accelerator Physics*, pages 1–27. American Institute of Physics CP405, 1997. 16, 17

[13] M. Berz and K. Makino. Verified integration of ODEs and flows using differential algebraic methods on high-order Taylor series. *Reliable Computing*, 4:361–369, 1998. 5, 39

[14] M. Berz and K. Makino. Performance of taylor model methods for validated integration of ODEs. *Lecture Notes in Computer Science*, 3732:65–74, 2006. 47

[15] B. Bhattacharjee, P. Lemonidis, W. H. Green, Jr., and P. I. Barton. Global solution of semi-infinite programs. *Mathematical Programming*, 103(2):283–307, June 2005. 26

[16] L. Biegler. An overview of simultaneous strategies for dynamic optimization. *Chemical Engineering and Processing*, 46(11):1043–1053, 2007. 12, 13

[17] L. T. Biegler, A. M. Cervantes, and A. Wchter. Advances in simultaneous strategies for dynamic process optimization. *Chemical Engineering Science*, 57(4):575 – 593, 2002. 12

[18] H. Bock and K. Plitt. A multiple shooting algorithm for direct solution of optimal control problems. In *Bridge Between Control Science and Technology. Proceedings of the Ninth Triennial World Congress of IFAC*, pages 1603 – 8, Oxford, UK, 1985. 13, 129

[19] A. Bompadre and A. Mitsos. Convergence rate of McCormick relaxations. *Journal of Global Optimization*, 52(1):1 – 28, 2012. 5, 26, 28, 29

[20] A. Bompadre, A. Mitsos, and B. Chachuat. Convergence Analysis of Taylor Models and McCormick-Taylor Models. In revision, 2012. 26, 27, 28, 29, 30, 100, 102, 104, 106

[21] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, New York, NY, USA, 2004. 19, 51

[22] A. Cervantes and L. T. Biegler. Optimization strategies for dynamic systems. In *In C. Floudas, P. Pardalos (Eds), Encyclopedia of Optimization*. Kluwer Academic Publishers, 1999. 13, 124

[23] A. Cervantes, S. Tonelli, A. Brandolin, J. Bandoni, and L. Biegler. Large-scale dynamic optimization for grade transitions in a low density polyethylene plant. *Computers and Chemical Engineering*, 26(2):227 – 237, 2002. 2

[24] B. Chachuat. http://www3.imperial.ac.uk/people/b.chachuat/research. 118

[25] B. Chachuat. *Nonlinear and Dynamic Optimization: From Theory to Practice - IC-32: Spring Term 2009*. Polycopiés de l'EPFL. EPFL, 2009. 11, 12, 13, 20, 51, 124

[26] B. Chachuat and M. A. Latifi. A new approach in deterministic global optimization of problems with ordinary differential equations. In C. A. Floudas and P. M. Pardalos, editors, *Frontiers in Global Optimization*, volume 74 of *Nonconvex Optimization and Its Applications*, pages 83–108. Kluwer Academic Publishers, 2003. 5, 71, 80, 125

[27] B. Chachuat and M. Villanueva. Bounding the solutions of parametric odes: When taylor models meet differential inequalities. In I. D. L. Bogle and M. Fairweather, editors, *22nd European Symposium on Computer Aided Process Engineering*, volume 30 of *Computer Aided Chemical Engineering*, pages 1307 – 1311. Elsevier, 2012. 149

[28] T. W. Chen and V. S. Vassiliadis. Inequality path constraints in optimal control: a finite iteration -convergent scheme based on pointwise discretization. *Journal of Process Control*, 15(3):353 – 362, 2005. 10

[29] L. Ching-Jong. A new node selection strategy in the branch-and-bound procedure. *Computers and Operations Research*, 21(10):1095 – 1101, 1994. 53

[30] G. F. Corliss and R. Rihm. Validating an a priori enclosure using high-order Taylor series. In G. Alefeld, A. Frommer, and B. Lang, editors, *Scientific Computing and Validated Numerics: Proceedings of the International Symposium on Scientific Computing, Computer Arithmetic and Validated Numerics - SCAN'95*, pages 228–238, Berlin, Germany, 1996. Akademie Verlag. 39

[31] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. The MIT Press, USA, 2nd edition, 2001. 55

[32] S. Dadebo and K. McAuley. Dynamic optimization of constrained chemical engineering problems using dynamic programming. *Computers and Chemical Engineering*, 19(5):513–525, 1995. 131

[33] B. K. E., C. S. L., and P. L. R. *Numerical Solution of Initial-Value Problems in Differential-Algebraic Equations*. Society for Industrial and Applied Mathematics, Philadelphia, USA, 1996. 147

[34] W. R. Esposito and C. A. Floudas. Deterministic global optimization in nonlinear optimal control problems. *Journal of Global Optimization*, 17:96–126, 2000. 125

[35] W. R. Esposito and C. A. Floudas. Global optimization for the parameter estimation of differential- algebraic systems. *Industrial and Engineering Chemistry Research*, 39(5):1291 – 1310, 2000. 4

[36] C. A. Floudas, P. M. Pardalos, C. S. Adjiman, W. R. Esposito, Z. H. Gumus, S. T. Harding, J. L. Klepeis, C. A. Meyer, and C. A. Schweiger. *Handbook of Test Problems in Local and Global Optimization*. Kluwer Academic Publishers, Dordrecht, The Netherlands, 1999. 98, 136, 137

[37] D. Goldberg. *Genetic algorithms in search, optimization, and machine learning*. Artificial Intelligence. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1989. 50

[38] A. Griewank. On automatic differentiation. In M. Iri and K. Tanabe, editors, *Mathematical Programming: Recent Developments and Applications*, pages 83–108, Dordrecht, 1989. Kluwer Academic Publishers. 24, 100, 141

[39] A. Griewank. *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation*. Frontiers in Applied Mathematics, SIAM, Philadelphia, USA, 2000. 100, 141

[40] J.-B. Hiriart-Urruty and C. Lemarechal. *Fundamentals of Convex Analysis*. Springer-Verlag, Berlin, Germany, 2001. 23

[41] K. R. Jackson and N. S. Nedialkov. Some recent advances in validated methods for IVPs for ODEs. *Applied Numerical Mathematics*, 42:269–284, 2002. 42

[42] R. B. Kearfott. Discussion and empirical comparisons of linear relaxations and alternate techniques in validated deterministic global optimization. *Optimization Methods and Software*, 21:715–731, 2006. 6, 107

[43] O. Knüppel. PROFIL/BIAS V 2.0. http://www.ti3.tu-harburg.de/Software/PROFILEnglisch.html, 1999. 121, 124

[44] R. E. Korf. Linear-space best-first search. *Artificial Intelligence*, 62(1):41 – 78, 1993. 55, 56

[45] D. B. Leineweber, I. Bauer, H. G. Bock, and J. P. Schlöder. An efficient multiple shooting based reduced SQP strategy for large-scale dynamic process optimization. Part 1: theoretical aspects. *Computers and Chemical Engineering*, 27(2):157 – 166, 2003. 13

[46] M. Lerch, G. Tischler, J. W. V. Gudenberg, W. Hofschuster, and W. Krämer. FILIB++, a fast interval library supporting containment computations. *ACM Transactions on Mathematical Software*, 32(2):299–324, June 2006. 121, 124

[47] Y. Lin and M. A. Stadtherr. Deterministic global optimization for parameter estimation of dynamic systems. *Industrial and Engineering Chemistry Research*, 45:8438–9448, 2006. 5, 6, 80, 108, 109, 110, 143

[48] Y. Lin and M. A. Stadtherr. Deterministic global optimization of nonlinear dynamic systems. *AIChE Journal*, 53(4):866–875, 2007. 5, 6, 80, 86, 108, 109, 110, 115, 125, 131, 143, 146

[49] Y. Lin and M. A. Stadtherr. Validated solutions of initial value problems for parametric ODEs. *Applied Numerical Mathematics*, 57(10):1145–1162, 2007. ix, 5, 16, 18, 39, 44, 45, 46, 47, 70, 71, 72, 73, 76, 78, 84, 98, 105

[50] J. T. Linderoth and M. W. P. Savelsbergh. A computational study of search strategies for mixed integer programming. *INFORMS Journal on Computing*, 11(2):173 – 187, 1999. 53, 56

[51] R. J. Lohner. Computation of guaranteed enclosures for the solutions of ordinary initial and boundary value problems. In J. R. Cash and I. Gladwell, editors, *Computational Ordinary Differential Equations*, volume 1, pages 425–436. Clarendon Press, 1992. 5, 39, 42

[52] R. Luus. Optimal control by dynamic programming using systematic reduction in grid size. *International Journal of Control*, 51(5):995 – 1013, 1990. 125

[53] R. Luus and D. E. Cormack. Multiplicity of solutions resulting from the use of variational methods in optimal control problems. *Canadian Journal of Chemical Engineering*, 50:309–311, 1972. 2, 4

[54] R. Luus, J. Dittrich, and F. J. Keil. Multiplicity of solutions in the optimization of a bifunctional catalyst blend in a tubular reactor. *The Canadian Journal of Chemical Engineering*, 70(4):780–785, 1992. 2

[55] K. Makino and M. Berz. Efficient control of the dependency problem based on Taylor model methods. *Reliable Computing*, 5(1):3–12, 1999. 16

[56] K. Makino and M. Berz. Taylor models and other validated functional methods. *International Journal of Pure and Applied Mathematics*, 4:379–456, 2003. 6, 16, 17, 18

[57] K. Makino and M. Berz. Verified global optimization with Taylor model based range bounders. *WSEAS Transactions on Computers*, 4(11):1611 – 1618, 2005. 18

[58] G. P. McCormick. Computability of global solutions to factorable nonconvex programs: Part I – Convex underestimating problems. *Mathematical Programming*, 10:147–175, 1976. 5, 14, 21, 26, 32

[59] F. Messine. Deterministic global optimization using interval constraint propagation techniques. *RAIRO - Operations Research*, 38(4):277 – 293, 2004. 54

[60] A. Mitsos, B. Chachuat, and P. I. Barton. Towards global bilevel dynamic optimization. *Journal of Global Optimization*, 45(1):63 – 93, 2009. 3

[61] A. Mitsos, B. Chachuat, and P. L. Barton. McCormick-based relaxations of algorithms. *SIAM Journal on Optimization*, 20(2):573–601, 2009. 23, 100, 141

[62] R. E. Moore, R. B. Kearfott, and M. J. Cloud. *Introduction to Interval Analysis*. SIAM, Philadelphia, PA, 2009. 5, 14, 15, 24, 26, 148

[63] N. Nedialkov. Interval tools for ODEs and DAEs. In *Proc. 12th GAMM-IMACS Int Symp SCAN 2006*. IEEE Computer Society, 2006. ix, 70, 72, 73

[64] N. S. Nedialkov. *Computing Rigorous Bounds on the Solution of an Initial Value Problem for an Ordinary Differential Equation*. PhD thesis, University of Toronto, Toronto, Canada, 1999. 70, 101

[65] N. S. Nedialkov, K. R. Jackson, and G. F. Corliss. Validated solution of initial value problems for ordinary differential equations. *Applied Mathematics and Computation*, 105:21–68, 1999. 5, 6, 39

[66] N. S. Nedialkov, K. R. Jackson, and J. D. Pryce. An effective high-order interval method for validating existence and uniqueness of the solution of an IVP for an ODE. *Reliable Computing*, 7:449–465, 2001. 40, 70, 72, 88, 90

[67] M. Neher. From interval analysis to Taylor models - an overview. In *IMACS*, Paris, France, 2005. 15

[68] A. Neumaier. Taylor forms - use and limits. *Reliable Computing*, 9(1):43–79, 2002. 18

[69] A. Neumaier. Complete search in continuous global optimization and constraint satisfaction. *Acta Numerica*, 13:271 – 369, 2004. 3, 50, 51, 52, 54

[70] I. Papamichail and C. S. Adjiman. A rigorous global optimization algorithm for problems with ordinary differential equations. *Journal of Global Optimization*, 24:1–33, 2002. 5, 71, 98

[71] L. S. Pontryagin, V. G. Boltyanskii, R. V. Gamkrelidze, and E. F. Mishchenko. *The mathematical theory of optimal processes*. Pergamon Press, New York, 1964. 11

[72] N. Ramdani, N. Meslem, and Y. Candau. A hybrid bounding method for computing an over-approximation for the reachable set of uncertain nonlinear systems. *IEEE Transactions on Automatic Control*, 54(10):2352 – 2364, 2009. 148

[73] R. Rardin. *Optimization in Operations Research*. Prentice Hall, New Jersey, USA, 1998. 49

[74] R. Rihm. On a class of enclosure methods for initial value problems. *Computing*, 53:369–377, 1994. 42

[75] H. Ryoo and N. Sahinidis. Global optimization of nonconvex NLPs and MINLPs with applications in process design. *Computers and Chemical Engineering*, 19(5):551 – 551, 1995. 53, 54, 55

[76] H. S. Ryoo and N. V. Sahinidis. A branch-and-reduce approach to global optimization. *Journal of Global Optimization*, 8:107–138, 1996. 53

[77] N. V. Sahinidis. Baron: A general purpose global optimization software package. *Journal of Global Optimization*, 8(2):201 – 205, 1996. 3

[78] N. V. Sahinidis. BARON Branch And Reduce Optimization Navigator, User's Manual, Version 4. http://archimedes.cheme.cmu.edu/baron/manuse.pdf, 2000. 3

[79] N. V. Sahinidis, C. Bleik, C. Jermann, and A. Neumaier. Global optimization and constraint satisfaction: The branch-and-reduce approach. *Lecture Notes in Computer Science*, 2861:1–16, 2003. 53

[80] A. M. Sahlodin and B. Chachuat. A discretize-then-relax approach for state relaxations in global dynamic optimization. In *Computer Aided Chemical Engineering, Proceedings of 20th European Symposium on Computer Aided Process Engineering*, volume 28, pages 427–432, 2010. 86

[81] A. M. Sahlodin and B. Chachuat. Convex/concave relaxations of parametric ODEs using Taylor models. *Computers and Chemical Engineering*, 35(5):844–857, 2011. 81

[82] A. M. Sahlodin and B. Chachuat. Discretize-then-relax approach for convex/-concave relaxations of the solutions of parametric ODEs. *Applied Numerical Mathematics*, 61(7):803–820, 2011. 5, 86, 93, 141

[83] A. M. Sahlodin and B. Chachuat. Tight convex and concave relaxations via Taylor models for global dynamic optimization. In M. G. E.N. Pistikopoulos and A. Kokossis, editors, *21st European Symposium on Computer Aided Process Engineering*, volume 29 of *Computer Aided Chemical Engineering*, pages 537 – 541. Elsevier, 2011. 81

[84] V. Sakizlis, J. D. Perkins, and E. N. Pistikopoulos. Recent advances in optimization-based simultaneous process and control design. *Computers and Chemical Engineering*, 28(10):2069 – 2086, 2004. 147

[85] A. Schbel and D. Scholz. The theoretical and empirical rate of convergence for geometric branch-and-bound methods. *Journal of Global Optimization*, 48:473–495, 2010. 25

[86] J. Scott and P. Barton. Convex and concave relaxations for the parametric solutions of semi-explicit index-one differential-algebraic equations. *Journal of Optimization Theory and Applications*, In press. DOI: 10.1007/s10957-012-0149-8. 148

[87] J. Scott and P. Barton. Improved relaxations for the parametric solutions of ODEs using differential inequalities. *Journal of Global Optimization*, In press. DOI: 10.1007/s10898-012-9909-0. 5

[88] J. K. Scott and P. I. Barton. Tight, efficient bounds on the solutions of chemical kinetics models. *Computers and Chemical Engineering*, 34(5):717 – 731, 2010. 5

[89] J. K. Scott, B. Chachuat, and P. I. Barton. Nonlinear convex and concave relaxations for the solutions of parametric ODEs. *Optimal Control Applications and Methods*, In press. DOI: 10.1002/oca.2014. 5

[90] J. K. Scott, M. D. Stuber, and P. I. Barton. Generalized McCormick relaxations. *Journal of Global Optimization*, 51(4):569 – 606, 2011. 21, 22, 23, 65, 84, 148

[91] R. Serban and A. C. Hindmash. CVODES, the sensitivity-enabled ODE solver in SUNDIALS. In *Proceedings of the ASME International Design Engineering Technical Conferences and Computers and Information in Engineering Conference - DETC2005*, volume 6 A, pages 257 – 269, Long Beach, CA, United states, 2005. 123, 124

[92] J. P. Shectman and N. V. Sahinidis. Finite algorithm for global minimization of separable concave programs. *Journal of Global Optimization*, 12(1):1 – 36, 1998. 57

[93] A. Singer. *Global Dynamic Optimization*. PhD thesis, Massachusetts Institute of Technology, Boston, USA, 2004. 57, 119

[94] A. Singer and P. Barton. Global solution of optimization problems with parameter-embedded linear dynamic systems. *Journal of Optimization Theory and Applications*, 121(3):613–646, 2004. 139, 148

[95] A. B. Singer and P. I. Barton. Bounding the solutions of parameter dependent nonlinear ordinary differential equations. *SIAM Journal on Scientific Computing*, 27(6):2167–2182, 2006. 5, 76, 95

[96] A. B. Singer and P. I. Barton. Global optimization with nonlinear ordinary differential equations. *Journal of Global Optimization*, 34(2):159–190, 2006. 5, 80, 125

[97] E. Smith and C. Pantelides. A symbolic reformulation/spatial branch-and-bound algorithm for the global optimisation of nonconvex MINLPs. *Computers and Chemical Engineering*, 23(4-5):457 – 478, 1999. 21, 31, 108

[98] M. Tawarmalani and N. V. Sahinidis. *Convexification and Global Optimization in Continuous and Mixed-Integer Nonlinear Programming: Theory, Algorithms, Software, and Applications*. Nonconvex Optimization and Its Applications. Kluwer Academic Publishers, Boston, MA, 2002. 6, 107

[99] M. Tawarmalani and N. V. Sahinidis. Global optimization of mixed-integer nonlinear programs: A theoretical and computational study. *Mathematical Programming*, 99(3):563 – 591, 2004. 32, 33, 54, 108

[100] K. L. Teo, C. J. Goh, and K. H. Wong. *A Unified Computational Approach to Optimal Control Problems*. Longman Scientific & Technical, New York, 1991. 10, 11

[101] V. S. Vassiliadis, R. W. H. Sargent, and C. C. Pantelides. Solution of a class of multistage dynamic optimization problems. 2. Problems with path constraints. *Industrial and Engineering Chemistry Research*, 33(9):2123–2133, 1994. 10

[102] A. Wachter and L. T. Biegler. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical Programming*, 106(1):25 – 57, 2006. 124

[103] W. Walter. *Differential and Integral Inequalities*. Springer-Verlag, Berlin, Germany, 1970. 5, 95

[104] W. Zhang and R. E. Korf. Depth-first vs. best-first search: new results. In *Proceedings of the National Conference on Artificial Intelligence*, pages 769 – 775, Washington, DC, USA, 1993. 55, 56

[105] Y. Zhao and M. A. Stadtherr. Rigorous global optimization for dynamic systems subject to inequality path constraints. *Industrial and Engineering Chemistry Research*, 50(22):12678–12693, 2011. 10, 134

AppendixA

Illustrative Example of MC++ GDO Solver

In this chapter, the setup of a dynamic optimization problem in MC++ is illustrated, and the output results are explained. The setup is done in a `main` file that includes necessary header files. All the technical details relevant to the `main` file (e.g. header files, complier setup, local structure and class definitions) can be found in the doxygen documentation accompanying the code. With avoiding those details, this section aims to give the reader a flavor of the developed interface for GDO. Consider the van der Pol problem (5.7) with the variable bounds changed to $p \in [0.5, 1]$. The model setup is discussed in the following.

A.1 Model Setup

The dynamic model comprised of the ODE system and objective/constraint functions is formulated in a class called `DYNOPT`. It is derived from the class `DOSTRUCT` that defines variables and methods for storing the structure of the dynamic optimization problem. `DYNOPT` can be placed in the main file that calls the GDO solver or in a header file. The setup of Problem (5.7) in `DYNOPT` is illustrated in List. A.1. For convenient access, the problem's dimensions are assigned to *global* variables. According to control vector parameterization, the total number of variables `NP` is equal to the number of time stages `NS` for this one-variable problem.

The ODE right-hand sides, their initial conditions, objective function, and constraints are input in the member functions `RHS`, `IC`, `OBJ`, and `CTR`, respectively. The template declaration allows these methods to work with any data-types, like the ones defined for different bounding techniques. Three ODEs are defined in `RHS`, where the third one corresponds to the objective function. Note that the variable `p` must be indexed with the index of the current time stage `is`, as passed by `RHS`. This is to account for the control vector parameterization that creates multiple variables each corresponding to a specific time stage.

Unlike the `RHS` method, the state variables appear as two-dimensional arrays in the `OBJ` and `CTR` methods. The first dimension represents the time stage, at the end of which the values of the states are desired. The second dimension denotes the index of the state itself in the vector of state variables. Since the values at the final time are required in this problem, the index of the final stage `ns` is used in the objective and constraint functions. Note than these indexes start from zero. See in-line comments for more information.

Listing A.1: Setup of Problem (5.7) in MC++

```
1  ////////////////////////////////////////////////////////
2  // VAN DER POL PROBLEM IN BOCK AND PLITTE'S PAPER (1984)
3  ////////////////////////////////////////////////////////
4  //Specify problem dimensions
5  const unsigned int NS = 1;  //number of time stages in control vector
       parametrization
6  const unsigned int NP = NS; //total number of parameters
7  const unsigned int NX = 3;  //number of state variables
8  const unsigned int NC = 1;  //number of constraints (both equality and inequality)
9
10 //Class for defining dynamic model
11 class DYNOPT : public virtual mc::DOSTRUCT
12 {
13 public:
14   DYNOPT()
15     : mc::DOSTRUCT( NP, NX, NC ) {}
16
17   //Template method for defining ODE system
18   template <typename TX, typename TP> TX RHS
19     ( const unsigned int ix, const TP*p, const TX*x, const unsigned int is )
20     {
```

```
21    //is: index of current time stage
22      assert( ix < nx() );
23      using mc::sqr;
24      //Enter ODE right-hand side; [is] increments time stage
25      switch( ix ){
26        case 0: return x[1];                    //dx1/dt
27        case 1: return -x[0] + (1.-sqr(x[0]))*x[1] + p[is];   //dx2/dt
28        case 2: return 0.5*(sqr(x[0]) + sqr(x[1]) + sqr(p[is]));//dx3/dt
29        default: throw std::runtime_error("invalid size");
30      }
31    }
32
33  //Template method for defining ODE initial conditions
34  template <typename T> T IC
35    ( const unsigned int ix, const T*p )
36    {
37      assert( ix < nx() );
38
39      //Enter initial conditions
40      switch( ix ){
41        case 0: return 1.;  //x1(0)
42        case 1: return 0.;  //x2(0)
43        case 2: return 0.;  //x3(0)
44        default: throw std::runtime_error("invalid size");
45      }
46    }
47
48  //Template method for defining objective function
49  template <typename T> std::pair<T,t_OBJ> OBJ
50    ( const T*p, T* const*xk, const unsigned int ns )
51    {
52      //Enter objective function; xk[index of time stage][index of state]
53      return std::make_pair( xk[ns][2], MIN );  //index ns refers to final time
54    }
55
56  //Template method for defining constraints
57  template <typename T> std::pair<T,t_CTR> CTR
58    ( const unsigned int ic, const T*p, T* const*xk, const unsigned int ns )
59    {
```

```
60        //Enter constraints; xk[index of time stage][index of state]
61        switch( ic ){
62          case 0: return std::make_pair( xk[ns][0]-xk[ns][1]+1., EQ );//ns refers to
                final time
63          default: throw std::runtime_error("invalid size");
64        }
65      }
66  };
```

A.2 Solver Call and Settings

In this subsection, the settings and execution of the GDO solver are presented. These are done inside the `main` function of C++, as illustrated in List. A.2. Also specified in `main` are the bounds on the variables, initial guesses, and the final integration time. As shown in List. A.2, the GDO solver is created by instantiating an object of the class `DOSBB`. Now, the solver options can be specified. These options consist of the settings for the built-in libraries as well as the third-party packages. All these options are interfaced to the `DOSBB` class, and therefore, can be called directly using the object of `DOSBB`. This gives the user the convenience of controlling all the solver settings in one place. A number of these settings for the SBB, NLP solver (i.e. IPOPT), and ODE Bounder are given in List. A.2. In particular, observe that the Taylor model method `TM` has been selected as the bounding strategy. Note that polyhedral relaxations are automatically incorporated into any of the chosen bounding strategies.

After the solver's options have been set, the problem can be solved by calling the method `solve` of the class `DOSBB`. On termination, the return value of `solve` is a pair of two values: the incumbent value and its location on the variable domain. If the SBB terminates normally, these values correspond to the global solution.

In addition to the optimal solution, details of the SBB procedure can be output, as discussed in the next subsection.

Listing A.2: Solver call and settings for global solution of (5.7) using MC++

```
1  ////////////////////////////////////////
2  int main()
```

```
3  ////////////////////////////////////////
4  {
5    double p0[NP], tk[NS+1]; //vectors holding initial guesses and time intervals
6    I P[NP];     //vector holding parameter bounds
7    tk[0] = 0.; //initial time
8    for( unsigned int is=0; is<NS; is++ ){
9      P[is]  = I( 0.5, 1. );  //Interval bounds on parameters
10     p0[is] = 0.;       //initial guesses for parameters
11     tk[1+is] = tk[is]+5./(double)NS;//create vector of time intervals, final time=5
12   }
13
14   mc::DOSBB<I,DYNOPT> GDO;  //Instantiate an object of GDO solver
15
16   GDO.options.TAYLOR_MODEL_ORDER = 4; //Taylor model order
17   //Choose bounding strategy to be incorporated with polyhedral relaxations:
         Interval analysis: IA, McCormikc relaxations: MC,
18   // Taylor models: TM, McCormick-Taylor models: MCTM
19   GDO.options.IVP_BOUNDING_STRATEGY = mc::DOSBB<I,DYNOPT>::Options::TM;
20   GDO.options.USE_DOMAIN_REDUCTION  = true;
21   GDO.options.USE_CONSTRAINT_PROPAGATION  = false;
22   GDO.options_TModel().BOUNDER_TYPE = mc::TModel<I>::Options::LSB; //LSB: Lin and
         Stadtherr's approach of range bounder
23
24   //SBB settings
25   GDO.options_SBB().BRANCHING_STRATEGY = mc::SBB<I>::Options::MIDPOINT; //Branch on
         midpoint
26   //Choose criteria for selecting branch variable
27   GDO.options_SBB().BRANCHING_VARIABLE_CRITERION = mc::SBB<I>::Options::RGABS;
         //largest absolute range
28   GDO.options_SBB().ABSOLUTE_TOLERANCE = 1e-3;  //absolute tolerance in SBB
29   GDO.options_SBB().RELATIVE_TOLERANCE = 1e-3;  //relative tolerance in SBB
30
31   //ODE bounder settings
32   GDO.options_ODEBND().TSORDER = 10;  //order of Taylor series expansion in ODE
         bounder
33   GDO.options_ODEBND().HMIN = 1.e-8;  //minimum stepsize for integration in ODE
         bounder
34
35   //IPOPT settings
```

```
36   GDO.options_DOSEQ().CVTOL     = 1e-5;  //convergence tolerance for IPOPT
37   GDO.options_DOSEQ().GRADIENT = mc::DOSEQ<DYNOPT>::Options::FORWARD;
38   GDO.options_DOSEQ().SPARSE    = true;
39   GDO.options_DOSEQ().HESSIAN   = mc::DOSEQ<DYNOPT>::Options::LBFGS;
40
41   std::cout << GDO;
42   std::pair<double,const double*> optim;
43   optim = GDO.solve( NS, tk, P, p0 ); //execute the solver
44
45   std::cout << std::fixed << std::setprecision(1)
46      << "     ODE bounding: " << GDO.stats.cumul_ODEBND/GDO.stats.cumul_SBB*1e2 <<
          "%"
47      << "     Polyhedral relaxations:  " <<
          GDO.stats.cumul_FPREL/GDO.stats.cumul_SBB*1e2 << "%"
48      << "     Domain reduction:  " << GDO.stats.cumul_REDUC/GDO.stats.cumul_SBB*1e2
          << "%"
49      << "     LP solution:  " << GDO.stats.cumul_DOREL/GDO.stats.cumul_SBB*1e2 << "%"
50      << std::endl;
51   return 0;
52 }
```

A.3 Screen Output

During the SBB procedure, a screen report of the node-by-node results and solution statistics is provided. Such a report for the solution of Problem (5.7) is given in List. A.3. The level of the details shown on screen can be changed by the user. Major solver settings can be shown by the statement `std::cout << GDO`, where `GDO` is the object of `DOSBB` (see List. A.2). Also, a multi-column report is printed as the SBB proceeds. The information provided by each column is as follows.

- `INDEX:` the index of the current node, starting from 1 for the root node.
- `STACK:` the number of nodes currently stored in memory that have yet to be explored and decided whether they must be deleted or branched.
- `CUMUL TIME:` the cumulative CPU time taken by the solver so far.

- `RELAX:` the bound obtained from solving a relaxation of the *parent* of the current node. It will be the lower bound of the parent node for a minimization problem, and its upper bound for a maximization problem.
- `INC:` the incumbent value.
- `PARENT:` the index of the parent node. For the root node, the parent node is indexed as 0 (i.e. no parent node).
- `LBD:` the lower bound on the objective function obtained for the current node. It is the solution of the relaxed problem for a minimization, and the solution of the actual problem for a maximization.
- `UBD:` the upper bound on the objective function obtained for the current node. It is the solution of the actual problem for a minimization, and the solution of the relaxed problem for a maximization.
- `ACTION:` the index of the branching variable (starting from 0) in case the node must be branched. A node that must be deleted appears as FATHOM in this column.

Remark A.1. *The numbers -1e20 and +1e20 in List. A.3 represent $-\infty$ and $+\infty$, which are used at the root node to initialize the lower bound and upper bounds on the global solution, respectively. Also, the lower bound at other nodes is set to -1e20 if the LBS method is terminated with FAILURE.*

Remark A.2. *For a particular node, the* `UBD` *solution is skipped if the* `LBD` *turns out* `INFEASIBLE`*. This is consistent with the procedure outlined in §5.3, and is due to the fact that infeasibility of the relaxed problem always ensures that of the actual problem.*

As shown in List. A.3, the node-by-node results are followed by a solution summary. It includes the optimal solution, CPU time, total number of nodes, and maximum number of nodes stored in memory at the same time. Also reported is the portions of the CPU time spent for the LBS and UBS. For this problem, the LBS took about 85% of the total CPU time, while the UBS requierd only 15%. A breakdown of the CPU times taken by different tasks in LBS can also be output. This is

given in the last line of List. A.3 (see the bottom of List. A.2 for the statement used to generate these outputs). Note that the percentages shown are computed out of the total SBB CPU time.

Listing A.3: MC++ output for Problem (5.7)

```
-------------------------------------------------------------------------------
                        GLOBAL DYNAMIC OPTIMIZER IN MC++
-------------------------------------------------------------------------------
SPATIAL BRANCH-AND-BOUND OPTIONS:

  ABSOLUTE CONVERGENCE TOLERANCE                  1.0e-03
  RELATIVE CONVERGENCE TOLERANCE                  1.0e-03
  MAXIMUM NUMBER OF ITERATIONS (PARTITIONS)       NO LIMIT
  MAXIMUM CPU TIME (SEC)                          1.0e+06
  DISPLAY LEVEL                                   2
  INTERNAL VALUE FOR INFINITY                     1.0e+20
  UPDATE ENTIRE TREE AFTER AN INCUMBENT IMPROVEMENT? Y
  BRANCHING STRATEGY FOR NODE PARTITIONING        MIDPOINT
  BRANCHING TOLERANCE FOR VARIABLE AT BOUND       1.00e-03
  BRANCHING STRATEGY FOR VARIABLE SELECTION       RGABS
  INITIALIZATION THRESHOLD IN RELIABILITY BRANCHING 1
  SCORE PARAMETER IN RELIABILITY BRANCHING        0.17
  BOUNDING STRATEGY FOR PARAMETRIC IVP            TM
  ORDER OF TAYLOR MODEL                           4
  USE CONSTRAINT PROPAGATION?                     N
  USE OPTIMIZATION-BASED DOMAIN REDUCTION?        Y
  OPTIMIZATION-BASED REDUCTION MAX LOOPS          4
  LOOP THRESHOLD FOR OPTIMIZATION-BASED REDUCTION 20%
-------------------------------------------------------------------------------
INDEX STACK   CUMUL TIME          RELAX          INC      PARENT      LBD          UBD          ACTION

**********************************************************************
This program contains Ipopt, a library for large-scale nonlinear optimization.
 Ipopt is released as open source code under the Common Public License (CPL).
            For more information visit http://projects.coin-or.org/Ipopt
**********************************************************************

    1     1  0.000000e+00 -1.000000e+20  1.000000e+20      0      FAILURE      FAILURE  BRANCH0
    2     2  3.200200e-02 -1.000000e+20  1.000000e+20      1      FAILURE  2.764030e+00 BRANCH0
    3     3  6.400400e-02 -1.000000e+20  2.764030e+00      1      FAILURE      FAILURE  BRANCH0
    4     4  1.000070e-01 -1.000000e+20  2.764030e+00      2    INFEASIBLE     SKIPPED   FATHOM
    5     3  1.200080e-01 -1.000000e+20  2.764030e+00      2  1.130204e+00 2.764030e+00 BRANCH0
    6     4  1.440090e-01 -1.000000e+20  2.764030e+00      3    INFEASIBLE     SKIPPED   FATHOM
    7     3  1.600100e-01 -1.000000e+20  2.764030e+00      3    INFEASIBLE     SKIPPED   FATHOM
    8     2  1.720110e-01  1.130204e+00  2.764030e+00      5    INFEASIBLE     SKIPPED   FATHOM
    9     1  1.880120e-01  1.130204e+00  2.764030e+00      5  2.763998e+00     SKIPPED   FATHOM

#  NORMAL TERMINATION: 0.244016 CPU SEC  (LBD:85.2%  UBD:14.8%)
#  INCUMBENT VALUE:  2.764030e+00
#  INCUMBENT POINT:  6.885169e-01
#  INCUMBENT FOUND AT NODE: 5
#  TOTAL NUMBER OF NODES:   9
#  MAXIMUM NODES IN STACK:  4

  ODE bounding: 78.7%    Polyhedral relaxations: 3.3%   Domain reduction: 3.3%   LP solution: 0.0%
```