Global and Robust Optimization of Process Models with Embedded Simulations

Matthew E. Wilhelm, Ph.D.

University of Connecticut, 2022

ABSTRACT

Simulations arising from first-principles models or data-driven approaches find ubiquitous applications across technical fields. In the chemical industry, simulations are used to control unit operations, establish safe/efficient process configurations for producing a wide variety of products, and design key experiments. In a broader context, simulations often help to define value propositions for numerous products. Furthermore, robust optimization approaches, which rigorously account for uncertainty, are desirable when the consequences of decisions are extremely high. However, deterministic optimization (and, in turn, robust optimization) of general nonlinear forms remains a significant challenge due to the inherent computational complexity. This thesis addresses two key problems in nonconvex and robust optimization.

The focus of the first section is on advancing general methods for reduced-space deterministic global and robust optimization. The use of these methods for nonlinear models remains limited by high computational costs and strict structural requirements. The former problem may be mitigated by using fast and accurate relaxation methods. I present the development of two such methods. The first focuses on a specialized relaxation approach for emergent artificial neural networks, and the second advances reduced-space relaxation methods for intermediate composite bilinear terms. The development of the global optimizer EAGO, in which each of these methods is implemented,

Matthew E. Wilhelm, Ph.D.

University of Connecticut, 2022

is described. The composite relaxation approach used in EAGO allows for deterministic global and robust optimization of highly complex model formulations surmounting strict structural requirements.

The second portion of this thesis concerns the global solution of optimization problems with embedded systems of parametric ordinary differential equations. These extremely difficult problems are of interest in the verification of recurrent neural networks, optimal control of batch processes, as well as in methods for the detection and isolation of faults. The handful of existing algorithms which address these problems are subject to significant limitations. Prior work has focused on explicit discrete-time methods and continuous-time relaxation approaches, in contrast to this thesis which describes previously unexplored implicit relaxation approaches. These methods are typically more accurate than explicit approaches, less expansive, and are well-suited to challenging problems which embed stiff dynamical systems.

Global and Robust Optimization of Process Models with Embedded Simulations

Matthew E. Wilhelm

M.S., Columbia University in the City of New York, 2011

B.S., University of North Carolina at Greensboro, 2009

A Dissertation

Submitted in Partial Fulfillment of the

Requirements for the Degree of

Doctor of Philosophy

at the

University of Connecticut

2022

APPROVAL PAGE

Doctor of Philosophy Dissertation

Global and Robust Optimization of Process Models with Embedded Simulations

Presented by

Matthew E. Wilhelm, B.S. Math., M.S. ChemE.

Major Advisor	Prof. Matthew Stuber, PhD
Associate Advisor	Prof. George Bollas, PhD
Associate Advisor	Prof. Douglas Cooper, PhD
Associate Advisor	Prof. Kamil Khan, PhD
Associate Advisor	Prof. Ranjan Srivastava, PhD

University of Connecticut

2022

ACKNOWLEDGMENTS

I find myself at this milestone reflecting on the totality of the doctoral student experience. As much as a doctorate is a distinction built on my own efforts, it reflects the people in my life that helped me along the way. I am fortunate to have a great number of people in my life who have made this experience truly special. More people deserve recognition here than I can reasonably commit to page, but I will try and hopefully avoid the most egregious omissions.

There is no doubt in my mind that my primary advisor Prof. Stuber, deserves the first mention here. Choosing a first year professor as an advisor is a gamble by any means, but it is a gamble that I am certain I won. It is entirely clear that Prof. Stuber epitomizes the ”yes-and” advisor who and engages thoroughly with new ideas and find ways to improve upon them. Moreover, being able to do this while remaining flexible enough to allow students to shape their own experience is a truly remarkable feat. I am truly grateful for the wonderful experience.

I would further like to acknowledge my committee members: Prof. Douglas Cooper, Prof. George Bollas, Prof. Kamil Khan, and Prof. Ranjan Srivastava. In particular, Prof. Bollas was the first UCONN faculty I met with when I was initially considering coming back for a PhD. He sold me on UCONN as an institution and a great department culture. Moreover, he encouraged me to fully explore my options with respect to advisors and projects when I first joined the program and handled me joining the Stuber group with a great deal of grace, something I am quite thankful for. Additionally, I would like to thank Prof. Khan for helping show me the ropes of the AIChE and ISMP conferences. I found the discussions of both differential McCormick relaxations and broader optimization approaches invaluable. Moreover, the preliminary feedback on EAGO provided by

him and his doctoral students Huiyi Cao and Yingkai Song was invaluable to the early stage software development. It was a pleasure to TA control theory with Prof. Cooper, whose practical prospective is something I find invaluable. Prof. Srivastava, while being the much overworked department-head, always found a way embody what is great about the ChEG departments culture – a warm and inquisitive collegiality.

I am grateful to Professors Mikhail Bragin and Liang Zhang for providing me with a deeper exposure to the traditional operations research and scheduling community as part of a wonderful collaboration. I am truly grateful to have worked with a number of wonderful graduate students and undergraduate students during my time at UCONN: Chenyu Wang, William Hale, Robert Gottlieb, Anne Le, and Shaylin Cetegen. Chenyu, my friend, I wish you all the best going forward. The experience being the first graduate students in a new group is one wrought with challenge, and I could not have asked for a better colleague to share this with. The time I have spent working with Robert while in the penultimate stage of the doctoral program has assured me that the prospects of this research group are quite bright indeed.

As important as my experience at UCONN has been for my development, I have to acknowledge the huge impact that that all of my mentors Corning Incorporated had on me. Dr. Srini Vemury, Bill Kish, Dr. Victor Jin, and Casey Volino, you all made the day-to-day work a joy and provided me with a clear view of what lifelong learning looks like. Moreover, I am sure that I would have never made it to this point without the support of Prof. Laura Kaufman when I was at Columbia and the wonderful group she cultivated there or Prof. Maya Chhetri, Prof. Jan Rychtář and all the wonderful friends I made at UNCG.

I would like to thank my close friends during my time here. With respect to Chris Forney and Alexis Woodbury congratulations on the engagement. It was a pleasure having Tim and Erin Murdock just a few hours away. All the weekend adventures certainly helped me stay relatively sane during this whole PhD experience. Agnieszka Rec was always there with the next well-timed

pun, Stephanie Ranks was always up for the next misadventure, and John Burden remains the best house guest we have had to this day. I would also like to acknowledge my dog Carly for doing an admirable job in warming my feet while I write these acknowledgements.

I would be remiss if I did not also express my gratitude to all the teachers, friends, and mentors that made such an impact on my formative years. Foremost among these are my parents, Steve and Anne Wilhelm. I learned patience and gained a sense of perspective from you. Your love and support are the reason I am here. I should also thank my brothers Rosja and Justin as well as close friends from my early years: Lee Sult, Trevor Hudsepth, Josue Monge, Dylan/Julia Youngsmith you are always a reminder of where I come from.

Lastly, and most importantly, I would like to thank my wonderful partner Gina Hurley for her unconditional support. Your company and encouragement mean everything to me.

This research was supported by funding from the Air Force Office of Scientific Research, the National Science Foundation, and the University of Connecticut.

Contents

When one admits that nothing is certain one must, I think, also admit that some things are much more nearly certain than others.

- Bertrand Russell

We demand guaranteed rigidly defined areas of doubt and uncertainty.

- Douglas Adams

List of Figures

List of Tables

Publications Related to this Thesis

1. Chapter 3 is based on ”M.E. Wilhelm and M.D. Stuber. EAGO.jl: easy advanced global optimization in Julia. *Optimization Methods and Software*. Pages 1-26. 2020. doi: 10.1080/10556788.2020.1786566”

2. Chapter 4 is based on ”M.E. Wilhelm, C. Wang, and M.D. Stuber. Convex and Concave Envelopes of Artificial Neural Network Activation Functions for Deterministic Global Optimization. *Journal of Global Optimization*, In Press 2022, doi: s10898-022-01228-x.”

3. Chapter 5 is based on ”M.E. Wilhelm, and M.D. Stuber. Improved convex and concave relaxations of composite bilinear forms. *Under review.*.”

4. Chapter **??** is based on ”M.E. Wilhelm, and M.D. Stuber. DynamicBounds.jl: Software for computing state bounds and relaxations of parametric differential equations. *Under review.*”

5. Chapter 6 is based on M Wilhelm, A. Le, M Stuber. Global Optimization of Stiff Dynamical Systems. *AIChE Journal*, Futures Issue, 65 (12): e16836, 2019. doi: 10.1002/aic.16836.

6. Chapter 7 is based on ”M.E. Wilhelm, and M.D. Stuber. Convex/Concave Implicit Relaxation Methods for Parametric Ordinary Differential Equations for Deterministic Global Optimization. *Under review.*.”

7. Chapter 8 is based in part on ”Chenyu Wang, M.E. Wilhelm, and M.D. Stuber. Semi-Infinite Optimization with Hybrid Models. Industrial & Engineering Chemistry Research. 2022, 61, 15, 5239-5254. doi: 10.1021/acs.iecr.2c00113”

Chapter 1

Introduction

1.1 Motivation

Mathematical models and simulations thereof are ubiquitous throughout technical disciplines. These models/simulations inform the design and operation of many technologies encountered in our everyday life. Recent advances in machine learning (ML) have expanded the domains informed by mathematical modeling by providing tractable approaches to problems spanning image recognition [126], natural language processing [85], and process control [142]. As the complexity of model-based approaches have grown, more and more applications incorporate multiple types of simulations in different aspects. This has led to interest in modular, equation-oriented, and acausal models to streamline the workflow associated with addressing the modeling challenges. While hybrid data-driven modeling approaches, such as physics-informed neural networks, are key enablers of Industry 4.0 and digital twin paradigms, a number of extant challenges exist. In particular, the development of effective approaches for dynamic systems that

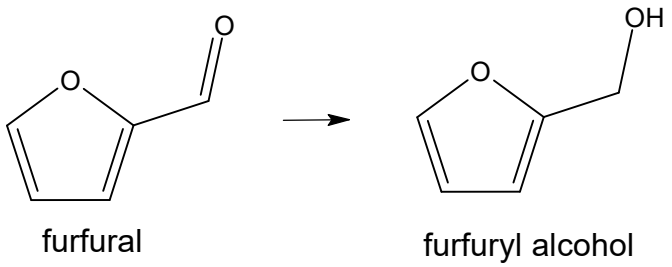

FIGURE 1.1.1: Conversion of furfural to furfuryl alcohol.

adequately address uncertainty is of significant interest. These issues can be made concrete by through examination of a proposed electrochemical synthesis approach for furfuryl alcohol.

Furfuryl alcohol (FA) is the principal monomer used in the synthesis of furan resins, which are key components in numerous coatings, adhesives, and thermoset polymers. This biomass-derived component plays a key role in the development of renewable alternatives to a number of commodity chemicals and fuels. Currently, the synthesis of FA makes use of a high temperature and pressure process step resulting in inherent process hazards. One proposed means of mitigating these hazards while providing adequate selectivity and specificity for FA production is through the use of a low-temperature electrochemical synthesis pathway. This is a two-step method in which furfural (FF) is adsorbed onto a surface, and then reacts with adsorbed hydrogen. Mass balance considerations yield the following system of differential equations (1.1.1) for a semi-batch tank reactor where Q_{in} is the input flowrate, Q_{out} is the output flowrate, V

is the reactor volume, and a is the effective surface area [107].

$$\begin{aligned}\frac{dx_1}{dt} &= C(r_1 - r_2) \\ \frac{dx_2}{dt} &= \frac{Q_{in}}{V}x_{2,in} - r_1 a - \frac{Q_{out}}{V}x_2 \\ \frac{dx_3}{dt} &= \frac{Q_{in}}{V}x_{3,in} + r_2 a - \frac{Q_{out}}{V}x_3\end{aligned} \tag{1.1.1}$$

The rate of adsorption of FF onto the surface, r_1 and the rate of conversion from FF to FA, r_2, are then given by equations (1.1.2) and (1.1.3), respectively,

$$r_1 = k_1 x_2 \theta_v, \tag{1.1.2}$$

$$r_2 = k_2 \theta_H (E)^2 x_1. \tag{1.1.3}$$

A simple multi-layer perception (MLP), $\theta_H = \theta_H(E)$, relates the over-potential E with the absorption of hydrogen by the Volmer reaction, θ_H, which in turn determines the production rate of FA. This MLP-based hybrid model outperforms predictions based derived solely from first-principles considerations. The impact of uncertainty in empirically-estimated kinetic rate coefficients on yield and cycle time must be taken into account when designing an FA synthesis process. Moreover, biomass-derived feedstocks such as FF may be subject to substantial variations in composition, which may dramatically affect techno-economic analyses. Formally accounting for uncertainty in this FA production design problem is tantamount to answering a simple question:

> *Can one select a design Q_{in}, Q_{out}, E, such that for any realization of $(k_1, k_2, x_{1,0})$ within specified bounds desired yield and cycle time are achieved?*

These conservative, or worst-case design under uncertainty problems ensure that reasonable

performance is guaranteed prior to substantial capital investment. In practice, one typically formulates this question as multi-level optimization problem (see Section 2.5) that is solved to select design parameters [120]. Moreover, worst-case designs are of particular interest for safety-critical systems and timeline-sensitive projects in which the cost of a single failure dramatically outweighs a marginally larger capital investment.

The furfuryl alcohol model leverages a physics-informed neural ordinary differential equation (ODE), an emerging class of machine learning that builds on a now common recurrent neural network (RNN) framework. While a RNN [178]) builds elaborate transformations through the sequential compositions of simpler transformations $\mathbf{f} : \mathbb{R}^{n_y} \times \mathbb{R}^{n_\theta} \rightarrow \mathbb{R}^{n_y}$ as a applied to hidden states, $\mathbf{y}_t$,

$$\mathbf{y}_{t+1} = \mathbf{y}_t + \mathbf{f}(\mathbf{y}_t, \boldsymbol{\theta}) \tag{1.1.4}$$

for $t \in 0, \ldots, T$, the neural ODE is an analogous continuous transformation (1.1.5)

$$\frac{d\mathbf{y}}{dt} = \mathbf{f}(\mathbf{y}(t), \theta), \tag{1.1.5}$$

which leads to formulation in which $\mathbf{f}$, the right-hand side function participating in the ODE, resembles a simple neural network. Highly successful machine learning architectures such as ResNet [126] and FractalNet [159] can be interpreted as Euler and Runge-Kutta discretizations of (1.1.5), respectively. These neural ODEs are memory-advantaged relative to formulation (1.1.4) and do not require the specification of the number of network layers beforehand.
Physics-informed dynamic neural networks incorporate these benefits as well as rigorously accounting for physical laws, such as mass conservation. Unfortunately, models of this form are distinct from the typical algebraic equation structure addressed by commercial software.

The presence of nonconvex constraints implied by the incorporation of (1.1.5) into an optimization problem leads to a nonconvex dynamic program which is a particularly challenging class of optimization problem before accounting for uncertainty. Moreover, the mere verification of system-level robustness, requires a program of this form to be solved to global optimality [188] which necessitates the use of NP-complete methods at the forefront of numerical research. The full problem formulation is a bilevel programs with coupling equality constraints that implicitly define functions [78]. General problems of this form cannot be readily solved. However, it may be addressed by exploiting a implicit function reformulation to a semi-infinite programs with existing solution methods [298]. In practice, this requires the calculation of convex/concave relaxation of implicit functions [300] and is enabled by composite relaxation methods [191, 271] that allow for the modular consideration of substructures.

Robust optimization and associated methods play a pivotal role in addressing a number of concerns beyond the worst-case design under uncertainty problem. As the preponderance of machine learning architectures are opaque to users, a number of formal methods have been employed to ensure that trained models exhibit desired properties. Neural network verification, as a specialization of model verification, seeks to answer basic "yes" or "no" questions meant to rule out undesirable behavior [86, 141, 233]. These common *adversarial verification methods* (or *adversarial attacks*) seek to identify inputs that lead to undesirable behavior termed *adversarial examples* and training methodologies that can formally ensure that for any input, in any current (and potentially unknown) state of a system, undesirable behaviors cannot occur, which is itself a robust optimization problem [32].

This thesis advances methods for reduced-space deterministic global optimization and robust optimization that are general enough to addresses forms such as (1.1.1). This begins with the development of the reduced-space global optimizer EAGO.jl and the establishment of specialized

relaxation approaches for common operators including bilinear term and activation functions. Subsequently, a reduced-space dynamic optimization method is described along with its corresponding implementation and robust optimization applications are considered.

1.2 Thesis Structure

The subsequent chapters of this thesis are organized in the following manner. Mathematical preliminaries are initially reviewed in Chapter 2. Chapters 3 - 5 describe the development of general methods pertinent to reduced-space global optimization while Chapters **??** - 7 specialize these methods to dynamic optimization. In Chapter 8, a few applications are highlighted which make use of the aforementioned methods in a robust optimization context. Finally, the thesis concludes by summarizing research contributions made herein and discussing potential avenues for future research. A detailed summary of each chapter is provided below.

- Chapter 2 provides an overview of the mathematical preliminaries relevant to this research and a discussion of prior work. The topics covered in this chapter include problem formulations, global optimization, robust optimization, and methods for enclosing the range of a function.

- Chapter 3 details the development of an extensible deterministic global optimizer (EAGO). This optimizer is based on a first-of-its-kind implementation of McCormick arithmetic using a combination of source code transformation, multiple dispatch, and context-specific approaches. Moreover, the performance of EAGO is comparable and may exceed state-of-the-art commercial optimizer performance.

- Chapter 4 describes general methods used to compute convex/concave relaxations of

activation functions that are commonly chosen for use in artificial neural networks (ANNs). This chapter details the development of a library of envelopes of thoroughly studied rectifier-type and sigmoid activation functions, in addition to the novel self-gated sigmoid-weighted linear unit (SiLU) and Gaussian error linear unit (GELU) activation functions. These envelopes of activation functions lead to tighter relaxations of ANNs on their domain and lead to a significant reduction in CPU runtime required to solve global optimization problems involving ANN models.

- Chapter 5 details a novel theory used to generate necessarily tighter reduced-space McCormick relaxations when *a priori* convex/concave relaxations of intermediate bilinear terms are known. This chapter then details three different approaches to generating *a priori* relaxations of the arguments participating in intermediate bilinear terms. It is then shown that the use of subgradient-based *a priori* relaxations result in an improvement to CPU runtime of the branch-and-bound algorithm.

- Chapter 6 presents a deterministic global optimization method that uses unconditionally-stable implicit integration methods to reformulate ODE-constrained optimization problems into a nonconvex nonlinear program (NLP) with implicit functions embedded. This problem is then solved to global optimality in finite time using a spatial B&B framework that progressively evaluates convex/concave relaxations of implicit functions in a block sequential fashion.

- Chapter 7 adapts the method presented in Chapter 6 to incorporate rigorous bounds on the truncation error of embedded parametric differential equation systems. This allows for the incorporation of error bounds to ensure state bounds are rigorous despite them being calculated using a numerical method. In turn, this allows for the use of adaptive step-sizing approaches that leads to a computational speed-up; overcoming the time cost associated

with computing error bounds.

- Chapter 8 details an application of robust optimization which incorporates advances made throughout this thesis.
- Chapter 9 summarizes the contributions presented in this thesis and potential future avenues of research are outlined.

Chapter 2

Mathematical Background

2.1 Interval Analysis

Throughout this thesis, scalar quantities are represented as lower-case letters (e.g., a) and vectors are denoted by boldface lower-case letters (e.g., $\mathbf{a}$). A_i representing the i-th component of the vector A. A set B^n is defined as the Cartesian product $B^n \equiv B \times \cdots \times B$ for $B \subset \mathbb{R}$. Additionally, $X = [\mathbf{x}^L, \mathbf{x}^U]$ will represent an n-dimensional interval that is a nonempty compact set defined as $X = \{\mathbf{x} \in \mathbb{R}^n : \mathbf{x}^L \leq \mathbf{x} \leq \mathbf{x}^U\}$ with $\mathbf{x}^L$ and $\mathbf{x}^U$ the lower and upper bounds of the interval, respectively. A set X^n is defined as the Cartesian product $X^n = X \times X \times \cdots \times X$. Further, let $\mathbb{IR}^n = \{[\mathbf{x}^L, \mathbf{x}^U] \subset \mathbb{R}^n\}$ be the set of all n-dimensional real intervals and for any $D \subset \mathbb{R}^n$, $\mathbb{I}D = \{X \in \mathbb{IR}^n : X \subset D\}$ is the set of all interval subsets of D. The image of X under the mapping $\mathbf{f} : D \to \mathbb{R}^n$ will be denoted $\mathbf{f}(X)$ whereas the inclusion monotonic interval extension of $\mathbf{f}$ on X will be denoted $F(X)$. From the Fundamental Theorem of Interval Analysis (Thm. 1.4.1 in Neumaier [219]), we have $\mathbf{f}(X) \subset F(X),\ \forall X \in \mathbb{I}D$. Further, let the *diameter* of a scalar-valued

interval, X, be defined as $\text{diam}(X) = x^U - x^L$ and the *radius* be given by $\text{rad}(X) = \text{diam}(X)/2$.

An interval extension of algebraic functions can generally be computed by applying interval extensions to each subexpression in the function, the *natural interval extension*. This naturally leads to overestimation due to either the dependency problem or wrapping effects. The *dependency problem* arises due to the inability of interval analysis to identify and compensate for the appearance of the same variable at multiple points in a given expression [193]. The function $g : x \to x/x$ evaluated on $x \in X = [1, 2]$ clearly illustrates this phenomena. Obviously, g takes the value of 1 for all $x \in [1, 2]$. However, the natural interval extension of $g(X)$ evaluates to $[1, 2]/[1, 2] = [1/2, 2] \neq [1, 1]$. In comparison, the *wrapping effect* arises from the fact that the image set of a function on n-dimensional interval is not itself an m-dimensional interval. These limitations motivate interest in developing alternative enclosure methodologies that mitigate this overestimation.

2.2 Convex Analysis

Convex and concave relaxations provide a means of rigorously enclosing the range of a function that is computationally useful in global optimization. The pertinent definitions and background information for convex sets, convex functions, and, in turn, convex relaxations are defined herein.

Definition 2.2.1 (Convex Set [51])**.** A set $Z \subset \mathbb{R}^n$ is said to be convex if, for every $\mathbf{x}, \mathbf{y} \in Z$, we have $\alpha\mathbf{x} + (1 - \alpha)\mathbf{y} \in Z$, $\forall \beta \in [0, 1]$.

Definition 2.2.2 (Convex/Concave Function [51])**.** Given a convex set $Z \subset \mathbb{R}^n$, a function $f : Z \subset \mathbb{R}^n \to \mathbb{R}$ is said to be convex if

$$f(\beta\mathbf{x} + (1 - \beta)\mathbf{y}) \leq \beta f(\mathbf{x}) + (1 - \beta) f(\mathbf{y}) \tag{2.2.1}$$

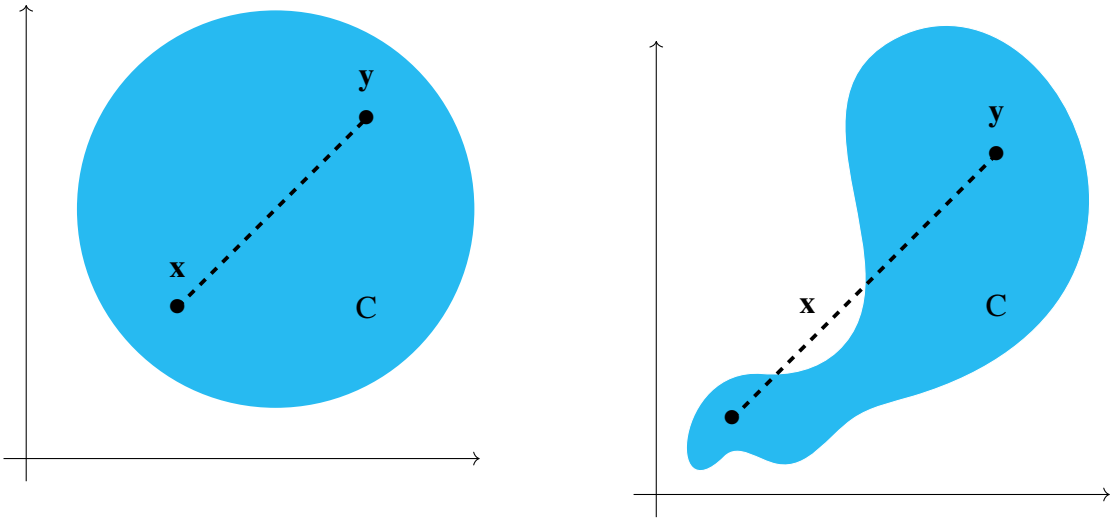

FIGURE 2.2.1: Illustration of a (**left**) convex set and a (**right**) nonconvex set.

for every $\mathbf{x}, \mathbf{y} \in Z$, we have $\beta\mathbf{x} + (1 - \beta\mathbf{x}) \in Z$, $\forall \beta \in [0, 1]$. Similarly, a function $f : Z \to \mathbb{R}$ is said to be concave if the reverse of inequality (2.2.1) holds.

Graphically, a function f is convex, provided that any line segment connecting the points $(\mathbf{x}, f(\mathbf{x}))$ and $(\mathbf{y}, f(\mathbf{y}))$ is above f on Z as depicted in Figure 2.2.2.

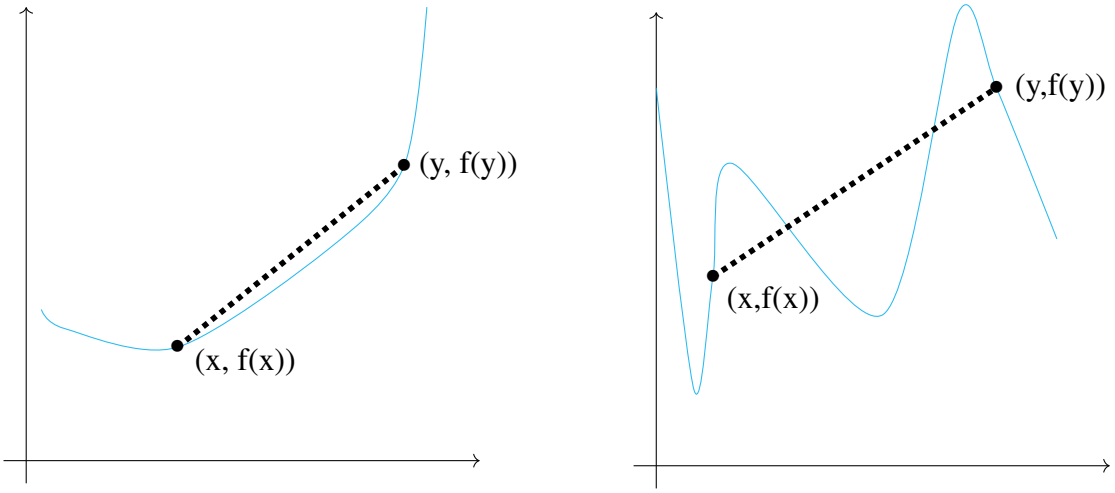

FIGURE 2.2.2: An illustration of (**left**) a convex function and (**right**) nonconvex function.

Definition 2.2.3 (Underestimators and Overestimators)**.** Given $Z \subset \mathbb{R}^n$, and define $w : Z \to \mathbb{R}$. Then, a function $u : Z \to \mathbb{R}$ is called an *underestimator* of w on Z if and only if $u(\mathbf{z}) \leq w(\mathbf{z})$ for every $\mathbf{z} \in Z$. Similarly, $o : Z \to \mathbb{R}$ is called an *overestimator* of w on Z if and only if $o(\mathbf{z}) \geq w(\mathbf{z})$ for every $\mathbf{z} \in Z$.

Definition 2.2.4 (Convex and Concave Relaxations [191])**.** Given a convex set $Z \subset \mathbb{R}^n$ and a function $w : Z \to \mathbb{R}$, $w^{cv} : Z \to \mathbb{R}$ is a *convex relaxation* of w on Z if and only if it is both convex and an underestimator of w on Z. Similarly, $w^{cc} : Z \to \mathbb{R}$ is a *concave relaxation* of w on Z provided it is both concave and an overestimator of w on Z.

Note that this definition involves scalar functions. Convex and concave relaxations of vector-valued functions of $\mathbf{f} : Z \to \mathbb{R}^n$ are then defined by applying the above inequalities componentwise [300]. The tightest convex relaxation of f on the domain Z and the tightest concave relaxations are the convex and concave envelope, respectively, as given by Definition 2.2.5.

Definition 2.2.5 (Convex and Concave Envelope [334])**.** Let $f : Z \to \mathbb{R}$ where $Z \subset \mathbb{R}^n$ is a nonempty convex set. The *convex envelope* of f on Z is the convex relaxation $f^{cv,env} : Z \to \mathbb{R}$ such that $f^{cv}(\mathbf{z}) \leq f^{cv,env}(\mathbf{z})$ holds for all $\mathbf{z} \in Z$ and every convex relaxation f^{cv} of f on Z. Similarly, the *concave envelope* of f on Z is the concave relaxation $f^{cc,env} : Z \to \mathbb{R}$ such that $f^{cc}(\mathbf{z}) \geq f^{cc,env}(\mathbf{z})$ holds for all $\mathbf{z} \in Z$ and every concave relaxation f^{cc} of f on Z.

Definition 2.2.6 (Subgradients [191])**.** Let $Z \subset \mathbb{R}^n$ be a nonempty convex set, $w^{cv} : Z \to \mathbb{R}$ be convex, and $w^{cc} : Z \to \mathbb{R}$ be concave. A vector $\mathbf{s}_w^{cv} \in \mathbb{R}^n$ is a *subgradient* of w^{cv} on Z if for each $\bar{\mathbf{z}} \in Z, w^{cv}(\mathbf{z}) \geq w^{cv}(\bar{\mathbf{z}}) + (\mathbf{s}_w^{cv})^{\mathrm{T}}(\mathbf{z} - \bar{\mathbf{z}}), \forall \mathbf{z} \in Z$. Similarly, a vector $\mathbf{s}_w^{cc} \in \mathbb{R}^n$ is a *subgradient* of w^{cc} on Z if for each $\bar{\mathbf{z}} \in Z, w^{cc}(\mathbf{z}) \leq w^{cc}(\bar{\mathbf{z}}) + (\mathbf{s}_w^{cc})^{\mathrm{T}}(\mathbf{z} - \bar{\mathbf{z}}), \forall \mathbf{z} \in Z$.

Remark 2.2.1. Note that subgradients of vector-valued functions and subgradients of matrix-valued functions will be defined analogously and will be matrix-valued functions and

third-order tensor-valued functions, respectively. The subgradients of w^{cv} and w^{cc} at $\bar{\mathbf{z}} \in Z$ are then denoted by $\mathbf{s}_w^{cv}(\bar{\mathbf{z}})$ and $\mathbf{s}_w^{cc}(\bar{\mathbf{z}})$, respectively.

2.3 Global Optimization

The general form of optimization problems are given by (2.3.1):

$$\begin{aligned} f^* = \min_{\mathbf{z} \in Z = [\mathbf{z}^L, \mathbf{z}^U]} & f(\mathbf{z}) \\ \text{s.t.} \quad & \mathbf{h}(\mathbf{z}) = \mathbf{0} \\ & \mathbf{g}(\mathbf{z}) \leq \mathbf{0} \\ & \mathbf{A}\mathbf{z} = \mathbf{b} \end{aligned} \tag{2.3.1}$$

where $f : Z \to \mathbb{R}$ is the *objective* function, and $\mathbf{g} : Z \to \mathbb{R}^{n_g}$ and $\mathbf{h} : Z \to \mathbb{R}^{n_h}$ are *constraint* functions. The lower and upper bounds on the variables are given by $\mathbf{z}^L, \mathbf{z}^U \in \mathbb{R}^n$, respectively. The matrix equation specified by $\mathbf{A} \in \mathbb{R}^{m \times n}$, $\mathbf{b} \in \mathbb{R}^m$ defines the linear equality constraints. In general, optimization problems are classified by either the functional forms of the participating objectives and constraints, the characteristics of participating variables, or by the implied mathematical properties. A problem (2.3.1) is a *linear program* (LP) if $f, \mathbf{g}, \mathbf{h}$ are linear and a *second-order cone program* (SOCP) if $\mathbf{g}$ consists only of second-order conic constraints and $f, \mathbf{h}$ are affine functions, as formalized in Definitions 2.3.1, and 2.3.2. Accordingly, 2.3.1 is a *nonlinear program* (NLP) if even a single constraint or the objective function is a nonlinear function. A problem (2.3.1) is *continuous* if all variables $\mathbf{z}$ are continuous (i.e., may take any value in a continuous range), *integer* if all such variables are drawn from sets of integers, or *mixed-integer* if both continuous and integer variables participate in the formulation.

Definition 2.3.1 (Linear Program [29])**.** A *linear* optimization problem takes the form

$$\begin{aligned} &\min_{\mathbf{z}} \mathbf{c}^T\mathbf{z} + d \\ &s.t.\ \ A\mathbf{z} \leq \mathbf{0}, \end{aligned} \tag{2.3.2}$$

where $\mathbf{z} \in R^n$ is the vector of decision variables, $\mathbf{c} \in R^n$ and $d \in \mathbb{R}$ form the objective, A is an $m \times n$ constraint matrix, and $\mathbf{b} \in \mathbb{R}^m$ is the right hand side vector.

Definition 2.3.2 (Second-Order Cone Program (SOCP)[29])**.** A *second order conic* optimization problem takes the form

$$\begin{aligned} &\min_{\mathbf{z}} \mathbf{c}^T\mathbf{z} \\ &s.t.\ \ ||A_i\mathbf{z} - b_i||_2 \leq \mathbf{c}_i^T\mathbf{z} - d_i, \qquad 1 \leq i \leq m \end{aligned} \tag{2.3.3}$$

where $\mathbf{z} \in R^n$ is the vector of decision variables, $\mathbf{c} \in R^n$ and $d \in \mathbb{R}$ form the objective, A is an $m \times n$ constraint matrix, and $\mathbf{b} \in \mathbb{R}^m$.

In order to ensure that problem (2.3.1) is well-posed, it is typical to assume that all functions are continuous and suitably bounded [136]. In this thesis, we refer to variables that appear in any nonlinear expressions as *nonlinear* variables and variables that appear in any nonconvex equality, inequalities, or objective as *nonconvex* variables. The convexity tests commonly used in optimization routines check sufficient conditions such as determining that natural interval extension of the Hessian of the augmented Lagrangian is everywhere positive-definite [220]. As a consequence, the expressions referred to as *convex* should be understood to be expressions that have passed sufficient conditions for convexity.

Most general solution methods for nonconvex problems of the form (2.3.1) rely on some

variation of the spatial branch-and-bound algorithm (B&B) [136]. In this algorithm, the decision space is iteratively partitions into subdomains $Z_l, Z_{l'}, \ldots$ until the algorithm converges. A lower bound f_l^{LBD} and an upper bound f_l^{UBD} of the solution to (2.3.1) restricted to a given subdomain Z_l are computed. Subdomains are then fathomed, discarded from consideration, based on infeasibility when f^{LBD} is less than a known global upper bound of α_k. Most complete global solvers intersperse these bounding calculations with a series of methods that shrink the size of the subdomain Z_l [234]. Any feasible point on subdomain Z_l is a valid upper bound f_l^{UBD}; however, suboptimal local solutions of (2.3.1) are commonly used to compute upper bounds. In general, nonconvex solution methods make use of advanced methods that rely on solving under-approximating mixed-integer convex optimization problems to rigorously furnish lower bounds.

A continuous optimization problem of the form (2.3.1) is convex, provided that: 1) the objective function f is convex, 2) all inequality constraint functions $\mathbf{g}$ are convex, and 3) all participating equality constraints are affine $n_h = 0$ [51]. As an immediate consequence of the limitations on the constraints, the feasible set of the convex optimization problem is convex. Accordingly, we may state that a convex optimization problem is the minimization of a convex objective function over a convex set. Convexity of an optimization problem implies that every *local minima* on a particular neighborhood is a *global minima*. As a consequence, any convex problem that outer approximates the feasible set of (2.3.1) and under estimates the objective function value, may be solved locally to rigorously compute a lower bound of a nonconvex optimization problem (2.3.1). This *convex relaxation* of (2.3.1) may then be constructed by taking the convex relaxation of all participating nonconvex constraints and the objective function, succinctly below in (2.3.4),

$$
\begin{aligned}
f^* = \quad & \min_{\mathbf{z}\in Z=[\mathbf{z}^L,\mathbf{z}^U]} f^{cv}(\mathbf{z}) \\
\text{s.t.} \quad & \mathbf{g}^{cv}(\mathbf{z}) \leq \mathbf{0} \\
& \mathbf{h}^{cv}(\mathbf{z}) \leq \mathbf{0} \\
& \mathbf{h}^{cc}(\mathbf{z}) \geq \mathbf{0} \\
& \mathbf{A}\mathbf{z} = \mathbf{b}
\end{aligned}
\tag{2.3.4}
$$

The solution of (2.3.4) is a valid lower bound of (2.3.1). Pointwise convergence of relaxations of the underlying functions f, $\mathbf{g}$, and $\mathbf{h}$ is sufficient to ensure pointwise convergence to the solution f^* of (2.3.4) under mild assumptions [136]. In turn, this ensures that the B&B algorithm furnishes an ϵ-optimal global solution to (2.3.1) within a finite number of iterations provided that the upper bounding method is also convergent; as is the case when using locally optimal solutions of (2.3.1).

Given that a large number of convex NLPs would typically need to be solved during B&B, the solution of a convex NLP of form (2.3.4) often represents a significant computational cost and may lead to numerical robustness issues. This contrasts existing LP and MILP optimization methods that typically solve a problem of similar dimension in much less CPU time, and are typically more robust to numerical scaling issues. Moreover, a highly-accurate solution of (2.3.4) may not yield a substantial improvement in the lower bound of (2.3.1) on Z for a weak approximation of (2.3.4) due to the inherent underestimation that arises from relaxing the original problem. Each of these factors motivates the use of linearization methods as opposed to a direct solution of (2.3.4).

Relaxation and subgradient pairs are calculated at specific reference points to construct affine relaxation at a particular points, $\mathbf{z} = \bar{\mathbf{z}}$. These affine relaxations are then used in place of the

original convex/concave relaxations in (2.3.4) to form LP or MILP outer-approximations of the underlying set given by (2.3.5).

$$
\begin{aligned}
f^* = \quad & \min_{\mathbf{z}\in Z=[\mathbf{z}^L,\mathbf{z}^U],\eta} \eta && && (2.3.5)\\
\text{s.t.} \quad & \mathbf{g}_j^{cv}(\bar{\mathbf{z}}_i) + \mathbf{s}_g^{cv,T}(\bar{\mathbf{z}}_{if})(\mathbf{z}-\bar{\mathbf{z}}_{if}) \le \mathbf{0} && \text{for } i=1,\dots,n_{zg},\ j=1,\dots,n_g\\
& \mathbf{h}^{cv}(\bar{\mathbf{z}}_i) + \mathbf{s}_h^{cv,T}(\bar{\mathbf{z}}_{if})(\mathbf{z}-\bar{\mathbf{z}}_{if}) \le \mathbf{0} && \text{for } i=1,\dots,n_{zhu}\\
& \mathbf{h}^{cc}(\bar{\mathbf{z}}_i) + \mathbf{s}_h^{cc,T}(\bar{\mathbf{z}}_{if})(\mathbf{z}-\bar{\mathbf{z}}_{if}) \ge \mathbf{0} && \text{for } i=1,\dots,n_{zho}\\
& f^{cv}(\bar{\mathbf{z}}_i) + \mathbf{s}_f^{cv,T}(\bar{\mathbf{z}}_{if})(\mathbf{z}-\bar{\mathbf{z}}_{if}) \le \eta && \text{for } i=1,\dots,n_{zi}\\
& \mathbf{A}\mathbf{z} = \mathbf{b}
\end{aligned}
$$

For general convex functions, efficient linearization strategies remain an area of active research, as the selection of the set $\bar{\mathbf{z}}_1,\dots,\bar{\mathbf{z}}_n$ may significantly affect the quality of the relaxation [210]. However, specialized approaches, such as the sandwich method [304] can be employed when problems consist solely of unary or binary operators.

2.3.1 Auxiliary Variable Method

Many modern global optimization methods exploit the *auxiliary variable method* (AVM) [291, 303, 305] to compute relaxations. This method consists of a two-step process in which a factorable program is decomposed into an equivalent problem through the introduction of auxiliary variables corresponding to factors. A mathematical program in which all participating functions are factorable, as described in Definition 2.3.4, is referred to as a *factorable program*.

Definition 2.3.3 (Univariate Intrinsic Function [271])**.** The function $u : B \subset \mathbb{R} \to \mathbb{R}$ is a *univariate intrinsic function* if, for any $A \in \mathbb{I}B$, where $\mathbb{I}B = \{X \in \mathbb{IR} : X \subset B\}$, the following are

known and can be evaluated computationally:

1. an interval extension of u on A that is an inclusion function of u on A,

2. a concave relaxation of u on A,

3. a convex relaxation of u on A.

Definition 2.3.4 (Factorable Function [271])**.** A function $\mathcal{F} : Z \subset \mathbb{R}^n \to \mathbb{R}$ is *factorable* if it can be expressed in terms of a finite number of factors $v_1, \ldots, v_m$, such that given $\mathbf{z} \in Z$, $v_i = z_i$ for $i = 1, \ldots, n$, and v_k is defined for $n \leq k \leq m$ as either

1. $v_k = v_i + v_j$, with $i, j < k$, or

2. $v_k = v_i v_j$, with $i, j < k$, or

3. $v_k = u_k(v_i)$, with $i < k$, where $u_k : B_k \to \mathbb{R}$ is a univariate intrinsic function,

and $\mathcal{F}(\mathbf{z}) = v_m(\mathbf{z})$, for every $\mathbf{z} \in Z$. A vector-valued function is factorable if each of its components are factorable functions.

The concept of a factorable function, formalized in Definition 2.3.4, is general enough to incorporate the most common algebraic expressions such as ($\div$, $-$, log, and sin). For example, the function $f(z_1, z_2) = (\exp(z_1) - \frac{z_1 z_2}{z_1 + 2})^2$ on $\mathbf{z} \in [1, 2] \times [2, 3]$ can then be written as the following list of factors.

$$
\begin{aligned}
v_1 &= z_1 & v_5 &= v_1 + 2 \\
v_2 &= z_2 & v_6 &= v_4 / v_5 \\
v_3 &= \exp(v_1) & v_7 &= v_3 - v_6 \\
v_4 &= v_1 v_2 & v_8 &= v_7^2
\end{aligned}
$$

This results in a new formulation that contains only nonlinear unary and binary terms that are in a library of intrinsic functions. For each term present in the formulation, relaxations and bounds may be readily computed [169, 177, 237, 292]. Interval bounds for each auxiliary variable introduced are determined through interval constraint propagation. Consider the problem detailed in Example 2.3.6,

$$
\begin{aligned}
f^* = \quad & \min_{\mathbf{z} \in [1,2], \eta} \eta \\
& \text{s.t.} \;\; z + \exp(z) z \leq \eta \\
& \qquad z / \log(z) \leq 1
\end{aligned}
\tag{2.3.6}
$$

An equivalent program formed using AVM leads to

$$
\begin{aligned}
f^* = \quad & \min_{\mathbf{z}\in[1,2],\mathbf{v},\eta} \eta \\
\text{s.t.} \quad & v_1 = \exp(z) \\
& v_2 = z \times v_1 \\
& v_3 = z + v_2 \\
& v_4 = \log(z) \\
& v_5 = 1/v_4 \\
& v_6 = z \times v_5 \\
& v_3 \leq \eta \\
& v_6 \leq 1.
\end{aligned}
\tag{2.3.7}
$$

Nonlinear terms may then relaxed. Bilinear terms are typically replaced with polyhedral envelopes developed independently by McCormick [177] and Al-Khayyal [8] and colleagues. Fractional terms $v_k = v_i/v_j$ are rewritten as the equivalent $v_i = v_k v_j$ [304] and then treated as bilinear expressions. Convex and concave relaxations of univariate terms are computed based on known convexity properties [177] and subsequently linearized. The reader is directed to [177] and [259] for succinct definitions of the envelopes of convex, concave, and convexoconcave functions.

The introduction of numerous auxiliary variables reduces the complexity of underlying constraints and enables the use of a number of efficient subroutines. For instance, the sandwich algorithm can efficiently compute linearization points for unary convex functions [304] where approaches for n-ary functions requires the solutions of optimization problems or lack formal guarantees of convergence [210]. Subsequent works have expanded on this approach in a number of ways. Mixed-integer linear relaxations of unary functions and bilinear forms that exploit the

dramatic improvement of MIP solver efficiency in the past years have be detailed and were shown to significantly improvement solution time for challenging problems [199, 301, 302]. Additional efforts have focused on general model structure and potentially higher arity functions to efficiently compute tight relaxations, such as the work of Khajavirad et al. [150] on convex transformable intermediates.

Approaches such as the reformulation-linearization technique [281] address specialized forms such as nonconvex QP or pooling problems [183, 184, 185] begin with the introduction of auxiliary variables followed by symbolic rearrangements and simplifications. In some cases, the use of auxiliary variables may lead to tighter and more efficient cutting planes than treating the problem in the original decision space [186]. This approach is particularly powerful in the case of 0-1 mixed-integer formulations were the application of arithmetic identities for 0-1 variables, such as $z^2 = z$, significantly simplify expressions containing products. Auxiliary variable methods suffer from two distinct drawbacks. First, the programs addressed must be factorable, which substantially limits the applicability of this method to classes of implicitly-defined function [300] such as those which naturally arise from dynamic formulations [123]. Additionally, B&B algorithms that underlie all deterministic nonconvex optimization algorithms suffer from worst-case exponential run times, and as such, the introduction of a large number of variables may dramatically increase solve time [136]. It is for these reasons, this thesis focuses on the less explored reduced-space optimization methodologies.

2.3.2 αBB Approaches

A popular family of reduced-space methods used to compute convex relaxations is that of αBB [5, 6, 13]. These approaches compute convex relaxations for twice-continuously differential functions $f : \mathbb{R}^n \to \mathbb{R}$ by adding a convex quadratic term $\alpha \mathbf{x}^T \mathbf{x}$ to $f(\mathbf{x})$ wherein the resulting sum

is convex. Unlike AVM-based approaches, the underlying relaxation is computed in the same decision space as the original function $f(\mathbf{x})$. Moreover, these relaxations exhibit a quadratic convergence order under mild assumptions [44]; however, the relaxations computed may be overly-conservative and computationally expensive. The primary difficulty associated with applying this method lies in the determination of α. As the determination of the α value which leads to the tightest relaxation would require the determination of a minimal eigenvalue of the Hessian matrix associated with $f(\mathbf{x})$, a nonconvex optimization problem, approximations based on interval arithmetic are commonly used [5, 6]. A small body of work [7, 134, 182, 217, 288, 289] has focused on improving the initial Gerschgorin theorem [110] and interval matrix eigenvalue solution methodologies used to calculate α values.

The αBB method has been used in applications ranging from protein folding [105] to reactor network synthesis [94]. This method forms the basis of general nonlinear relaxation methods used by commercial MINLP solvers such as ANTIGONE [185]. Additional recent investigations have focused on combining αBB with transformations, such as that of μ-subenergy [147]. The αBB approach was subsequently adapted to allow for the construction of edge-concave underestimators for nonconvex functions and in turn extended to dynamic optimization [19, 124]. However, the identification of less conservative, less expensive, and more generally applicable methods for computing convex relaxations remains active areas of research within the global optimization community that motivates interest in the development of alternative relaxation approaches.

2.3.3 McCormick Relaxations

A general method for constructing relaxations of arbitrary factorable functions was first presented by Garth McCormick in 1976 [177]. In the past decade, a significant effort was made to further generalize this approach beginning with the development of a general operator-overloading

approach for constructing McCormick relaxations [191] analogous to automatic differentiation [23]. The rules for composition, addition, and multiplication are reviewed in Propositions 2.3.6 - 2.3.8. Relaxations constructed in this manner are no weaker than interval bounds (and often significantly tighter) [191].

Definition 2.3.5 (Cumulative Mapping [271])**.** Let the *cumulative mapping* v_k be the mapping $v_k : Z \to \mathbb{R}$ defined for each $\mathbf{z} \in Z$ by the value $v_k(\mathbf{z})$ when the factors of $\mathcal{F}$ are computed recursively, as per Definition 2.3.4, beginning from $\mathbf{z}$.

Proposition 2.3.6 (Univariate McCormick Composition Rule[191])**.** Let $Z \subset \mathbb{R}^n$ and $X \subset \mathbb{R}$ be nonempty convex sets. Consider the composite function $w = \phi \circ q$ where $w : Z \to \mathbb{R}$ is continuous, $\phi : X \to \mathbb{R}$, let $q(Z) \subset X$. Let $q^{cv} : Z \to \mathbb{R}$ and $q^{cc} : Z \to \mathbb{R}$ be convex and concave relaxations of q on Z, respectively. Let $\phi^{cv} : X \to \mathbb{R}$ and $\phi^{cc} : X \to \mathbb{R}$ be convex and concave relaxations of ϕ on X, respectively. Let $\xi^*_{\min} \in X$ be a point at which ϕ^{cv} attains its infimum on X and let $\xi^*_{\max} \in X$ be a point at which ϕ^{cc} attains its supremum on X. Then the convex and concave relaxations are, respectively, given by

$$w^{cv} : Z \to \mathbb{R} : \mathbf{z} \mapsto \phi^{cv}(\mathrm{mid}(q^{cv}(\mathbf{z}), q^{cc}(\mathbf{z}), \xi^*_{\min})) \tag{2.3.8}$$

$$w^{cc} : Z \to \mathbb{R} : \mathbf{z} \mapsto \phi^{cc}(\mathrm{mid}(q^{cv}(\mathbf{z}), q^{cc}(\mathbf{z}), \xi^*_{\max})). \tag{2.3.9}$$

In the above, the $\mathrm{mid}(\cdot, \cdot, \cdot)$ function takes the median value of its three arguments. Generally, the application of Proposition 2.3.6 requires the use of closed-form expressions, which are available for all standard operations (e.g. +, ×, exp, log), to determine the values of $\xi^*_{\max}$ and $\xi^*_{\min}$ in conjunction with defined forms of the relaxations ϕ^{cv} and ϕ^{cc}. Convex and concave envelopes of univariate intrinsic functions (such as exp and tanh) are available in existing relaxation libraries (e.g. [329]) and are used throughout this thesis for calculations unless

otherwise specified. In the case of "×", the rules presented in [191] are used for the calculation of relaxations and the max operator is addressed using the standard reformulation $\max(x, y) = (x + y + |x - y|)/2$.

Proposition 2.3.7 (McCormick Multiplication Rule [191])**.** Let $Z \subset \mathbb{R}^n$ be a nonempty convex set. Let $w, x_1, x_2 : Z \to \mathbb{R}$ such that $w(\mathbf{z}) = x_1(\mathbf{z})x_2(\mathbf{z})$. Let $x_1^{cv} : Z \to \mathbb{R}$ and $x_1^{cc} : Z \to \mathbb{R}$ be convex and concave relaxations of x_1 on Z, respectively. Let $x_2^{cv} : X \to \mathbb{R}$ and $x_2^{cc} : X \to \mathbb{R}$ be convex and concave relaxations of x_2 on Z, respectively. Further, let $x_1^L, x_1^U, x_2^L, x_2^U \in \mathbb{R}$ be bounds on x_1, x_2 such that

$$x_1^L \leq x_1(\mathbf{z}) \leq x_1^U \qquad \text{and} \qquad x_2^L \leq x_2(\mathbf{z}) \leq x_2^U, \qquad \forall \mathbf{z} \in Z.$$

Let the following intermediate functions $\alpha_1, \alpha_2, \beta_1, \beta_2, \gamma_1, \gamma_2, \delta_1, \delta_2 : Z \to \mathbb{R}$ be defined as

$$\begin{aligned}
\alpha_1(\cdot) &= \min\{x_2^L x_1^{cv}(\cdot), x_2^L x_1^{cc}(\cdot)\}, & \alpha_2(\cdot) &= \min\{x_1^L x_2^{cv}(\cdot), x_1^L x_2^{cc}(\cdot)\}, \\
\beta_1(\cdot) &= \min\{x_2^U x_1^{cv}(\cdot), x_2^U x_1^{cc}(\cdot)\}, & \beta_2(\cdot) &= \min\{x_1^U x_2^{cv}(\cdot), x_1^U x_2^{cc}(\cdot)\}, \\
\gamma_1(\cdot) &= \max\{x_2^L x_1^{cv}(\cdot), x_2^L x_1^{cc}(\cdot)\}, & \gamma_2(\cdot) &= \max\{x_1^U x_2^{cv}(\cdot), x_1^U x_2^{cc}(\cdot)\}, \\
\delta_1(\cdot) &= \max\{x_2^U x_1^{cv}(\cdot), x_2^U x_1^{cc}(\cdot)\}, & \delta_2(\cdot) &= \max\{x_1^L x_2^{cv}(\cdot), x_1^L x_2^{cc}(\cdot)\}.
\end{aligned}$$

Then, convex and concave relaxations of w on Z are given by $w_{\times,0}^{cv}$ and $w_{\times,0}^{cc}$,

$$\begin{aligned}
w_{\times,0}^{cv} &: Z \to \mathbb{R} : \mathbf{z} \mapsto \max\{\alpha_1(\mathbf{z}) + \alpha_2(\mathbf{z}) - x_1^L x_2^L, \beta_1(\mathbf{z}) + \beta_2(\mathbf{z}) - x_1^U x_2^U\} \\
w_{\times,0}^{cc} &: Z \to \mathbb{R} : \mathbf{z} \mapsto \min\{\gamma_1(\mathbf{z}) + \gamma_2(\mathbf{z}) - x_1^U x_2^L, \delta_1(\mathbf{z}) + \delta_2(\mathbf{z}) - x_1^L x_2^U\},
\end{aligned}$$

respectively.

Proposition 2.3.8 (McCormick Addition Rule [191])**.** Let $Z \subset \mathbb{R}^n$, and $w, q, r : Z \to \mathbb{R}$ such that

$w(\mathbf{z}) = q(\mathbf{z}) + r(\mathbf{z})$. Let $q^{cv}, q^{cc} : Z \to \mathbb{R}$ be convex and concave relaxations of q on Z, respectively. Similarly, let $r^{cv}, r^{cc} : Z \to \mathbb{R}$ be convex and concave relaxations of r on Z, respectively. Then, $w^{cv}, w^{cc} : Z \to \mathbb{R}$, such that

$$w^{cv}(\mathbf{z}) = q^{cv}(\mathbf{z}) + r^{cv}(\mathbf{z}), \qquad w^{cc}(\mathbf{z}) = q^{cc}(\mathbf{z}) + r^{cc}(\mathbf{z}), \tag{2.3.10}$$

are, respectively, a convex and concave relaxation of w on Z.

The subgradients of McCormick relaxations of factorable functions may be computed via the forward-mode automatic differentiation approach detailed in [191]. The relevant theorem for multiplication is included in Theorem 2.3.9.

Theorem 2.3.9 (Multiplication Rule for Subgradients [191])**.** Suppose that $Z \subset \mathbb{R}^n$ is a nonempty convex set, and $\mathbf{z} \in Z$. Let $w, x_1, x_2 : Z \to \mathbb{R}$ such that $w(\mathbf{z}) = x_1(\mathbf{z})x_2(\mathbf{z})$. Let $x_1^{cv} : Z \to \mathbb{R}$ and $x_1^{cc} : Z \to \mathbb{R}$ be convex and concave relaxations of x_1 on Z, respectively. Let $x_2^{cv} : X \to \mathbb{R}$ and $x_2^{cc} : X \to \mathbb{R}$ be convex and concave relaxations of x_2 on Z, respectively. Further, let $x_1^L, x_1^U, x_2^L, x_2^U \in \mathbb{R}$ be bounds on x_1, x_2 such that

$$x_1^L \leq x_1(\mathbf{z}) \leq x_1^U \qquad \text{and} \qquad x_2^L \leq x_2(\mathbf{z}) \leq x_2^U, \qquad \forall \mathbf{z} \in Z.$$

Let the intermediate functions $\alpha_1, \alpha_2, \beta_1, \beta_2, \gamma_1, \gamma_2, \delta_1, \delta_2 : Z \to \mathbb{R}$ and convex/concave relaxations $w_{\times,0}^{cv}, w_{\times,0}^{cc} : Z \to \mathbb{R}$ be defined as in Proposition 2.3.7. Then the subgradients of

$\alpha_1, \alpha_2, \beta_1, \beta_2, \gamma_1, \gamma_2, \delta_1, \delta_2$ at $\bar{\mathbf{z}} \in Z$ are respectively given by:

$$\mathbf{s}_{\alpha_1}(\bar{\mathbf{z}}) = \begin{cases} x_2^L \mathbf{s}_{x_1}^{cv}(\bar{\mathbf{z}}) & \text{if } x_2^L \geq 0, \\ x_2^L \mathbf{s}_{x_1}^{cc}(\bar{\mathbf{z}}) & \text{otherwise,} \end{cases} \qquad \mathbf{s}_{\alpha_2}(\bar{\mathbf{z}}) = \begin{cases} x_1^L \mathbf{s}_{x_2}^{cv}(\bar{\mathbf{z}}) & \text{if } x_1^L \geq 0, \\ x_1^L \mathbf{s}_{x_2}^{cc}(\bar{\mathbf{z}}) & \text{otherwise,} \end{cases} \tag{2.3.11}$$

$$\mathbf{s}_{\beta_1}(\bar{\mathbf{z}}) = \begin{cases} x_2^U \mathbf{s}_{x_1}^{cv}(\bar{\mathbf{z}}) & \text{if } x_2^U \geq 0, \\ x_2^U \mathbf{s}_{x_1}^{cc}(\bar{\mathbf{z}}) & \text{otherwise,} \end{cases} \qquad \mathbf{s}_{\beta_2}(\bar{\mathbf{z}}) = \begin{cases} x_1^U \mathbf{s}_{x_2}^{cv}(\bar{\mathbf{z}}) & \text{if } x_1^U \geq 0, \\ x_1^U \mathbf{s}_{x_2}^{cc}(\bar{\mathbf{z}}) & \text{otherwise,} \end{cases} \tag{2.3.12}$$

$$\mathbf{s}_{\gamma_1}(\bar{\mathbf{z}}) = \begin{cases} x_2^L \mathbf{s}_{x_1}^{cc}(\bar{\mathbf{z}}) & \text{if } x_2^L \geq 0, \\ x_2^L \mathbf{s}_{x_1}^{cv}(\bar{\mathbf{z}}) & \text{otherwise,} \end{cases} \qquad \mathbf{s}_{\gamma_2}(\bar{\mathbf{z}}) = \begin{cases} x_1^U \mathbf{s}_{x_2}^{cc}(\bar{\mathbf{z}}) & \text{if } x_1^U \geq 0, \\ x_1^U \mathbf{s}_{x_2}^{cv}(\bar{\mathbf{z}}) & \text{otherwise,} \end{cases} \tag{2.3.13}$$

$$\mathbf{s}_{\delta_1}(\bar{\mathbf{z}}) = \begin{cases} x_2^U \mathbf{s}_{x_1}^{cc}(\bar{\mathbf{z}}) & \text{if } x_2^U \geq 0, \\ x_2^U \mathbf{s}_{x_1}^{cv}(\bar{\mathbf{z}}) & \text{otherwise,} \end{cases} \qquad \mathbf{s}_{\delta_2}(\bar{\mathbf{z}}) = \begin{cases} x_1^L \mathbf{s}_{x_2}^{cc}(\bar{\mathbf{z}}) & \text{if } x_1^L \geq 0, \\ x_1^L \mathbf{s}_{x_2}^{cv}(\bar{\mathbf{z}}) & \text{otherwise,} \end{cases} \tag{2.3.14}$$

where $\mathbf{s}_{x_1}^{cv}$, $\mathbf{s}_{x_1}^{cv}$, $\mathbf{s}_{x_1}^{cv}$, $\mathbf{s}_{x_1}^{cv}$, are, respectively, subgradients of x_1^{cv}, x_1^{cc}, x_2^{cv}, x_2^{cc} on Z at $\bar{\mathbf{z}} \in Z$. Finally, the subgradients $\mathbf{s}_{w_{\times,0}}^{cv}$, $\mathbf{s}_{w_{\times,0}}^{cc}$ of $w_{\times,0}^{cv}$, $w_{\times,0}^{cc}$ on Z at $\bar{\mathbf{z}} \in Z$ are, respectively, given by:

$$\mathbf{s}_{w_{\times,0}}^{cv}(\bar{\mathbf{z}}) = \begin{cases} \mathbf{s}_{\alpha_2}(\bar{\mathbf{z}}) + \mathbf{s}_{\alpha_2}(\bar{\mathbf{z}}), & \text{if } \alpha_1(\bar{\mathbf{z}}) + \alpha_2(\bar{\mathbf{z}}) - x_1^L x_2^L \geq \beta_1(\bar{\mathbf{z}}) + \beta_2(\bar{\mathbf{z}}) - x_1^U x_2^U, \\ \mathbf{s}_{\beta_1}(\bar{\mathbf{z}}) + \mathbf{s}_{\beta_2}(\bar{\mathbf{z}}), & \text{otherwise,} \end{cases} \tag{2.3.15}$$

$$\mathbf{s}_{w_{\times,0}}^{cc}(\bar{\mathbf{z}}) = \begin{cases} \mathbf{s}_{\gamma_1}(\bar{\mathbf{z}}) + \mathbf{s}_{\gamma_2}(\bar{\mathbf{z}}), & \text{if } \gamma_1(\bar{\mathbf{z}}) + \gamma_2(\bar{\mathbf{z}}) - x_1^U x_2^L \geq \delta_1(\bar{\mathbf{z}}) + \delta_2(\bar{\mathbf{z}}) - x_1^L x_2^U \\ \mathbf{s}_{\delta_1}(\bar{\mathbf{z}}) + \mathbf{s}_{\delta_2}(\bar{\mathbf{z}}), & \text{otherwise.} \end{cases} \tag{2.3.16}$$

For details relating to computing relaxations and associated subgradients of factors using other functional forms, the reader is referred to [191]. For convex/concave relaxations $v_k^{cv}, v_k^{cc} : Z \to \mathbb{R}$ computed through the recursive application of these rules to each factor v_k, the factorable function $\mathcal{F}$ also constitutes a cumulative mapping.

Theoretical developments for constructing convex/concave composite relaxations from arbitrary convex and concave functions were established [271]. Namely, a composite function of the form $\phi(\mathbf{f}(\cdot))$ on Z, where the outer function ϕ is factorable, but the inner function $\mathbf{f}$ may not be relaxed using the standard McCormick relaxation technique. However, provided that convex and concave relaxations of $\mathbf{f}(\cdot)$ on Z are known, the *g*eneralized McCormick relaxation approach forms the framework by which convex and concave relaxations of $\phi(\mathbf{f}(\cdot))$ on Z may be recursively computed. Throughout this thesis, the terms generalized McCormick relaxation and McCormick relaxation will be used interchangeably with the "generalized" descriptor often omitted for conciseness.

A subsequent theory that extended composition rules to multivariate functions was developed that resulted in the formulation of tighter composition rules involving multiplication and maxima operators [207, 310]. A further refinement by Khan and colleagues [151, 152] produced an alternative methodology with theoretical guarantees of differentiability. McCormick relaxations were extended to allow for the global optimization of Gaussian process models [263, 267]. Additional works have focused on developing improvements to functional forms that appear in numerous model formulations including that of the IAPWS-IF97 property models for steam [50] as well as additional thermodynamic property and cost models [209].

One distinct feature of McCormick relaxations is that they exhibit quadratic pointwise convergence properties under mild assumptions [44, 146, 203, 205]. Quadratic convergence can substantially mitigate clustering about global minimizers during B&B; a key factor that often limits solver performance [323]. Moreover, the use of tighter interval bounds has been noted as a key contributing factor by which positive offset may be reduced for McCormick relaxations and convergence rates further accelerated [205].

Numerous extensions of the generalized McCormick relaxation framework have been

proposed, including the use of Taylor-McCormick models [45, 255]. Subsequently, Wechsung and colleagues described a method of propagating McCormick relaxations backwards on a directed acyclic graph [324]. A method of tightening interval bounds obtained through forward and reverse interval propagation using the subgradient information associated with convex/concave relaxations was detailed [206]. Furthermore, methods by which convex/concave relaxations of implicitly-defined functions have been described [300] and even approaches for evaluating relaxations of the expected values of continuous random variables have been created [275]. More recently, a number of key global optimization subroutines, such as linearization [210], have been extended to efficiently make use of McCormick relaxations.

Tight relaxations formed in reduced-space have contributed to the outstanding performance of the McCormick relaxation approach in numerous applications with simulations embedded. A series of works on process flow sheet optimization in the energy sector were detailed in which the McCormick relaxation approach outperformed commercial solvers implementing the auxiliary variable approach [47, 48, 49]. Similar work on heat exchanger network design and integration has demonstrated additional computational benefits [98]. Moreover, preliminary investigations into the global optimization of artificial neural networks in reduced-space [264] and the application of these methods to hybrid surrogate models for flash calculations [265] and thermodynamic models [266], have further demonstrated the outstanding computational performance of McCormick relaxations when the decision variables that naturally participate in a formulation are greatly outnumbered by intermediate terms and state variables.

This thesis advances the McCormick relaxation methodology in three substantial ways. First, a performant and extendable open-source global optimizer, EAGO.jl, was created that uses the McCormick methodology to construct relaxations of arbitrary nonlinear functions, and is detailed in Chapter 3. In Chapter 4, the advances in constructing convex relaxations of the

activation functions participating in artificial neural networks are detailed. In Chapter 5, novel approaches for computing less conservative relaxations of functions with intermediate bilinear terms are described. Moreover, additional contributions were made pertaining to global dynamic optimization.

2.4 Dynamic Optimization

Dynamic optimization plays a key role in process startup and shutdown, the control of batch and semi-batch processes, and the verification of select neural network architectures. Locally optimal solutions of dynamic optimization problems can be readily attained using multiple shooting methods [43] or the state-of-the-art orthogonal collocation on finite elements method (also known as direct transcription) [39]. However, the incorporation of dynamics introduces a number of challenges due to the inclusion of differential equations, such as the handling of stiff systems, scaling issues, and the selection of discretization parameters, along with ever present concerns about accuracy and stability. Further difficulties arise when deterministic solutions of dynamic optimization problems are required; as is the case of safety-critical processes. Namely, rigorous and tight bounds on the solution trajectories must be calculated. This remains an active area of research and the dynamic optimization portion of this thesis focuses on our recent advances of these methods.

The first deterministic global methods for solving dynamic optimization problems used the αBB to construct convex relaxations [3, 4, 229] or McCormick relaxations [177, 282, 283]. Generally applicable methods for bounding solutions parametric partial differential equations, delay differential equations, and other dynamic systems currently do not exist. In order to simplify the analysis, we consider the nonconvex dynamic optimization problem given below in

(2.4.1a) - (2.4.1d):

$$\phi^* = \min_{\mathbf{p} \in P \subset \mathbb{R}^{n_p}} \phi(\mathbf{x}(\mathbf{p}, t_f), \mathbf{p}) \tag{2.4.1a}$$

$$\text{s.t. } \dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}(\mathbf{p}, t), \mathbf{p}, t), \quad \forall t \in T \subset I = [t_0, t_f] \tag{2.4.1b}$$

$$\mathbf{x}(\mathbf{p}, t_0) = \mathbf{x}_0(\mathbf{p}) \tag{2.4.1c}$$

$$\mathbf{g}(\mathbf{x}(\mathbf{p}, t), \mathbf{p}) \leq \mathbf{0}, \quad \forall t \in T \subset I \tag{2.4.1d}$$

where T is finite. The dynamic optimization problem (2.4.1a)-(2.4.1d) embeds a parametric system of ordinary differential equations. We abbreviate a parametric ordinary differential equation (singular) as pODE and a parametric ordinary system of differential equations (plural) as pODEs.

2.4.1 Parametric Ordinary Differential Equations

A system of pODEs is formalized as

$$\begin{aligned} \dot{\mathbf{x}}(\mathbf{p}, t) = \frac{d\mathbf{x}}{dt}(\mathbf{p}, t) = \mathbf{f}(\mathbf{x}(\mathbf{p}, t), \mathbf{p}, t), \quad t \in T \subset I = [t_0, t_f], \ \mathbf{p} \in P \\ \mathbf{x}(\mathbf{p}, t_0) = \mathbf{x}_0(\mathbf{p}), \quad \mathbf{p} \in P, \end{aligned} \tag{2.4.2}$$

with mappings $\mathbf{f} : D \times \Pi \times T \to \mathbb{R}^{n_x}$ and $\mathbf{x}_0 : P \to D$, with $D \subset \mathbb{R}^{n_x}$, $\Pi \subset \mathbb{R}^{n_p}$, and $T \subset \mathbb{R}$ open with $P \in \mathbb{I}\Pi$. A pODEs is well-posed provided that Assumption 2.4.1 holds.

Assumption 2.4.1. The pODE-IVP system (2.4.2) satisfies the following conditions:

1. $\mathbf{x}_0 : P \to D$ is locally Lipschitz continuous on P.
2. $\mathbf{f}$ is continuously differentiable on $D \times \Pi \times T$.

A solution of (2.4.2) is any continuous $\mathbf{x} : P \times I \to D$ such that, for every $\mathbf{p} \in P$, $\mathbf{x}(\mathbf{p}, \cdot) : T \to D$ is continuously differentiable and satisfies (2.4.2) on I. Further, it is assumed that a unique solution exists over the time domain I for every $\mathbf{p} \in P$.

Several approaches have been developed by which convex/concave relaxations of the solutions of pODEs may be computed [122, 123, 270, 272, 293] and parametric differential algebraic equation systems (pDAEs) [270]. In general, pDAEs remain a challenging class of problems as current approaches to constructing relaxations of implicit algebraic equations rely on the underlying an existence and uniqueness criteria and expansiveness of the state relaxations often leads to ill-posed problems due to the expansiveness of state variable enclosures. While this remains a fundamental limitation for broadly relaxing pDAEs systems, many index-1 pDAEs may be reformulated as pODEs and addressed with general methods. As such, the pODE approaches discussed here have broad applicability. Most approaches for computing convex/concave relaxations originate from adaptions of methods by which state bounds may be determined.

Definition 2.4.1 (Adapted from [270])**.** Continuous functions $\mathbf{x}^L, \mathbf{x}^U : T \to \mathbb{R}^{n_x}$ are called *state bounds* for (2.4.2) on $T \times P$ if $\mathbf{x}^L(t) \leq \mathbf{x}(\mathbf{p}, t) \leq \mathbf{x}^U(t)$, $\forall (\mathbf{p}, t) \in P \times T$.

Definition 2.4.2. Continuous functions $\mathbf{x}^{cv}, \mathbf{x}^{cc} : T \to \mathbb{R}^{n_x}$ are called *state relaxations* for (2.4.2) on $T \times P$ if $\mathbf{x}^{cv}(\mathbf{p}, t) \leq \mathbf{x}(\mathbf{p}, t) \leq \mathbf{x}^{cc}(\mathbf{p}, t)$, $\forall (\mathbf{p}, t) \in P \times T$, provided that $\mathbf{x}^{cv}$, $\mathbf{x}^{cc}$ are, respectively, convex and concave on P, $\forall t \in T$.

Generally, methods for computing state relaxations can be divided into two categories: discrete-time relaxation methods (also called discretize-then-relax methods) [256] and continuous-time relaxation methods (also called relax-then-discretize) [270] in which an auxiliary pODE system is constructed that describes convex and concave relaxations. The auxiliary pODEs may then be integrated using a modern numerical solution method such as SUNDIALS [274]. The alternative discrete-time approach discretizes the pODEs into set of algebraic expressions,

and then computes state relaxations at each discrete time point.

2.4.2 Continuous-Time Relaxation Methods

A general nonlinear method, the differential-inequality approach, which furnishes convex/concave relaxations originated with Singer and Barton [285]. This approach that leads to *relaxation amplifying dynamics* (RAD), in which the overestimation of the state variables grow monotonically in time, was subsequently refined to a *relaxation preserving dynamics* (RPD) approach [270]. The RPD methodology generates relaxations with less overestimation without this undesirable monotonicity property by exploiting the continuity of the underlying solution. As an immediate consequence, relaxations at this point of crossing can be evaluated and are generally more restrictive. This is formalized in Definition 2.4.3 and Proposition 2.4.4.

Definition 2.4.3 (Adapted from [270])**.** For every $i \in \{1, \ldots, n_x\}$, we define $\mathcal{R}_i^{cv}, \mathcal{R}_i^{cc} : \mathbb{R}^{n_x} \times \mathbb{R}^{n_x} \to \mathbb{R}^{n_x} \times \mathbb{R}^{n_x}$ by $\mathcal{R}_i^{cv}(\phi, \psi') = (\phi, \psi')$, where $\psi'_k = \psi_k$ if $k \neq i$ and $\psi'_i = \phi_i$, and $\mathcal{R}_i^{cc}(\phi, \psi) = (\phi', \psi)$, where $\phi'_k = \phi_k$ if $j \neq i$ with $\phi'_i = \psi_i$.

Proposition 2.4.4. Suppose that state bounds $x^L, x^U : T \to \mathbb{R}^n$ are available, and let

$u_i(\cdot,\cdot,\cdot,\cdot) = f_i^{cv}(\mathcal{R}_i^{cv}(\cdot,\cdot),\cdot,\cdot)$ and $o_i(\cdot,\cdot,\cdot,\cdot) = f_i^{cc}(\mathcal{R}_i^{cc}(\cdot,\cdot),\cdot,\cdot)$. The auxiliary pODEs-IVP given by

$$\dot{x}_i^{cv}(\mathbf{p},t) = \begin{cases} u_i(\mathbf{x}^{cv}(\mathbf{p},t),\mathbf{x}^{cc}(\mathbf{p},t),\mathbf{p},t), & \text{if } \mathbf{x}^{cv}(\mathbf{p},t) \in [\mathbf{x}_i^L(t),\mathbf{x}_i^U(t)] \\ \max\left(\dot{x}_i^L(t), u_i(\mathbf{x}^{cv}(\mathbf{p},t),\mathbf{x}^{cc}(\mathbf{p},t),\mathbf{p},t)\right), & \text{if } \mathbf{x}^{cv}(\mathbf{p},t) < \mathbf{x}_i^L(t) \\ \min\left(\dot{x}_i^U(t), u_i(\mathbf{x}^{cv}(\mathbf{p},t),\mathbf{x}^{cc}(\mathbf{p},t),\mathbf{p},t)\right), & \text{if } \mathbf{x}^{cv}(\mathbf{p},t) > \mathbf{x}_i^U(t) \end{cases} \tag{2.4.3}$$

$$\dot{x}_i^{cc}(\mathbf{p},t) = \begin{cases} o_i(\mathbf{x}^{cv}(\mathbf{p},t),\mathbf{x}^{cc}(\mathbf{p},t),\mathbf{p},t), & \text{if } \mathbf{x}^{cc}(\mathbf{p},t) \in [\mathbf{x}_i^L(t),\mathbf{x}_i^U(t)] \\ \max\left(\dot{x}_i^L(t), o_i(\mathbf{x}^{cv}(\mathbf{p},t),\mathbf{x}^{cc}(\mathbf{p},t),\mathbf{p},t)\right), & \text{if } \mathbf{x}^{cc}(\mathbf{p},t) < \mathbf{x}_i^L(t) \\ \min\left(\dot{x}_i^U(t), o_i(\mathbf{x}^{cv}(\mathbf{p},t),\mathbf{x}^{cc}(\mathbf{p},t),\mathbf{p},t)\right), & \text{if } \mathbf{x}^{cc}(\mathbf{p},t) > \mathbf{x}_i^U(t) \end{cases} \tag{2.4.4}$$

$$x_i^{cv}(\mathbf{p},t_0) = \max\left(x_i^L(t_0), x_{i,0}^{cv}(\mathbf{p})\right) \tag{2.4.5}$$

$$x_i^{cc}(\mathbf{p},t_0) = \min\left(x_i^U(t_0), x_{i,0}^{cc}(\mathbf{p})\right) \tag{2.4.6}$$

for each $i = 1,\ldots,n_x$ fully specify state relaxations $\mathbf{x}^{cv},\mathbf{x}^{cc} : P \to X \subset \mathbb{R}^n$.

In practice, the ODE system with a discontinuous right-hand-side presented in Proposition 2.4.4, is a hybrid system that can be solved accordingly. The method employed to evaluate relaxations in the initial work of [270] introduced boolean indicator variables used in an event detection scheme to select the appropriate behavior of the right-hand side function. Recent efforts have shown, under mild certain assumptions, that valid convex/convex relaxations may be achieved when empty McCormick relaxation objects are encountered reducing Proposition 2.4.4 to a standard ODE system. Recent work has shown that the RPD method inherits second-order convergence properties under mild assumptions [261]. This approach was generalized to an optimization-based formulation by Song and Khan [293] that is at least as tight as the RPD formulation.

A series of similar adaptations were presented in Harwood [122, 123] which derive

polyhedral and affine relaxation of state variables. In [277], use of model invariants that introduced redundant expressions, was shown to tighten the reachable set enclosure calculated using differential-inequalities. A discrete-time adaptation of the RPD approach was described, in which certain monotonicity properties were satisfied [337] as a means by which tight and accurate bounds on uncertainty could be determined; both desirable properties in a dynamic global optimization context.

2.4.3 Discrete-Time Relaxation Methods

The preponderance of previous investigations into discrete-time relaxation methodologies focused on explicit approaches [191, 255, 256, 258]. Early works explored relaxation of standard numerical solution routines for (2.4.2) such as Explicit Euler [191] and a preliminary investigation into the Implicit Euler method [300]. Approaches were extended to a discretize-then-relax framework in which a two-step process is used to construct state relaxations [256]. The first-step of this method uses an existence and uniqueness test to establish state relaxations for the entire time interval $[t_i, t_{i+1}]$ for a single step. This is followed by a second contractor step that refines state relaxations at the new time t_{i+1}. These methods find their origin in analogous interval methods [33, 163, 213, 214]. A further adaptation focused on representing the uncertainty set as a Taylor model. These Taylor models use a variety of different set-valued arithmetics to enclose the remainder term [34, 138, 161, 214, 315]. However, a thorough investigation of implicit methods for evaluating state relaxations has never been conducted. Even investigations into implicit parametric interval methods have been limited.

Implicit parametric interval methods are particularly promising in that the strong stability preserving properties can substantially reduce the number of discretization points required. In [247], the use of a parametric interval Newton method to bound solution sets of pODEs was

investigated. Marciniak and collaborators make a series of advances to constructing interval bounds of implicit integrator forms: first, by providing multistep analogs of validated relaxations of the Nyström and Milne-Simpson types [170], then by analyzing the implicit Adams-Moulton forms [171], and finally by providing a predictor-corrector method that makes use of both explicit and implicit forms [174]. Marciniak [172] also adapted these methods to construct valid relaxations of a parallel Crank-Nicholson algorithm to bound parabolic PDEs. However, these investigations of implicit interval methods are primarily limited to considering direct interval extensions of the underlying numerical method. As a consequence, significant improvements may be achieved by considering mean-valued forms of the integration methods and more advanced numerical methods.

This thesis introduces novel second-order parametric implicit linear multistep methods for the computation of state relaxations of numerical solutions of (2.4.2). These are subsequently extended to rigorously account for truncation error. An analysis of alternative mean-value forms was also performed along with a comparison to Hermite-Obreschkoff-based relaxations. Furthermore, these approaches are integrated into the global optimizer EAGO [327], through the use of the DynamicBounds.jl abstraction layer developed as part of this thesis.

2.5 Robust Optimization

Nominal designs may dramatically fail as they do not account for the influence of parametric uncertainty at the design stage. These failures manifest as unsatisfied quality specifications or disastrous equipment damage when operating subject to uncertainty. The two most common approaches that account for underlying uncertainty are that of *stochastic programming* and *robust optimization*. Stochastic programs are often implemented as a two-stage decision problem in

which the first-stage decision (design variables) is made before a realization of the uncertain parameters occurs. A second-stage recourse problem is solved after the realization occurs. Using stochastic programming approaches, uncertainty is modeled as either a discrete or continuous set of random events through the use of probability measures. This requires *a priori information* pertaining to the probability that a given realization of uncertainty occurs. The reader is directed toward Li and Grossmann [160] for an extensive review of stochastic programming approaches for optimization under uncertainty. The focus of this thesis is on worst-case design problems. Accordingly, probabilistic approaches such as stochastic optimization do not provide adequate assurances that are required for our purposes.

A deterministic approach is preferred when system-level failures are extremely costly or when the uncertainty set is poorly characterized. In this case, bounds on uncertainty may be available when probabilistic characterization is either impractical or impossible. Worst-case formulations are characterized by the presence of uncertain variables $\mathbf{p}$ bounded by a compact *u*ncertainty set $P \subset \mathbb{R}^{n_p}$ leading to multilevel program formulations such as the *bilevel program*:

$$
\begin{aligned}
f^* = \quad & \min_{\mathbf{x},\hat{\mathbf{z}}} f(\mathbf{x}) \\
\text{s.t.} \quad & \max_{\hat{\mathbf{z}},\mathbf{p}} g(\mathbf{x}, \hat{\mathbf{z}}, \mathbf{p}) \leq 0, \\
\text{s.t.} \quad & \mathbf{h}(\mathbf{x}, \hat{\mathbf{z}}, \mathbf{p}) = \mathbf{0}, \\
& \mathbf{x} \in X, \mathbf{p} \in P, \hat{\mathbf{z}} \in D_z \subset \mathbb{R}^{n_z}.
\end{aligned}
\tag{2.5.1}
$$

In this formulation, $\mathbf{x}$ represents a vector of decision variables, $\hat{\mathbf{z}}$ represents a vector of internal state variables governed by the model equations, and $\mathbf{p}$ represents a vector of parameters. Bilevel programs are generally nonconvex and nonsmooth, and as such, we make no assumptions about the smoothness of g. As a direct consequence, multiple constraints $g_1, \ldots, g_n$, may be trivially

handled by reformulation into a single constraint $g = \max_i g_i$. Furthermore, additional constraints may exist in which only decision variables or uncertain variables participate, which, in turn, define X and P. We omit these constraints from (2.5.1) and further refinements thereof for brevity.

A number of papers have detailed industrially relevant applications of robust optimization. These range from the standard robust design problem to novel applications in model predictive control from optimal disturbance model identification [55] to economic nonlinear model predictive control formulations [235]. In addition, multilevel optimization problems play an important role in the optimal design of experiments, as illustrated by a recent investigation of trilevel formulations [318]. Chapter 8 details a key application of interest that was addressed during the course of this thesis.

The preliminary work on bilevel programs that focused on linear formulations or convexity conditions [28, 325] was subsequently extended to nonlinear programs [10, 76, 311]. Then, Gümüş and Floudas [116] described a nonconvex global optimization method that replaced the inner program with a set of nonlinear algebraic constraints under the assumption of the linear independence constraint qualification. The authors then leveraged αBB relaxations with a branch-and-bound algorithm to solve the KKT-reformulated NLP, despite the requirement of convexity for the KKT conditions to be necessary and sufficient. A primary limitation relating to the approach of Gümüş and Floudas [116] is that it cannot generally provide convergent upper bounds for bilevel problems with nonconvex inner programs. Mitsos et al. [190] subsequently describe an approach that can solve general nonconvex programs in the absence of coupling equality constraints.

Multiparametric programming was developed as an alternative strategy, which consists of recasting the bilevel program as a single-level deterministic optimization problem. The parametric solution of the inner program [226, 231] is explicitly characterized, which makes it suitable for

use in real-time optimization [226]. However, these methods are only applicable if the inner program is linear or quadratic; accordingly, this method is not suitable for general nonconvex programs. This thesis focuses on the most general formulation of the robust optimization problem, which includes nonconvex equality constraints. This motivates the consideration of equivalent SIP reformulations of the bilevel program, which may be more readily solved.

In many cases, the state variables are uniquely determined once decision variables are fixed for a given realization of the uncertain variables. The equality constraint $\mathbf{h}(\mathbf{x}, \hat{\mathbf{z}}, \mathbf{p}) = \mathbf{0}$ then implicitly specifies a function $\hat{\mathbf{z}} = \mathbf{z}(\mathbf{x}, \mathbf{p})$. Further, assume that $\mathbf{h} : D_x \times D_z \times D_p \to \mathbb{R}^{n_y}$ is continuously differentiable on the open set D_z, that relaxations of $\mathbf{h}$ and $\nabla_z \mathbf{h}$ are available, for instance by using McCormick relaxations provided that $\mathbf{h}$ is factorable. In this case, the equality constraints may be eliminated from (2.5.1) reducing the problem to the inequality-constrained bilevel program:

$$\begin{aligned} f^* = \quad & \min_{\mathbf{x},\mathbf{p}} f(\mathbf{x}) \\ \text{s.t.} \quad & \max_{\hat{\mathbf{z}},\mathbf{p}} g(\mathbf{x}, \mathbf{z}(\mathbf{x}, \mathbf{p}), \mathbf{p}) \leq 0, \\ & \mathbf{x} \in X, \mathbf{p} \in P, \hat{\mathbf{z}} \in D_z \subset \mathbb{R}^{n_z}. \end{aligned} \tag{2.5.2}$$

The inequality constrained bilevel program (2.5.2) may then be written as the following equivalent semi-infinite program (SIP):

$$\begin{aligned} f^* = \quad & \min_{\mathbf{x} \in X} f(\mathbf{x}) \\ \text{s.t.} \quad & g(\mathbf{x}, \mathbf{z}(\mathbf{x}, \mathbf{p}), \mathbf{p}) \leq 0, \ \forall \mathbf{p} \in P. \end{aligned} \tag{2.5.3}$$

In this formulation, the objective function $f : X \to \mathbb{R}$ depends solely on the decision variables

$\mathbf{x} \in X$ and the constraint $g : Z \times X \times P \to \mathbb{R}$ are parameterized by the uncertain variable $\mathbf{p}$ on a set of infinite cardinality P. The SIP formulation covers classes of robust design and optimization under uncertainty problems of specific interest to this thesis. The feasibility of a point $\bar{\mathbf{x}}$ with respect to (2.5.3) is determined by solving the inner program, as defined in Definition 2.5.1, to global optimality.

Definition 2.5.1 (Inner Program [298])**.** Given a point $\bar{\mathbf{x}} \in X$, the inner program is formulated as:

$$\bar{g}(\bar{\mathbf{x}}) = \max_{\mathbf{p} \in P} g(\mathbf{z}(\bar{\mathbf{x}}, \mathbf{p}), \bar{\mathbf{x}}, \mathbf{p}).$$

If $\bar{g}(\bar{\mathbf{x}}) \leq 0$, $\bar{\mathbf{x}}$ is feasible in (2.5.3).

The pioneering work of Blankenship and Falk [42, 100] provided a cutting-plane methodology to solve (2.5.3). In the absence of convexity assumptions, the convergence of this method is only guaranteed in the limit. An αBB estimation scheme was presented [104] that converges to a stationary point in the limit, and a further adaptation to arbitrary parameter sets has since been described [294]. A general nonconvex algorithm for solving SIPs in finite time was first developed by Bhattacharjee et al. [37], which added exponentially more cutting planes at each iteration of the algorithm. Mitsos [188] then introduced an adaption of this method, *SIPres*, that generates cutting planes that scale linearly with the number of iterations. A further adaptive methodology for formulating the discretization set was detailed by Djelassi and Mitsos [77]. These methods for solving SIPs require that a series of nonconvex optimization problems is solved to global optimality at each iteration of the algorithm [188, 298]. Lastly, an implicit approach to generating relaxations of SIPs with implicitly-defined functions was presented by Stuber and Barton [298], and was subsequently extended to generalized semi-infinite programs (GSIPs) and bilevel programs with coupling constraints [78]. For a complete review of recent contributions to nonconvex SIP algorithms and applications, the reader is directed to Djelassi

et al. [79]

A principal challenge underlying the solution of nonconvex SIPs is the efficient solution of the nonconvex inner program in reduced space. Chapters 3 - 7 detail a number of advances that address this very challenge for both nonconvex and dynamic optimization.

Chapter 3

EAGO.jl: Easy Advanced Global Optimization in Julia

In this chapter, an extensible open-source deterministic global optimizer (EAGO) programmed entirely in the Julia language, is presented. EAGO was developed to serve the need for supporting higher-complexity user-defined functions (e.g., functions defined implicitly via algorithms) within optimization models. EAGO embeds a first-of-its-kind implementation of McCormick arithmetic in an Evaluator structure allowing for the construction of convex/concave relaxations using a combination of source code transformation, multiple dispatch, and context-specific approaches. Utilities are included to parse user-defined functions into a directed acyclic graph representation and perform symbolic transformations enabling dramatically improved solution speed. EAGO is compatible with a wide variety of local optimizers, the most exhaustive library of transcendental functions, and allows for easy accessibility through the JuMP modeling language. Together with Julia's minimalist syntax and competitive speed, these powerful features make EAGO a versatile research platform enabling easy construction of novel meta-solvers, incorporation and utilization of new relaxations, and extension to advanced problem formulations encountered in engineering

and operations research (e.g., multilevel problems, user-defined functions). The applicability and flexibility of this novel software is demonstrated on a diverse set of examples. Lastly, EAGO is demonstrated to perform comparably to state-of-the-art commercial optimizers on a benchmarking test set.

3.1 Introduction

Mathematical optimization problems are ubiquitous in scientific and technical fields. Applications range from aerospace and chemical process systems to finance. However, even relatively simple physical processes such as mixing, may introduce significant nonconvexity into problem formulations [290]. As such, nonconvex programs often represent the most faithful representations of the system of interest. Multiple approaches have been developed to address these cases. Heuristics such as evolutionary algorithms, may approximate good solutions for select problems. However, heuristics may fail to guarantee that even a feasible solution is detected in finite time [14]. In scenarios where a guarantee is necessary, such as determining a reactor's maximum safe operating temperature, complete nonconvex methods must be applied. As nonconvex optimization problems are NP-hard, the development of appropriate optimizers remains an active area of research [136].

State-of-the-art complete global optimizers generally require that a problem can be constructed in an algebraic modeling language (AML). These languages provide a specialized interface that allows users to construct optimization problems in a manner that can be interpreted by optimizers; translating high-level syntax into the requisite C/Fortran code and reducing chances for user-input errors. In addition to serving this central function, AMLs have evolved to provide additional features such as embedding automatic differentiation (AD) schemes and

passing standard expression forms to optimizers [106]. This last feature is required by nearly every complete nonconvex optimizer available today. These regular expressions are then parsed by the optimizer via the introduction of auxiliary variables until all expressions in the model correspond to a form in the optimizers reference library of relaxations. In many cases, this reference library is limited to a few expressions: BARON [254] does not support trigonometric functions, ANTIGONE [185] and Couenne [27] limit expressions to those allowed by AMPL [106] or GAMS [176] environments. In addition, these complete optimizers do not support user-defined functions.

Many real-world optimization problems arise naturally from computer simulations and take non-canonical forms. Problems defined in terms of ordinary differential equation (ODE) and differential-algebraic equation (DAE) systems are pervasive in process systems engineering. Moreover, for many early-stage design problems, optimization of a model with an embedded simulation represents only one of many valuable tasks. Others may center around validating simulations, illustrating system dynamics, and assessing sensitivities [31]. It is often necessary to recast these problems into an AML prior to optimization. The manual reformulation of a model may be quite challenging for both subject matter experts and non-experts alike. While application-specific simulation packages attempt to bridge this gap, only gPROMS [24, 25, 92] provides access to state-of-the-art global optimizers. As a propriety software offering, gPROMS is not readily extensible to user-defined scripts, and suffers from the same lack of adaptability as other AMLs. Therefore, users that require non-canonical formulations must resort back to using AMLs or their own painstakingly implemented code. A few AMLs have mitigated this issue by offering simpler and more intuitive interfaces. Further enhancements to usability have been provided by subject area-specific extensions in AMLs, such as Pyomo [121] and JuMP [84].

As an alternative to AML extensions, set-valued arithmetic may be used to construct bounds.

In this approach, convex and concave relaxations of non-canonical functions are constructed using composition rules by overloading methods or objects. In each of these approaches, a set-valued data type is constructed and arithmetic rules such as $+$, $\div$, $\sin(\cdot)$, etc., are defined for this data type. Interval arithmetic [194, 219] was one of the first such approaches, but subsequent work has extended this to affine arithmetic [72], and most recently McCormick arithmetic [177, 191, 271]. While interval methods are quite general and can be readily used in global optimization, McCormick arithmetic offers quadratic convergence rates for a wide class of functions with tighter bounds [44, 203]. In addition, McCormick operator theory has been developed to allow for the relaxation of implicit functions, such as those arising in parametric ODE- and PDE-constrained systems and nonlinear algebraic systems lacking closed-form parametric solutions [191, 271, 296, 300]. These approaches allow for the relaxation of a wide variety of user-defined functions that arise in numerous application areas. For instance, a process flow diagram may be reformulated into a simulation that is solved in a sequential block fashion.

The usage of McCormick relaxations has been limited by two significant factors. First, no publicly available optimizers made use of McCormick composition rules to construct convex and concave relaxations. EAGO was the first optimizer to remedy this issue (by a matter of years) along with a recent release of a C++ implementation, MAiNGO [46]. Secondly, the bounds furnished by set-valued arithmetic may be markedly weaker than polyhedral approaches, slowing solution speed [221]. One of the reasons for this disadvantage is that operator-overloading approaches cannot recognize and exploit internal problem structure during the presolve phase and exploit this information throughout the course of the branch-and-bound algorithm when the recognition of linear terms, common subexpressions, and convexity properties—as well as the introduction of auxiliary variables—may be beneficial [290, 303, 305].

Our package EAGO—an acronym for **E**asy **A**dvanced **G**lobal **O**ptimization—seeks to

provide a unified framework for both auxiliary variable methods and set-valued arithmetic methods with a greatly simplified user interface. Included in EAGO are a development toolkit, a state-of-the-art deterministic global optimizer, and an API for optimizing problems defined as functions in Julia script. The primary appeal of constructing EAGO in the Julia language [36] is that it strikes a balance between the speed requirement of scientific computing and the high-level syntax. Using Julia, algorithms can be written in a syntax simpler than MATLAB^{TM} while achieving execution speeds as fast as C and Fortran [36]. A simple package management system is included in Julia's base distribution making propagation of software quite simple. In addition to the above advantages, EAGO exploits Julia's Lisp-like abstract syntax tree (AST) for handling expressions to implement tasks that would be extremely difficult, and in some cases impossible in other scientific programming languages. As a consequence, optimization problems may be formulated in Julia script and passed to an advanced global optimizer using a function no more complicated than MATLAB's `fmincon` [63]. This interface serves two purposes. First, it is expected to act as a bridge to new users, who may be unfamiliar with efficient problem formulations or AML syntax, by allowing them to try out a global optimizer on a specific class of test problems before investing the time and energy in learning an AML. In addition, it provides subject matter experts with unparalleled flexibility and the advanced capabilities needed to address the highly-complex problems on the forefront of optimization research.

In this chapter, we detail the EAGO deterministic global optimization package and its novel implementation. In Section 3.2, we describe a nonconvex optimizer developed using this toolkit. In Section 3.3, we provide preliminary benchmarking data illustrating the relative competitiveness of EAGO to extant approaches. In Section 3.4, we discuss EAGO's extensibility and flexibility. Lastly, we conclude in Section 3.5 by summarizing EAGO's capabilities and suggesting directions for future research.

3.2 Global Optimization Framework

The EAGO package maintains a high-performance deterministic global optimizer. At the time of writing this, the best optimizer available is detailed below. Further upgrades to EAGO's default optimizer will be described in the release notes on future tagged versions of the EAGO.jl package available through Julia's package manager. The EAGO optimizer is specialized to treat programs with nonlinear objectives and constraints via a simulation approach rather than via the auxiliary variable method. The distinction between the approaches is primarily the result of how the factorable representation of the problem is treated. In the following section, we discuss the implementation of the default EAGO optimizer. The relaxations supported are detailed in Section 3.2.2. A description of the presolve steps is presented in Section 3.2.3. An overview of the domain reduction techniques used is given in Section 3.2.5. In Section 3.2.6, the construction of the relaxed lower-bounding problem is discussed. In Section 3.2.7, we conclude by discussing the formulation of the upper-bounding problem.

3.2.1 A Flexible Branch-and-Bound Routine

EAGO includes an implementation of the spatial branch-and-bound algorithm [136]. When used with specific lower-bounding and node-selection routines, this algorithm furnishes an ϵ-optimal global solution after a finite number of iterations. Numerous approaches to generating bounds and selecting nodes have been shown to provide these guarantees. Further development of such routines remains the subject of active research. For a detailed discussion of the branch-and-bound algorithm, the reader is referred to the excellent review presented in [136]. Most complete global optimizers also perform some additional processing routines on each node to shrink the domain size [234]. EAGO allows the user to define preprocessing and

postprocessing functions as necessary. Information furnished from computing the relaxations can be readily accessed from either of these functions.

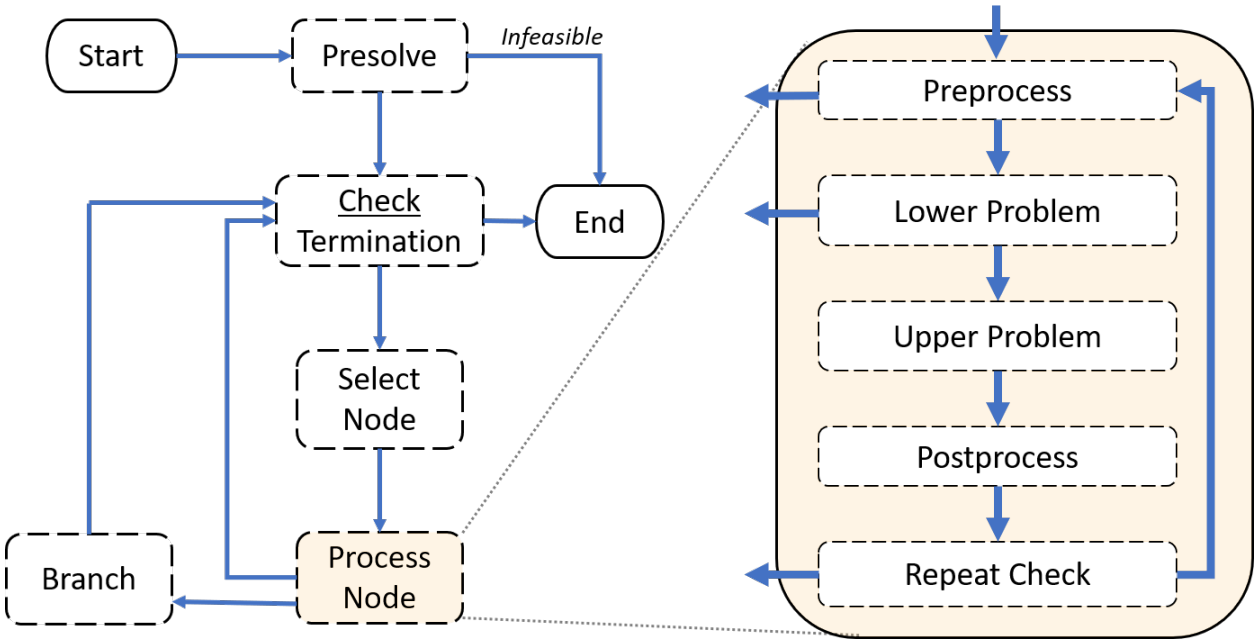

FIGURE 3.2.1: A block flow diagram depicting the main flexible branch-and-bound routine implemented in EAGO.

We will utilize the following notation within the flexible branch-and-bound framework (Alg. 1) presented below. Let $Y \in \mathbb{IR}^n$ be the box constraints on the decision variable $\mathbf{y}$ of program (2.3.1). Let the global lower and upper bound of f^* in (2.3.1) at iteration k be denoted by α_k and β_k, respectively. Let $Y_l \in \mathbb{I}Y$ correspond to a node in the branch-and-bound tree where f_l^{LBD} and f_l^{UBD} are lower and upper bounds of f on Y_l, respectively. Lastly, let $f^* = f(\mathbf{y}^*)$ be the ϵ-optimal objective function value where $\mathbf{y}^*$ is a feasible point corresponding to a solution of (2.3.1) at termination of the algorithm.

Algorithm 1 (Flexible Branch-and-Bound Framework)**.**

1. Initialization
   (a) Set the stack to $\Sigma = \{Y\}$.
   (b) Set the iteration number $k := 0$, set tolerances, set the global upper bound $\alpha_0 := +\infty$, and the global lower bound $\beta_0 := -\infty$.
2. Termination
   (a) If $\Sigma = \emptyset$, the algorithm terminates with (2.3.1) infeasible.

(b) Check if a termination condition is satisfied.[1] If the absolute tolerance condition is satisfied, terminate with $f^* := \alpha^k$ as the ϵ-optimal estimate for the optimal objective function value and $\mathbf{y}^*$ as a feasible point at which f^* is attained.

(c) Delete from Σ all nodes with $f_l^{LBD} > \alpha_k$ and set $\beta_k := \min_{Y^l \in \Sigma} f_l^{LBD}$.

3. Node Selection

   (a) Select and delete a node Y_l from the stack Σ according to a selection heuristic.

4. Preprocessing

   (a) Apply preprocessing routines[2]. If infeasibility on Y_l is established, go to Step 2, else set Y_l to the bounds determined by preprocessing routines.

5. Lower-Bounding Problem

   (a) Construct and solve the lower-bounding problem globally on Y_l.

   (b) If the problem is infeasible, set $f_l^{LBD} := +\infty$. Otherwise, set f_l^{LBD} to the value of the solution. If $f_l^{LBD} < \alpha^k$ and the feasible optimal solution found, $\check{\mathbf{y}}$, is also feasible in (2.3.1), then set $\alpha_k := f_l^{LBD}$ and $\mathbf{y}^* := \check{\mathbf{y}}$.

6. Upper-Bounding Problem (optional)

   (a) Solve (2.3.1) locally on Y_l.

   (b) If a feasible solution is found with $f_l^{UBD} < \alpha_k$, $\alpha_k := f_l^{UBD}$ and set $\mathbf{y}^*$ to the solution found.

7. Postprocessing

   (a) Apply postprocessing routine[2] and adjust bounds of Y_l, accordingly.

8. Fathoming

   (a) If $f_l^{LBD} = +\infty$ or $f_l^{LBD} > \alpha_k$, go to Step 2.

9. Repetition

   (a) Checks if the repetition condition is satisfied[3]. If so, proceed to Step 4.

10. Branching

[1] To ensure an ϵ-optimal solution is reached in finite time, the termination condition must be based on an absolute tolerance. Other conditions are included to deal with numerical issues. Additional assumptions must also be satisfied such as pointwise convergence of the relaxations and Lipschitz continuity of the functions involved.

[2] Typically, domain reduction algorithms may be applied at this step.

[3] If domain reduction in postprocessing yields a significant improvement, repetition may be beneficial.

(a) Select a branching dimension i according to a selection heuristic. Partition node Y_l in this dimension to form nodes $Y_{l'}$ and $Y_{l''}$.

(b) Set the lower bounds of the new nodes, $f^{LBD}_{Y_{l'}}, f^{LBD}_{Y_{l''}} := f^{LBD}_{l}$, and add these nodes to the working stack Σ.

(c) Advance the iteration number, $k := k + 1$, and go to Step 2.

While multiple software implementations of branch-and-bound exist, the degree to which the user can adapt the actual branch-and-bound algorithm to specialized problem formulations is often limited [2, 242]. For example, Juniper only provides a framework for branching on integer variables [156]. EAGO's flexible branch-and-bound routine is novel in that it circumvents these limitations by allowing the user to redefine any method used by the main routine. Each block depicted in Figure 3.2.1 can be set to user-defined subroutines. This is done by defining a subtype `EAGO.ExtensionType` and then extending the EAGO's base method to specialize on this newly defined type. A demonstration of this usage is provided in Section S2.2 of the Supplementary Materials. This modular architecture allows for the easy implementation of custom optimization schemes.

3.2.2 Relaxations

The relaxation libraries provided with EAGO are based on McCormick relaxation composition rules. The McCormick relaxation of the bilinear function was first introduced by McCormick in [177]. This relaxation which consists of bounding the bilinear term using a series of affine inequalities which many commercially available global optimizers such as ANTIGONE and BARON [66, 117, 185, 254]. In the past decade, a significant effort has been made to further generalize this approach to arbitrary nonlinear functions. An operator-overloading scheme for constructing McCormick-based relaxations of functions described by a class of algorithms was detailed by Mitsos et al. [191] and has been outlined in Section S1 of the Supplementary

Materials. Variations on this manner of constructing relaxations through operator overloading have been termed *McCormick relaxations*; a convention we adopt herein to maintain consistency with the extant body of literature.

Theoretical developments for generalizing the McCormick relaxation framework and constructing convex and concave composite relaxations using arbitrary convex and concave functions were established by Scott et al. [271]. More recently, tighter composition rules for multiplication and maxima operators were presented in [207, 310]. Methods of generating relaxations of implicit functions were developed by Stuber et al. [300]. Wechsung et al. [324] described a method of propagating McCormick relaxations backwards on a direct acyclic graph (DAG) representation of a problem. A method for tightening interval bounds was described in [206]. Alternative differentiable relaxations were introduced in [151, 152]. Additionally, McCormick relaxations have been shown to converge quadratically under reasonable assumptions [44, 203, 205]; a requirement for avoiding clustering with branch-and-bound [146]. While multiple papers have shown the utility of solving problems via McCormick relaxations of factorable functions, the availability of an optimizer using these relaxations is quite limited.

EAGO fills this void as a pioneering open-source optimization solver and research platform supporting McCormick-based relaxations of general factorable functions. Envelopes of simple expressions are used to generate relaxations of nonconvex intermediate functions. The following functions are currently supported in EAGO with correctly-rounded interval bounds:

- **Arithmetic**: +, -, ×, /, sqr, power, ln, log, sqrt, exp
- **Nonsmooth**: abs, sign, min, max
- **Trigonometric**: cos, sin, tan, sec, csc, cot
- **Inverse Trigonometric**: acos, asin, atan, asec, acsc, acot

- **Hyperbolic**: cosh, sinh, tanh, sech, csch, coth
- **Inverse Hyperbolic**: acosh, asinh, atanh, asech, acsch, acoth

Additional scaled versions of common operators and transcendental functions such as log10(·) and sind(·) are also supported. As such, relaxations of nearly all functions available in standard AD libraries are included in EAGO. Rules for computing subgradients of McCormick relaxations and gradients of the differentiable McCormick relaxations are also included.

The calculation of convex and concave envelopes for convexoconcave (a univariate function, $q : D \to \mathbb{R}$, for which $\exists p \in D$ such that q is convex on $\{d \in D \mid d \leq p\}$ and concave on $\{d \in D \mid d \geq p\}$), concavoconvex (the negation of a convexoconcave function), and general periodic functions requires the computation of anchor points at which line segments meet. These anchor points, that depend only on interval bounds, are calculated using a one-dimensional root-finding algorithm, such as a secant method or Newton's method. EAGO computes roots to an absolute tolerance of 10^{-10}. The computation of these anchor points and propagating correctly-rounded interval bounds may represent the main computational expense when generating relaxations as illustrated by Figure 3.2.2.

One of the powerful features of EAGO is its ability to decouple various components required to evaluate a relaxation via a source code transformation. A Wengert list [114] (synonymously *tape* or *trace*) is generated using a source code transformation technique that supports field access, nested-tape generation, and array operators that would be challenging for most state-of-the-art AD software packages [245]. The representation provided by the JuMP AML is used to create a specialized `Evaluator` for nonlinear and user-defined expressions. The `Evaluator` structure included in EAGO allows for the reuse of intermediate values obtained by computationally expensive protocols. Interval bounds and anchor points for each factor are only recomputed if the relaxation is being constructed on a new domain or if both forward and reverse

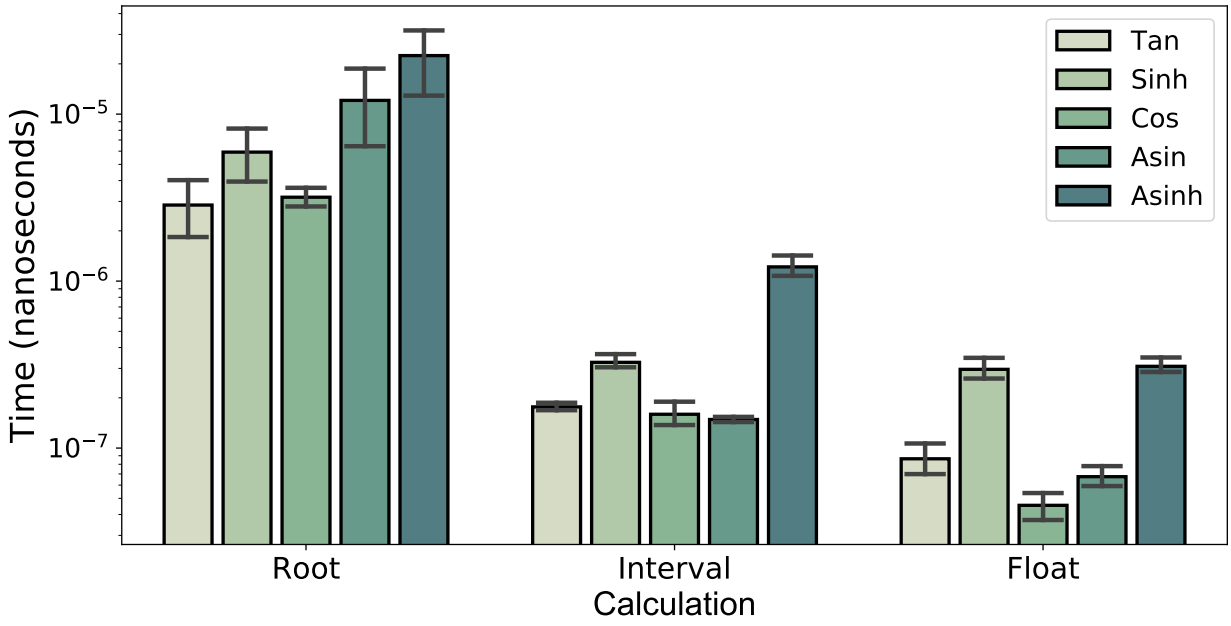

FIGURE 3.2.2: A breakdown of the run-time for computing relaxations associated with implementing an bjected-oriented approach by operator.

passes are performed (which may alter interval bounds). This expedites additional evaluations such as those required to add additional cutting planes to form outer approximations or as part of callback function evaluations requested by a local NLP optimizer. The interval subgradient cut detailed in Proposition 3.2.1, Section 3.5.4, is only performed during the first forward pass, after all forward-reverse passes are performed, or evaluation occurs on a new domain. This prevents the distinct relaxation from being evaluated at each point. Flowcharts depicting the distinction between this novel approach and a pure overloading-based implementation are provided in Figure 1 of the Supplementary Materials.

To supplement the source code transformation approach, a full multiple dispatch-based McCormick relaxation implementation is included. This library defines methods for McCormick operators that dispatch on the `struct MC` data type. Each instance of the `struct MC` stores the convex relaxation in the field `cv`, the concave relaxation in the field `cc`, the interval bounds in the

field `Intv`, and subgradients of convex and concave relaxations in the `cv_grad` and `cc_grad` fields, respectively. The user may force EAGO to use the multiple dispatch implementation to compute relaxations in order to reduce memory requirements using keyword arguments. Additionally, for some select expressions where source code transformations are not expected to yield improvements to computational speed, EAGO defaults to a multiple dispatch implementation.

3.2.3 Presolving

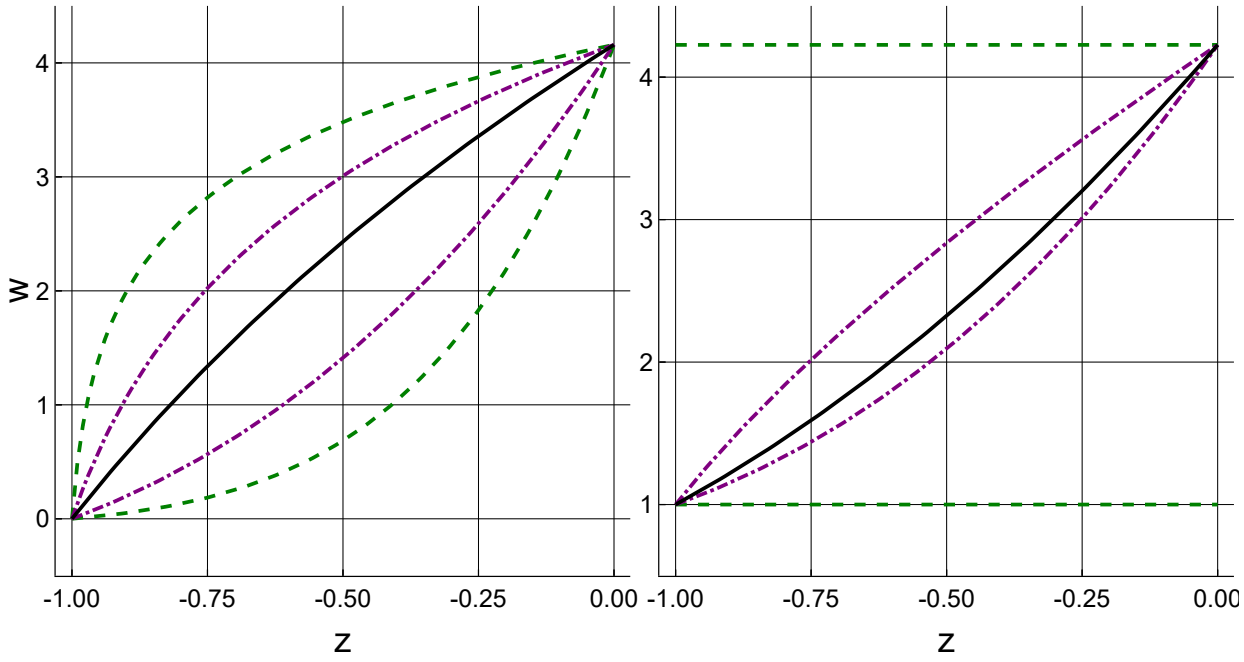

FIGURE 3.2.3: Suppose $z \in [-1, 0]$, $a = 2$, $g(z) = (z + 2)^3$ and the standard order of operations is used to evaluate expressions. **Left:** The function $w = a^{\log g(z)}$ (——) is plotted with convex/concave relaxations of $w = a^{\log g(z)}$ (- - -) and the rearrangement $w_r = g(z)^{\log a}$ (-·-·-). **Right:** The function $w = \log g(z)g(z)$ (——) is plotted with convex/concave relaxations of $w = \log g(z)g(z)$ (- - -) and the rearrangement $w_r = \log g(z) + \log g(z)$ (-·-·-).

Presolving may dramatically improve solution times by detecting special structures, rearranging algebraic terms to tighten relaxations, and potentially lifting problems into simpler

higher-dimensional forms. EAGO utilizes a variety of techniques to accomplish this goal. First, linear and quadratic constraints are stored in a sparse format during the solution routine. A list of variables appearing only in linear and quadratic terms are constructed. Subsequently, a DAG representation of every nonlinear objective and constraint function present in the model is generated. After construction, the DAG undergoes algebraic rearrangement to improve relaxation performance. These rearrangements simplify expressions and change terms with weak relaxations into equivalent terms with tighter relaxations (e.g., treating the subexpression $x\log(x)$ as the convex negative entropy function). This is similar to ANTIGONE's use of (3.2.1) and (3.2.6) for reducing the level at which auxiliary variables are introduced in standard factorable program [185] or using the rearrangements of Khajavirad and Sahinidis [149] to improve the inferences made via disciplined convex programming (DCP) [279].

EAGO's framework provides a simple syntax for implementing expression transformation by registering template DAGs corresponding to subexpressions. This is done by creating the appropriate `Template_Graph` objects then invoking the `register_substitution!`. Transformations to the Wengert list are made by locating subexpressions that match specified forms and substituting in equivalent subexpressions. The rearrangements given in Equations (3.2.1)-(3.2.7) are included in EAGO's optimizer and result in McCormick relaxations that may

be significantly tighter than the original form.

$$\exp(x)\exp(y) = \exp(x+y) \tag{3.2.1}$$

$$\log(a^x) = x\log(a) \tag{3.2.2}$$

$$(a^x)^b = (a^b)^x \tag{3.2.3}$$

$$(x^a)^b = x^{(ab)} \tag{3.2.4}$$

$$a^{\log(x)} = x^{\log(a)} \tag{3.2.5}$$

$$\log(xy) = \log(x) + \log(y) \tag{3.2.6}$$

$$\log(x/y) = \log(x) - \log(y) \tag{3.2.7}$$

An illustration of the tightening effect produced by applying (3.2.5) and (3.2.6) is given in Figure 3.2.3. Note that (3.2.1)-(3.2.4) reduce the number of subexpressions that are relaxed. This improves computation speed as calculations of relaxations are significantly more expensive than floating-point calculations. These rearrangements do not tighten natural interval bounds.

3.2.4 User-Defined Functions

The development of relaxations of functions implemented in a script form rather than an AML form can be exceedingly difficult for a complete global optimizer to handle. While operator and method overloading techniques can be applied to a broad class of problems, the introduction of auxiliary variables for these functions can be challenging. Moreover, basic overloading approaches can be subject to a number of flaws that limit their practical use. In the case of AD methods, perturbation errors may result from nested overloading (e.g., required to form higher-order derivatives) [245]. Additionally, overloading approaches to tape generation require

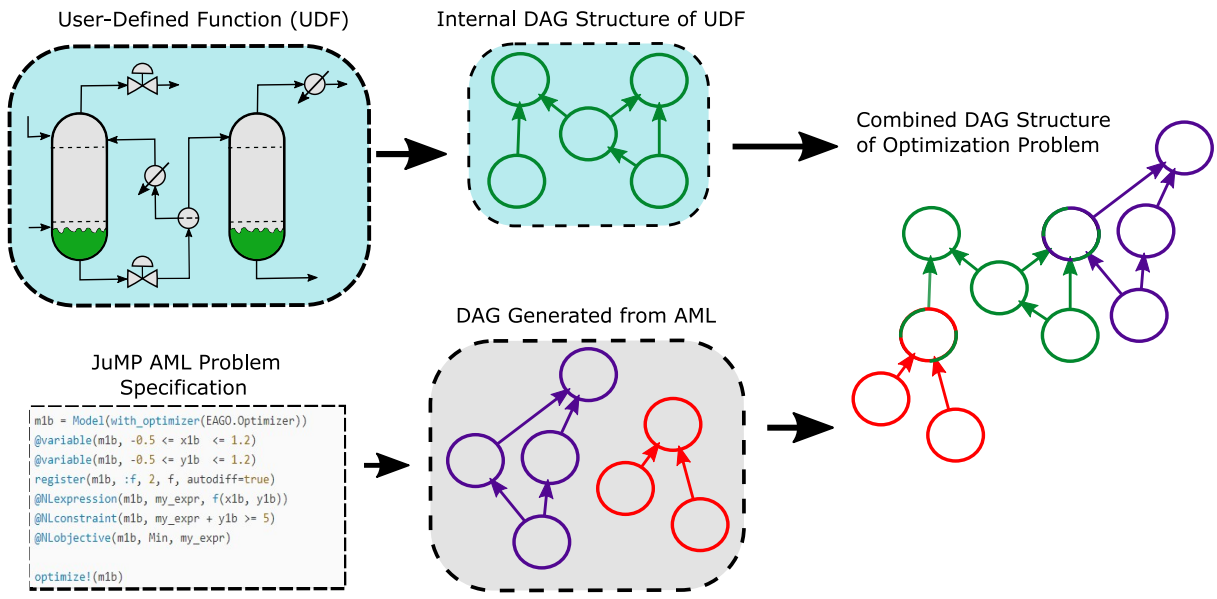

FIGURE 3.2.4: Given an optimization problem, the EAGO generates a DAG representation for all user-defined functions, composes these into a shared DAG representation of all nonlinear expressions, further detects special structures, solves the optimization problem, and returns the model with respect to the original variables.

that all potential promotions are anticipated and accounted for. This limits the ability to integrate such tracing methods with other packages that may use structures internally to perform calculations. EAGO makes use of context-oriented programming tools introduced with Julia 0.7 to address forms inaccessible to method overloading or operator overloading approaches [244]. This allows for the generation of DAG representations from functions defined by Julia script; a core feature of EAGO.

In many technical application areas, components of a simulation may be represented by a series of scripts. For instance, a chemical process flowsheet model will typically embed a series of equations of state, mass and energy balances, and rules for solving the system operating under particular equipment specifications (e.g., adiabatic flash) for each block representing a unit operation. These simulations may be addressed using methods that embed sequential modular

solves in these user-defined functions via a semi-explicit approach. EAGO's handling of user-defined functions allows for the construction of forward and reverse evaluation of blocks of these simulations and the determination of DAG structures that span from the process blocks to the entire model. Figure 3.2.4 illustrates how EAGO's framework allows it to reconcile process models with AML defined optimization problems. Additional user-defined functions may arise naturally as a description of dynamic behavior such as the chemical kinetic examples in [328] and Section 3.2.4.

AD is a powerful tool for addressing these formulations using local nonlinear optimizers by providing gradient information of the objective functions and constraints [114]. The direct optimization of problems with user-defined functions may also be desired in a global optimization context. To our knowledge, EAGO is the only deterministic global optimizer capable of handling user-defined factorable functions. User-defined functions may include type-assertions and calls to functions that aren't appropriate for direct overloading approaches as defined. Additionally, many implementations would require that the user change local storage objects. EAGO makes use of a context-oriented approach that overcomes each of these potential issues when generating the Wengert list for each function. In this manner, EAGO allows us to solve optimization problems that include partial derivatives of an arbitrary order as illustrated in Section 3.2.4.

Optimization of a Model With an Embedded Algorithm

Consider an aqueous *n*-butanol mixture undergoing a separation (by heating at constant pressure, P [bar], and temperature, T [K], in a fixed volume). Depending on the operating temperature, the system exhibits two immiscible liquid phases and potentially a minimum boiling azeotrope. We will assume the existence of two immiscible phases is undesirable due to equipment limitations.

This implies the following liquid phase stability conditions:

$$\frac{d\ln(\gamma_1 x_1)}{dx_1} > 0, \quad \frac{d\ln(\gamma_2 x_2)}{dx_2} > 0 \tag{3.2.8}$$

where x_1 is the liquid-phase mole-fraction of *n*-butanol, x_2 is the liquid-phase mole-fraction of water, and a van Laar activity coefficient model is given by

$$\ln(\gamma_1) = \frac{1253/T}{(1 + 2.62(x_1/x_2))^2} \tag{3.2.9}$$

$$\ln(\gamma_2) = \frac{479/T}{(1 + 0.382(x_2/x_1))^2}. \tag{3.2.10}$$

where γ_1 and γ_2, are the respective activity coefficients of *n*-butanol and water. A modified Raoult's Law and mass balance yield two other equality constraints:

$$0 = P - x_1\gamma_1 P_1^{vp} + x_2\gamma_2 P_2^{vp}$$

$$0 = x_1 + x_2 - 1,$$

where P_1^{vp} and P_2^{vp} are respectively the vapor pressures of water and *n*-butanol, given by:

$$P_1^{vp} = 1.33 \times \exp(11.83572 - 4169.84/(T - 17.665)$$

$$P_2^{vp} = 1.33 \times \exp(11.33986 - 3724.52/(T - 69.854).$$

An operating condition is sought with the minimum temperature such that the vapor phase mass fraction of *n*-butanol is at least 0.95 and no liquid-liquid phase split occurs. The vapor

composition specification is then represented by the following constraint:

$$x_1 \gamma_1 P_1^{vp} / P \geq 0.95$$

We can then write the objective function as:

$$f(x1, x2, T) = T \tag{3.2.11}$$

where we restrict to the ranges of interest as follows $x_1 \in [0.01, 0.99]$, $x_2 \in [0.01, 0.99]$, and $T \in [363.15, 398.15]$. Within 10.1 seconds, EAGO is able to prove that no such operating condition exists, which is consistent with visual inspection of the Txy-diagrams [307].

Listing 3.1: Script used to set up and optimize the example given in Section 3.2.4.

```
using JuMP, EAGO, ForwardDiff

# Define the activity model
gamma1_x1(z) = z[1]*(1253/z[3])/(1 + 2.62*(z[1]/z[2]))^2
gamma2_x2(z) = z[2]*(479/z[3])/(1 + 0.382*(z[2]/z[1]))^2
cons_1ex(z...) = ForwardDiff.gradient(z -> log(gamma1_x1(z)), collect(z))[1]
cons_2ex(z...) = ForwardDiff.gradient(z -> log(gamma2_x2(z)), collect(z))[2]

# Define the JuMP model and solve
m = Model(EAGO.Optimizer)
register(m, :cons_1ex, 3, cons_1ex, autodiff = true)
register(m, :cons_2ex, 3, cons_2ex, autodiff = true)

@variable(m, 0.01 <= x[i=1:2] <= 0.99)
@variable(m, 363.15 <= T <= 398.15)
@constraint(m, x[1] + x[2] == 1.0)
@NLexpression(m, P1, 1.33*exp(11.83572 - 4169.84/(T - 17.665)))
@NLexpression(m, P2, 1.33*exp(11.33986 - 3724.523/(T - 69.854)))
@NLconstraint(m, cons_1ex(x[1],x[2],T)*P1 + cons_2ex(x[1],x[2],T)*P2 == 1.02)
@NLconstraint(m, cons_1ex(x[1],x[2],T)*P1/1.02 >= 0.95)
@NLconstraint(m, cons1, cons_1ex(x[1], x[2], T) >= 0.001)
@NLconstraint(m, cons2, cons_2ex(x[1], x[2], T) >= 0.001)
@NLobjective(m, Min, T)
optimize!(m)
```

Kinetic Parameter Estimation

Consider the kinetic parameter estimation problem [191], which was adapted from [287, 306]. The reaction mechanism can be modeled using the initial value problem:

$$\begin{aligned}
\frac{dx_A}{dt} &= k_1 x_Z x_Y - c_{O_2}(k_{2f} + k_{3f})x_A + \frac{k_{2f}}{K_2}x_D + \frac{k_{3f}}{K_3}x_B - k_5 x_A^2 \\
\frac{dx_B}{dt} &= c_{O_2}k_{3f}x_A - \left(\frac{k_{3f}}{K_3} + k_4\right)x_B, \quad \frac{dx_Z}{dt} = -k_1 x_Z x_Y \\
\frac{dx_D}{dt} &= c_{O_2}k_{2f}x_A - \frac{k_{2f}}{K_2}x_D, \quad \frac{dx_Y}{dt} = -k_{1s} x_Z x_Y \\
x_A(0) &= 0, x_B(0) = 0, x_D(0) = 0, x_Y(0) = 0.4, x_Z(0) = 140
\end{aligned}$$

where x_j is the concentration of species $j \in \{A, B, D, Y, Z\}$. The constants are given by $T = 273$, $K_2 = 46\exp(6500/T - 18)$, $K_3 = 2K_2$, $k_1 = 53$, $k_{1s} = k_1 \times 10^{-6}$, $k_5 = 1.2 \times 10^{-3}$, and $c_{O_2} = 2 \times 10^{-3}$. Intensity versus time data is available in [296] as well as a known dependency on concentration, $I = x_A + \frac{2}{21}x_B + \frac{2}{21}x_D$[283]. The reaction rate constants $k_{2f} \in [10, 1200]$, $k_{3f} \in [10, 1200]$, and $k_4 \in [0.001, 40]$ are unknown and form the parameter vector $\mathbf{p} = (k_{2f}, k_{3f}, k_4)$.

An implicit Euler discretization was constructed in [300, 328] and solved to global optimality by constructing relaxations of implicit functions using a fixed-point method. While EAGO can replicate this implicit approach, we will consider the original case of the explicit Euler discretization of the problem [191] for simplicity's sake. In this example, a semi-explicit approach is used; the relaxations of intermediate factors $\mathbf{x}$ that arise during the simulation of the ODEs are computed. These factors are subsequently propagated through to the objective function effectively eliminating the need to explicitly consider the $\mathbf{x}$ values in the problem formulation, as

detailed in [48, 191]. A discretization consisting of 200 timesteps provides sufficiently high accuracy for this problem. The discretized model becomes:

$$
\begin{aligned}
x_A^{i+1} &= x_A^i + \Delta t\left(k_1 x_Y^i x_Z^i - c_{O_2}(k_{2f}+k_{3f})x_A^i + \frac{k_{2f}}{K_2}x_D^i + \frac{k_{3f}}{K_3}x_B^i - k_5(x_A^i)^2\right)\\
x_B^{i+1} &= x_B^i + \Delta t\left(k_{3f}c_{O_2}x_A^i - \left(\frac{k_{3f}}{K_3}+k_4\right)x_B^i\right)\\
x_D^{i+1} &= x_D^i + \Delta t\left(k_{2f}c_{O_2}x_A^i - \frac{k_{2f}}{K_2}x_D^i\right)\\
x_Y^{i+1} &= x_Y^i + \Delta t\left(-k_{1s}x_Y^i x_Z^i\right)\\
x_Z^{i+1} &= x_Z^i + \Delta t\left(-k_{1s}x_Y^i x_Z^i\right)
\end{aligned}
$$

where $i = 0, \ldots, 199$ and $\Delta t = 0.01$. While only three parameters are of interest in the original optimization problem, 1003 variables are required to specify this in an AML along with knowledge of reasonable variable box bounds. This is because the state vector:

$$
\mathbf{x} = \left(x_A^1, x_B^1, x_D^1, x_Y^1, x_Z^1, \cdots, x_A^{200}, x_B^{200}, x_D^{200}, x_Y^{200}, x_Z^{200}\right) \tag{3.2.12}
$$

must be accounted for as decision variables in typical formulations. Since EAGO computes composite relaxations, this problem formulation allows us to treat this state vector as a series of intermediate expressions to be evaluated; therefore, allowing us to define the problem with respect to the kinetic parameters of interest $\mathbf{p}$.

LISTING 3.2: Explicit Euler integration scheme used in the kinetic parameter estimation example (Ex. 3.2.4).

```
function explicit_euler_integration(p)

   x = zeros(typeof(p[1]),1005); # Data storage array
   x[4] = 0.4; x[5] = 140        # Sets initial condition
   T = 273; delT = 0.01;   cO2 = 2e-3; k1 = 53;  k1s = k1*1E-6
   K2 = 46*exp(6500/T-18); K3 = 2*K2;  h = delT; k5 = 1.2E-3

   for i=1:200                   # Offset by 1, initial condition is x[1:5]
      term1 = k1*x[5i-1]*x[5i]-cO2*(p[1]+p[2])*x[5i-4]
      term2 = p[1]*x[5i-2]/K2+p[2]*x[5i-3]/K3-k5*x[5i-4]^2
      x[5i+1] = x[5i-4] + h*(term1 + term2)
      x[5i+2] = x[5i-3] + h*(p[2]*cO2*x[5i-4]-(p[2]/K3+p[3])*x[5i-3])
      x[5i+3] = x[5i-2] + h*(p[1]*cO2*x[5i-4]-p[1]*x[5i-2]/K2)
      x[5i+4] = x[5i-1] + h*(-k1s*x[5i-1]*x[5i])
      x[5i+5] = x[5i] + h*(-k1s*x[5i-1]*x[5i])
    end

    return x
end
```

The objective function for this problem can then be given by

$$f(\mathbf{p}) = \sum_{i=1}^{n} \left(I_i^c(\mathbf{p}) - I_i^d \right)^2 \tag{3.2.13}$$

where I_i^c are the calculated intensity values at time step i from the model and I_i^d are the values corresponding to the experimental data. The EAGO optimizer converges to within 90% in just 2.6 seconds, and to within 95% in 8.2 seconds which is comparable to the time presented in [191].

Listing 3.3: Script to load data and define the objective function for the Example given in Section 3.2.4.

```
using EAGO, JuMP, DataFrames, CSV

# Loads data from csv to DataFrame
data = CSV.read("kinetic_intensity_data.csv")
pL = [10.0,  10.0,  0.001];   pU = [1200.0,  1200.0,  40.0]

# Defines function for intensity
I(xA,xB,xD) = xA + (2/21)*xB + (2/21)*xD

# Integrates the ODEs and calculates SSE
function objective(p...)
    x = explicit_euler_integration(p)
    SSE = zero(typeof(p[1]))
    for i=1:200
        SSE += (I(x[5i-4],x[5i-3],x[5i-2]) - data[:intensity][i])^2
    end
    return SSE
end
```

Listing 3.4: Build the JuMP model and calculate a ϵ-global optimal solution of Example in Section 3.2.4.

```
# Create model and add variables
m = Model(EAGO.Optimizer)
@variable(m, pL[i] <= p[i=1:3] <= pU[i])

# Register objective, add objective function, and optimize
fobj(p...) = objective(p...)
JuMP.register(m, :fobj, 3, fobj, autodiff=true)
@NLobjective(m, Min, fobj(p...))
JuMP.optimize!(m)
```

3.2.5 Domain Reduction

Various techniques are commonly used to shrink each subdomain Y^l generated within the branch-and-bound algorithm. These techniques may significantly speed up the branch-and-bound algorithm by eliminating large regions of the search space. EAGO makes use of three main families of these routines: optimality-based bounds-tightening (Sec. 3.2.5), feasibility-based

bounds-tightening (Sec. 3.2.5), and duality-based bounds-tightening (Sec. 3.2.5). EAGO executes the first two approaches during preprocessing while duality-based bounds-tightening is applied during postprocessing.

Optimization-Based Bounds-Tightening (OBBT)

In addition to the lower bound calculation, relaxations are also used to tighten bounds via optimization-based bounds-tightening (OBBT). For variables participating in a nonlinear term, problem (3.2.14) is solved to obtain potentially tighter lower and upper variable bounds. EAGO implements OBBT using a greedy algorithm with filtering [111].

$$\begin{aligned} \min_{\mathbf{y}} \ & \pm y_k \\ \text{s.t.} \ & \mathbf{g}^{cv}(\mathbf{y}) \leq \mathbf{0} \\ & \mathbf{h}^{cv}(\mathbf{y}) \leq \mathbf{0} \\ & \mathbf{h}^{cc}(\mathbf{y}) \geq \mathbf{0} \\ & f^{cv}(\mathbf{y}) \leq \alpha_k \end{aligned} \tag{3.2.14}$$

OBBT entails solving a large number of optimization problems. As such, EAGO uses OBBT at the root node and then uses a heuristic to determine if it will be used at deeper nodes in the branch-and-bound tree. For a node of depth $d \leq k$ (default k: 4), OBBT is performed. For nodes of depth $d > k$, the OBBT performed with probability 2^{k-d} [26].

Duality-Based Bounds-Tightening (DBBT)

Duality-based bounds-tightening (DBBT) is performed following the solution of the relaxed problem. The dual multiplier λ_i of each variable y_i is queried along with the lower bound *LBD*; all positive dual multipliers are used to shrink the variable bounds [251]:

$$y_i \geq y_i^U - \lambda_i^{-1}(\alpha_k - LBD)$$
$$y_i \leq y_i^L + \lambda_i^{-1}(\alpha_k - LBD).$$

Feasibility-Based Bounds-Tightening

Expression specific feasibility-based bounds-tightening is provided for linear constraints, univariate quadratic constraints [81], and bivariate quadratic constraints [312]. For linear constraints, the following relationships are used. The set of linear constraints $\sum_{j=1:n} a_{ij} y_j \leq b_i, i = 1, \ldots, m$. Each linear constraint i is then processed sequentially and the variable bounds are refined through application of the following relationship:

$$y_k^U \leq \frac{1}{a_{ij}} \left(b_i - \sum_{j \neq k} \min(a_{ij} y_j^U, a_{ij} y_j^L) \right) \qquad a_{ik} \geq 0$$
$$y_k^L \geq \frac{1}{a_{ij}} \left(b_i - \sum_{j \neq k} \min(a_{ij} y_j^U, a_{ij} y_j^L) \right) \qquad a_{ik} < 0.$$

For a full discussion of univariate and bivariate constraint bounds-tightening, the reader is encouraged to consult [81] and [312], respectively.

Constraint Propagation on the Directed Acyclic Graph

A two-stage constraint propagation scheme is used for nonlinear terms represented on the DAG. In this first stage, natural interval extensions along with McCormick relaxations (and with associated subgradients) computed at a reference point $\bar{\mathbf{y}}$ of the nonlinear constraints and the objective are calculated via a forward pass on the DAG in topological order. The subgradients are used to improve the interval bounds if possible, according to Proposition 3.2.1 [206, 300]. The computed bounds of the constraints are then intersected with constraint bounds. In the second stage, a reverse interval pass is then performed in reverse topological order [299]. This is repeated, inferring tighter variable values until either the variable bounds fail to tighten by a preset factor (default: 0.99) or a maximum number of repetitions is reached (default: 3)

Proposition 3.2.1. Let $v : Y \to V$ be a factor in the computation of McCormick relaxations such that $V = [v^L, v^U]$ are known interval bounds (e.g., by a natural interval extension) with convex/concave relaxations v^{cv}/v^{cc} of v on Y and their respective subgradients $\mathbf{s}_v^{cv}$, $\mathbf{s}_v^{cc}$ computed at $\mathbf{y} = \bar{\mathbf{y}} \in Y$. The functions $\omega, \mu : Y \to \mathbb{R}$ are the affine relaxations of the convex and concave relaxations of v on Y, respectively, and defined as:

$$\omega(\mathbf{y}) \equiv v^{cv}(\bar{\mathbf{y}}) + \mathbf{s}_v^{cv}(\bar{\mathbf{y}})^{\mathrm{T}}(\mathbf{y} - \bar{\mathbf{y}})$$
$$\mu(\mathbf{y}) \equiv v^{cc}(\bar{\mathbf{y}}) + \mathbf{s}_v^{cc}(\bar{\mathbf{y}})^{\mathrm{T}}(\mathbf{y} - \bar{\mathbf{y}}).$$

The lower and upper bounds of these relaxations are themselves valid bounds of v on Y. Valid lower and upper bounds of the image set v(Y) are given by:

$$v^{L,new} := \max(v^L,\ \omega^L(Y)) \tag{3.2.15}$$
$$v^{U,new} := \min(v^U,\ \mu^U(Y)). \tag{3.2.16}$$

By construction, the bounds furnished by (3.2.15) and (3.2.16) are at least as tight as the original interval bounds. EAGO uses the midpoint of Y as the reference point, $\bar{\mathbf{y}}$; this is repeated until either the variable bounds reduction falls below a preset threshold (default: 0.99) or until a maximum number of repetitions is reached (default: 5).

3.2.6 Lower-Bounding Problem

A lower bound on the optimal solution value is calculated by solving to global optimality the relaxation of (2.3.1), given as:

$$\begin{aligned} f^{LBD} = \quad & \min_{\mathbf{y}\in Y} f^{cv}(\mathbf{y}) \\ \text{s.t.} \quad & \mathbf{g}^{cv}(\mathbf{y}) \leq \mathbf{0} \\ & \mathbf{h}^{cv}(\mathbf{y}) \leq \mathbf{0} \\ & \mathbf{h}^{cc}(\mathbf{y}) \geq \mathbf{0}. \end{aligned} \tag{3.2.17}$$

EAGO's default optimizer further relaxes this form via polyhedral outer approximation of the McCormick-based relaxations. For nonlinear expressions, an affine relaxation is generated via an affine approximation of the expression at the midpoint of the domain using subgradient information [191]. Objective function value cuts taken at the midpoint are also added using the

current global upper bound, α_k:

$$
\begin{aligned}
f^{LBD} = \quad & \min_{\mathbf{y}\in Y} f^{cv}(\bar{\mathbf{y}}) + \mathbf{s}_f^{cv}(\bar{\mathbf{y}})^{\mathrm{T}}(\mathbf{y}-\bar{\mathbf{y}}) \\
\text{s.t.} \quad & \mathbf{g}^{cv}(\bar{\mathbf{y}}) + \mathbf{s}_g^{cv}(\bar{\mathbf{y}})^{\mathrm{T}}(\mathbf{y}-\bar{\mathbf{y}}) \leq \mathbf{0} \\
& \mathbf{h}^{cv}(\bar{\mathbf{y}}) + \mathbf{s}_h^{cv}(\bar{\mathbf{y}})^{\mathrm{T}}(\mathbf{y}-\bar{\mathbf{y}}) \leq \mathbf{0} \\
& \mathbf{h}^{cc}(\bar{\mathbf{y}}) + \mathbf{s}_h^{cc}(\bar{\mathbf{y}})^{\mathrm{T}}(\mathbf{y}-\bar{\mathbf{y}}) \geq \mathbf{0} \\
& f^{cv}(\bar{\mathbf{y}}) + \mathbf{s}_f^{cv}(\bar{\mathbf{y}})^{\mathrm{T}}(\mathbf{y}-\bar{\mathbf{y}}) \leq \alpha_k.
\end{aligned}
$$

This is done in part because the standard McCormick relaxations [191] are potentially nonsmooth and therefore may pose difficulty for gradient-based NLP optimizers. Additionally, the polyhedral relaxation can be significantly faster than a differentiable NLP relaxation for certain problems. The lower-bounding problem is then solved using the specified LP optimizer. Additional cutting planes are generated by adding constraints at new reference points based on the solution of prior relaxation (default: up to 3) and the objective is then updated with the new reference point. Alternatively, EAGO can supply an `Evaluator` structure to local NLP optimizers. This evaluator can be queried for function and subgradient values. If differentiable McCormick relaxations are selected a MathOptInterface-wrapped local NLP optimizer may be used to furnish lower bounds instead.

3.2.7 Upper-Bounding Problem

The use of tight upper bounds can accelerate the convergence of the branch-and-bound algorithm by increasing the rate at which nodes are fathomed. One popular choice is that of a feasible local solution. As solving an NLP to local optimality can be computationally expensive, EAGO makes use of a heuristic similar to that of Couenne [26] to limit the number of upper-bounding problems

solved. That is, for a node of depth d less than a tolerance k, the local NLP is solved. For nodes of depth $d > k$, the local NLP is solved with probability 2^{k-d} [26]. By default, EAGO uses Ipopt [317], but any other `MathOptInterface.jl` compatible NLP optimizer can be specified passing keyword arguments to the EAGO optimizer.

3.3 Numerical Experiments

The data files and code for all examples are freely available in the EAGO Git repository, `https://github.com/PSORLab/EAGO.jl`. Additional special use cases in the Supplementary Materials further illustrate EAGO's flexibility. EAGO version 0.4.0 was used with an absolute tolerance of $\epsilon_a = 10^{-3}$ and a relative tolerance of $\epsilon_r = 10^{-3}$. All numerical experiments were run three times on a single thread of a 3.60GHz (4.00GHz turbo) Intel Xeon E3-1270 v5 processor with 32GB in Ubuntu 18.02LTS and Julia v1.4.2. The lower-bounding problem was solved using Gurobi 9.0.2 [117]. The upper-bounding problem was solved using Ipopt v3.12.13 [317]. Julia, Ipopt, and CPLEX are all compiled with Intel MKL 2019 (Update 3) versions of LAPACK/BLAS. EAGO makes use of the JuMP mixed-mode AD scheme for general problems [84] and a forward-mode AD scheme for user-defined functions [245]. MathOptInterface v0.9.13 and JuMP version 0.21.2 were used to formulate problems and provide interfaces to sub-solvers in a myriad of internal subroutines. The ValidatedNumerics.jl library was used for correctly-rounded interval calculations [167] using the `:accurate` rounding mode that is slightly more conservative than the IEEE Standard 1788-2015 [143], but is often significantly faster.

Benchmark Performance

One of the primary advantages of EAGO is the relative speed of the Julia language. To illustrate this, we provide a comparison of the solution times between EAGO and three state-of-the-art deterministic global optimizers: BARON, ANTIGONE, and SCIP. Twenty problems were selected from the MINLP2 and GLOBAL library that contains only expressions supported by BARON, ANTIGONE, and SCIP. A list of the problems along with a brief summary of formulation traits is given in Table 3.3.3. A maximum of 1000 seconds are allowed for each numerical experiment. BARON v17.10.16, ANTIGONE v1.1, and SCIP v5.0 were used for these experiments. Optimizer performance is assessed using the methods presented in Dolan and Moore [80]. A performance profile for this test set is provided in Figure 3.3.1 and accompanying data is provided in Tables 2 and 3 of the Supplementary Materials. The *performance* of optimizer s is the time in CPU seconds, $t_{p,s}$, required to solve problem p. The *performance ratio* on problem p by optimizer s is the ratio of the optimizer's performance to the best optimizer's performance in the set:

$$r_{p,s} = \frac{t_{p,s}}{\min\{t_{p,s} : s \in S\}}.$$

The *performance profile* of optimizer s is the plot of the distribution function of the performance metric where $\rho_s(\tau)$ is the probability that a performance ratio $r_{p,s}$ is within a factor $\tau \in \mathbb{R}$ of the best possible ratio

$$\rho_s(\tau) = \frac{1}{n_p}\text{card}\{p \in \mathcal{P} : r_{p,s} \leq \tau\}$$

where card(S) denotes the cardinality of set S, $\mathcal{P}$ is the set of problems, $n_p = \text{card}(\mathcal{P})$. The percent relative gap remaining at time t for problem p is given by

$g_t = 100 \times (UBD - LBD)/\max(|UBD|, |LBD|)$. As illustrated by Figure 3.3.1, EAGO solves many problems with times comparable to state-of-the-art global optimizers. Of the problems not solved within the time limit, EAGO typically yields a smaller percent relative gap than the other optimizers. This suggests EAGO can provide meaningful run time results when used to develop novel optimization routines. However, no claim of superiority is appropriate at this time. This is particularly evident for nonconvex quadratic programs in which SCIP and BARON both outperform EAGO. This is expected as EAGO makes use of the McCormick-based relaxation framework for relaxing quadratic constraints whereas other optimizers use specialized routines for relaxing quadratic constraints. We leave a more exhaustive benchmarking analysis for a later date.

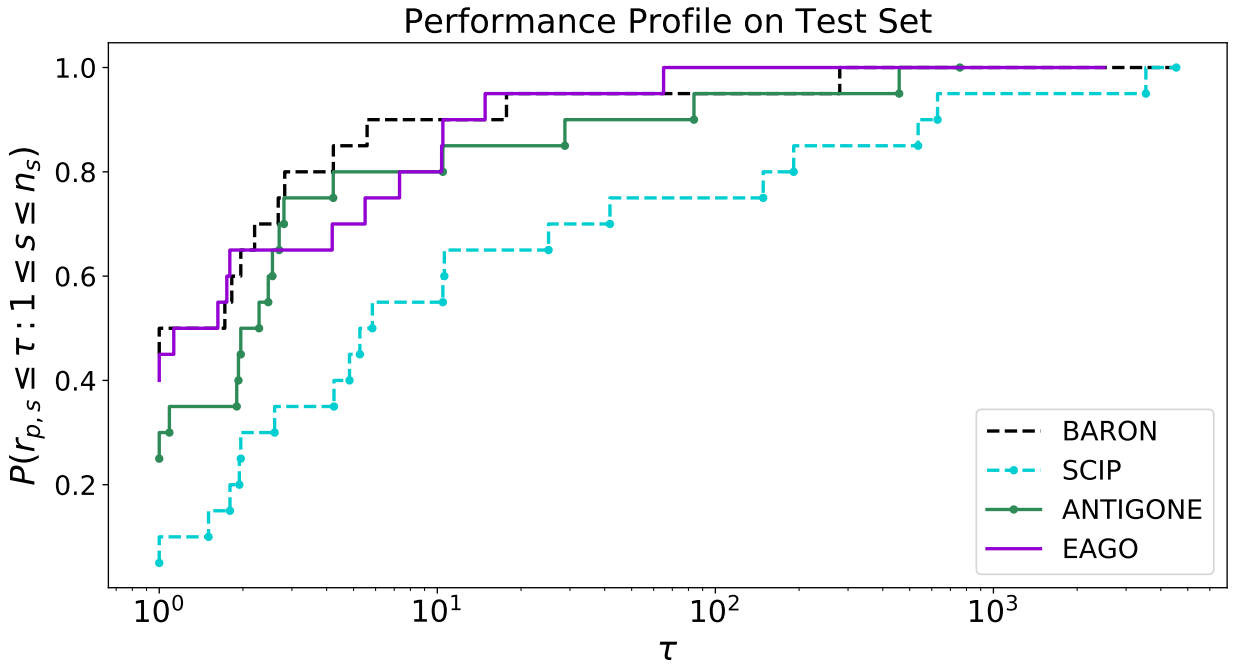

FIGURE 3.3.1: Performance profiles for the test set enumerated in Table 3.3.3.

3.4 Extensibility of EAGO

The EAGO framework has already been used to demonstrate multiple novel approaches to global optimization. The quasiconvexity of a hybrid-solar system was exploited allowing for the problems to be solved with a certificate of global optimality [297]. In subsequent works, novel affine relaxations [54] and relaxations of implicit functions [328], were implemented by customizing and extending EAGO, respectively. One illustration of this capability is contained in EAGO itself as nonconvex semi-infinite programs (SIPs) solution functionality in Section 3.4.1.

3.4.1 Solving Semi-Infinite Programs

EAGO implements the SIPres algorithm [188, 298] for solving general nonconvex SIPs that converges in a finite number of iterations under mild assumptions. For full implementation details, the reader is directed to the EAGO GitHub repository at `https://github.com/PSORLab/EAGO.jl`. Consider the standard-form SIP:

$$\begin{aligned} f^* = \quad & \min_{\mathbf{x}\in X} f(\mathbf{x}) \\ \text{s.t.} \quad & \mathbf{g}(\mathbf{x},\mathbf{p}) \leq 0, \ \forall \mathbf{p} \in P, \ |P| \leq \infty \\ & X = \{\mathbf{x} \in \mathbb{R}^{n_z} : \mathbf{x}^L \leq \mathbf{x} \leq \mathbf{x}^U\} \\ & P = \{\mathbf{p} \in \mathbb{R}^{n_p} : \mathbf{p}^L \leq \mathbf{p} \leq \mathbf{p}^U\}. \end{aligned} \tag{3.4.1}$$

A design centering problem developed in [104] is presented here. A location consisting of coordinates x_1, x_2 is sought such that a disk with maximal radius, x_3, can be inscribed within a

nonconvex container set. The SIP is stated formally as:

$$
\begin{aligned}
f^* = \quad & \max_{\mathbf{x}\in X} x_3 \\
\text{s.t.} \quad & g_1(\mathbf{x}, p) = 0.3\sin(\pi z_1(\mathbf{x}, p)) - z_2(\mathbf{x}, p) \leq 0, \ \forall p \in P \\
& g_2(\mathbf{x}, p) = z_1(\mathbf{x}, p)^2 + 0.3 z_2(\mathbf{x}, p)^2 - 1 \leq 0, \ \forall p \in P \\
& X = [-1.5, 1.5] \times [-1, 2] \times [0, 1.5] \\
& P = [0, 2\pi]
\end{aligned}
\tag{3.4.2}
$$

where the $\mathbf{z}$ terms are determined by the expressions:

$$
\begin{aligned}
z_1(\mathbf{x}, p) &= x_1 + x_3\cos(p) \\
z_2(\mathbf{x}, p) &= x_2 + x_3\sin(p).
\end{aligned}
$$

This problem is constructed in the few lines of code contained in Code Listing 3.5. Using EAGO's implementation of the SIPres routine in combination with its global optimizer, we were able to solve (3.4.2) and obtain certification of global optimality with an absolute tolerance of $\epsilon_a = 10^{-3}$ in 1.97 seconds. All prior attempts to address this problem required approximations of the constraint set or the use of approximate methods as done in the original work of Floudas [104].

LISTING 3.5: The code for solving the design centering problem given in Example 3.4.2 originally presented in [104]

```
using EAGO, Gurobi

f(x) = x[3]                                          # Define objective
z1(x,p) = x[1] + x[3]*cos(p[1])
z2(x,p) = x[2] + x[3]*sin(p[1])
gSIP1(x,p) = 0.3*sin(pi*z1(x,p)) - z2(x,p)           # Define SIP constraints
gSIP2(x,p) = z1(x,p)^2 + 0.3*z2(x,p)^2 - 1.0

xL = [-1.5, -1.0, 0.0];  xU = [1.5, 2.0, 1.5]        # Defines bounds
pL = [0.0];  pU = [2.0*pi]

opt = Gurobi.Optimizer()
result = explicit_sip_solve(xL, xU, pL, pU, f, [gSIP1, gSIP2],
                            sip_sense = :max, relaxed_optimizer = opt)
```

3.4.2 Customizing EAGO's Solver: A Quasiconvex Problem

In addition to supporting a wide variety of problem forms, EAGO can be readily adapted to suit the unique needs of users and application requirements. In this example, we will modify EAGO's branch-and-bound solver to solve a quasiconvex problem using a bisection technique. The problem consists of minimizing a quasiconvex function $f : C \to \mathbb{R}$ over a convex feasible set. This can be done using a few simple lines of code. Consider the quasiconvex problem presented

in [144]:

$$
\begin{aligned}
f^* = \quad & \min_{\mathbf{y}\in Y} f(\mathbf{y}) && (3.4.3)\\
\text{s.t.} \quad & \sum_{i=1}^{5} i \cdot y_i - 5 = 0 && (3.4.4)\\
& \sum_{i=1}^{5} y_i^2 - 0.5\pi \leq 0 && (3.4.5)\\
& -\left(\frac{1}{2}y_1^2 + \frac{1}{2}y_2^2 + y_3^2 + 2y_1y_2 + 4y_1y_3 + 2y_2y_3\right) \leq 0 && (3.4.6)\\
& -y_1^2 - 6y_1y_2 - 2y_2^2 + \cos(y_1) + \pi \leq 0 && (3.4.7)\\
& Y = [0,5]^5
\end{aligned}
$$

where

$$
f(\mathbf{y}) = -\frac{\ln((5+y_1)^2 + \sum_{i=1}^{5} y_i)}{1 + \sum_{i=1}^{5} y_i^2}. \tag{3.4.8}
$$

Interval analysis shows that $f^* \in F = [f^L, f^U] = [-5, 0]$. As such, we can introduce a new auxiliary variable $t \in T = F$ and formulate the equivalent problem below:

$$
\begin{aligned}
t^* = \quad & \min_{\mathbf{y}\in Y, t\in T} t\\
\text{s.t.} \quad & \text{(3.4.4)-(3.4.7)}\\
& f(\mathbf{y}) - t \leq 0\\
& Y = [0,5]^2, \quad T = [-5, 0].
\end{aligned}
$$

In order to solve this problem, we resort to a bisection algorithm with respect to the interval T. Let $\phi_\tau(\mathbf{y}) = f(\mathbf{y}) - \tau$ such that $\tau = (t^L + t^U)/2$, then we solve for $\mathbf{y}$ subject to constraints

(3.4.4)-(3.4.7) such that $\phi_\tau(\mathbf{y}) \leq 0$. If this problem is feasible, $t^* \in [t^L, \tau]$. Otherwise, $t^* \in [\tau, t^U]$. The other interval is then discarded (fathomed) and this manner of bisection is repeated until an interval containing a feasible solution with a width of at most ϵ is located [51]. Then, we simply need to bisect solely in the t dimension. To implement this new solver using EAGO, three main modifications to the optimizer must be made. First, we will short circuit the preprocessing, postprocessing, and upper-bounding steps.

LISTING 3.6: Preprocessing, postprocessing, and upper-bounding problem for example in Section 3.4.2.

```
using MathOptInterface, EAGO, JuMP
import EAGO: Optimizer

struct QuasiConvex <: EAGO.ExtensionType end

import EAGO: preprocess!, upper_problem!, postprocess!
function EAGO.preprocess!(t::QuasiConvex, x::Optimizer)
    x._preprocess_feasibility = true
end
function EAGO.upper_problem!(t::QuasiConvex, x::Optimizer)
    x._upper_feasibility = true
end
function EAGO.postprocess!(t::QuasiConvex, x::Optimizer)
    x._postprocess_feasibility = true
end
```

Next, one specifies that the algorithm should terminate on convergence to an ϵ-optimal point and return status codes should indicate a feasible optimal solution was returned.

LISTING 3.7: Modify convergence and termination criteria for example in Section 3.4.2.

```
import EAGO: convergence_check, termination_check, repeat_check
function EAGO.convergence_check(t::QuasiConvex, x::Optimizer)
    gap = (x._upper_objective_value - x._lower_objective_value)
    return (gap <= x._parameters.absolute_tolerance)
end
function EAGO.termination_check(t::QuasiConvex, x::Optimizer)
    flag = EAGO.convergence_check(t, x)
    if flag
        x._termination_status_code = MathOptInterface.OPTIMAL
        x._result_status_code = MathOptInterface.FEASIBLE_POINT
    end
    return flag
end
```

It is further specified that bisection should only occur in the t dimension and only the feasible node need be saved.

LISTING 3.8: Specify dimensions for bisection for example in Section 3.4.2.

```
branch_variable = [i == 6 for i=1:6]
EAGO.repeat_check(t::QuasiConvex, x::Optimizer) = true
```

We then specify how the subproblem should be solved at each iteration. Here, the problem is solved at the midpoint value of t using the native EAGO function `upper_problem!`. Bounds are then contracted depending on the feasibility of the subproblem.

LISTING 3.9: Modify lower-bounding problem to solve new subproblem for example in Section 3.4.2.

```
import EAGO: lower_problem!
function EAGO.lower_problem!(t::QuasiConvex, x::Optimizer)
  y = x._current_node
  lower = y.lower_variable_bounds[6]
  upper = y.upper_variable_bounds[6]
  midy = (lower + upper)/2.0
  y.lower_variable_bounds[6] = midy
  y.upper_variable_bounds[6] = midy
  EAGO.solve_local_nlp!(x)
  feas = x._upper_feasibility
  y.lower_variable_bounds[6] = feas ?  lower : midy
  y.upper_variable_bounds[6] = feas ?  midy : upper
  x._lower_objective_value = y.lower_variable_bounds[6]
  x._upper_objective_value = y.upper_variable_bounds[6]
  x._lower_feasibility = true
  return
end
```

We can now define the problem using a JuMP syntax and retrieve the solution. The two keyword options which don't reference previously-defined functions specify an absolute tolerance of $\epsilon_a = 10^{-8}$ should be used and the routine should not terminate when a feasible point is located (even though no objective is specified).

LISTING 3.10: Construct and optimize the JuMP model for example in Section 3.4.2

```
m = Model(optimizer_with_attributes(EAGO.Optimizer,
          "absolute_tolerance" => 1E-8,
          "branch_variable" => branch_variable,
          "ext_type" => QuasiConvex()))

@variable(m, ((i<6) ? 0 : -5) <= y[i=1:6] <= ((i<6) ? 5 : 0))
@constraint(m, sum(i*y[i] for i=1:5) - 5 == 0)
@constraint(m, sum(y[i]^2 for i=1:5) - 0.5*pi^2 <= 0)
@expression(m, expr1, 2*y[1]*y[2] + 4*y[1]*y[3] + 2*y[2]*y[3])
@constraint(m, -(0.5*y[1]^2 + 0.5*y[2]^2 + y[3]^2 + expr1) <= 0)
@NLexpression(m, expr2, log((5 + y[1])^2 + sum(y[i] for i=1:5)))
@NLconstraint(m, -y[1]^2 -6*y[1]*y[2] -2*y[2]^2 +cos(y[1]) + pi <= 0)
@NLconstraint(m, -expr2/(1 + sum(y[i]^2 for i=1:5)) - y[6] <= 0)
@objective(m, Min, y[6])
JuMP.optimize!(m)

# retrieve solution info
solution = JuMP.value.(y[1:5])
global_obj_value = JuMP.value.(y[6])
```

This problem can then be solved to an absolute tolerance of $\epsilon_a = 10^{-8}$ in just 0.7 seconds.

3.5 Concluding Remarks

We have presented the first openly-available McCormick relaxation-based global optimizer. This optimizer contains a number of features prevalent in modern nonconvex optimization software and each of these features can be readily included or omitted from user-developed routines. Among the features discussed were an interval constraint programming algorithm, an affine interval bounds-tightening algorithm, duality-based bounds-tightening algorithm, and a generic optimization-based bounds-tightening routine. Routines for constructing outer approximations of convex function envelopes were discussed.

The basic optimizer developed in EAGO performs impressively relative to state-of-the-art

deterministic global optimizers on the selected examples shown. The number of algebraic expressions supported by EAGO is significantly larger than the existing complete global optimizers. In fact, the library size is comparable to that of modern AD software [23]. Most notably, we have illustrated the use of our deterministic global optimizer with native Julia code. As illustrated, support for script-defined functions serves a two-fold purpose. First, it allows non-experts to explore simple optimization problems without domain-specific knowledge. Second, it provides a framework that experts can use to construct specialized algorithms for computing global optima of problems with embedded simulations, already utilized in recent works [54, 297, 328].

Multiple avenues exist to further improve EAGO's branch-and-bound optimization framework and default optimizer. One potential avenue of interest is exploring the use of tighter interval arithmetic in conjunction with McCormick-based relaxations. The use of tighter interval bounds such as those generated by interval-Taylor arithmetic [33] may yield significantly tighter relaxations and speed solution time [205]. Another potential improvement to EAGO lies in further specializing it to handle various categories of models with embedded simulations. One such improvement may lie in selectively adding auxiliary variables to the formulation in the presolve step to tighten relaxations and improve domain reduction. Another improvement may lie in the automatic detection of implicit function reformulations [300, 328] and the application of specialized routines to address these. One major adaptation of the EAGO toolkit presently underway is the incorporation of a branch-and-cut framework for solving mixed-integer nonlinear problems [305].

Mean Solution Time (CPU Seconds)				
Name	EAGO	SCIP	BARON	ANTIGONE
alkyl	0.07	0.742	0.296	0.16
BeckerLago	0.14	0.34	0.08	60.6
ex2_1_8	0.12	0.632	0.671	0.324
ex3_1_1	0.476	0.758	0.422	0.459
ex4_1_9	0.168	0.29	0.06	0.148
ex5_4_3	0.056	0.26	0.15	0.143
ex6_2_10	> 1000	> 1000	67.3	189
ex6_2_11	21.9	> 1000	9	5.228
ex6_2_13	> 1000	> 1000	95.5	> 1000
ex6_2_14	2.07	> 1000	0.8	0.283
ex7_2_1	0.22	> 1000	> 1000	0.424
ex7_2_3	509	> 1000	> 1000	> 1000
ex7_2_4	37.0	5.37	> 1000	3.57
ex8_4_1	> 1000	214.4	0.4	0.76
ex8_4_2	> 1000	> 1000	> 1000	> 1000
gold	0.69	4.03	1.52	316.17
hart6	5.21	2.01	0.08	0.338
meanvar	0.03	1.253	0.532	0.863
Model13	0.13	50.37	0.08	6.701
process	0.61	0.66	0.62	0.34

TABLE 3.3.1: The solution times (CPU seconds) of the benchmarking problems are reported for each of the solvers in the comparison study. The relative standard error (RSE) of the three trials was less than 5% for all instance with total run time less than 0.5 seconds and less than 2% in all other instances.

Relative Gap at Termination				
Name	EAGO	SCIP	BARON	ANTIGONE
ex6_2_10	0.031	949.96	-	-
ex6_2_11	-	4.01E7	-	-
ex6_2_13	0.185	1783	-	0.2284
ex6_2_14	-	0.47	-	-
ex7_2_1	-	3.83	0.013	-
ex7_2_3	-	234.42	0.698	0.114
ex8_4_1	0.404	-	-	-
ex8_4_2	0.967	Inf	0.7812	0.496

TABLE 3.3.2: The relative gap remaining for each solver and benchmarking problem pair that did not converge to the desired tolerance within 1000 CPU seconds.

Benchmarking Problem Set Summary				
Name	Variables	Inequalities	Equalities	Nonlinear Terms
alkyl	15	0	7	$\times$, $(\cdot)^2$
BeckerLago	2	0	0	$(\cdot)^2$ $\sqrt{(\cdot)}$
ex2_1_8	24	0	10	$\times$
ex3_1_1	8	6	0	$\times$
ex4_1_9	2	2	0	$(\cdot)^2$, $(\cdot)^4$
ex5_4_3	16	13	0	$\times$, $(\cdot)/(\cdot)$, $(\cdot)^a$
ex6_2_10	6	0	3	$\times$, log, $(\cdot)/(\cdot)$
ex6_2_11	3	0	1	$\times$, log, $(\cdot)/(\cdot)$
ex6_2_13	6	0	3	$\times$, log, $(\cdot)/(\cdot)$
ex6_2_14	4	0	2	$\times$, log, $(\cdot)/(\cdot)$
ex7_2_1	7	14	0	$\times$, $(\cdot)/(\cdot)$, $(\cdot)^2$
ex7_2_3	8	6	0	$\times$, $(\cdot)/(\cdot)$
ex7_2_4	8	0	7	$\times$, $(\cdot)/(\cdot)$, $(\cdot)^a$
ex8_4_1	22	0	10	$(\cdot)^2$
ex8_4_2	24	0	10	$(\cdot)^2$
gold	2	0	0	$\times$, $(\cdot)^2$
hart6	6	0	0	exp$(\cdot)$, $\times$, $(\cdot)^2$
meanvar	8	0	2	$\times$
Model13	6	0	0	exp$(\cdot)$, $\times$, $(\cdot)^2$
process	10	0	7	$\times$, $(\cdot)/(\cdot)$, $(\cdot)^2$

TABLE 3.3.3: The selected benchmarking problems are summarized by their scale and complexity in terms of the number of variables, inequality and equality constraints, and types of nonlinear terms.

Chapter 4

Convex and Concave Envelopes of Artificial Neural Network Activation Functions for Deterministic Global Optimization

In this chapter, we present general methods to construct convex/concave relaxations of the activation functions that are commonly chosen for artificial neural networks (ANNs). The choice of these functions is often informed by both broader modeling considerations balanced with a need for high computational performance. The direct application of factorable programming techniques to compute bounds and convex/concave relaxations of such functions often lead to weak enclosures due to the dependency problem. Moreover, the piecewise formulation that defines several popular activation functions, prevents the computation of convex/concave relaxations as they violate the factorable function requirement. To improve the performance of relaxations of ANNs for deterministic global optimization applications, this study presents the development of a library of envelopes of the thoroughly studied rectifier-type and sigmoid activation functions, in addition to the novel self-gated sigmoid-weighted linear unit (SiLU) and

Gaussian error linear unit (GELU) activation functions. We demonstrate that the envelopes of activation functions directly lead to tighter relaxations of ANNs on their input domain. In turn, these improvements translate to a dramatic reduction in CPU runtime required for solving optimization problems involving ANN models to epsilon-global optimality. We further demonstrate that the factorable programming approach leads to superior computational performance over alternative state-of-the-art approaches.

4.1 Introduction

Machine learning and general surrogate modeling approaches provide a means to describe physical phenomena when accurate first-principles models (FPMs) may lead to intractable formulations and when field-specific knowledge may not be adequate to formulate accurate FPMs [145]. These approaches make use of either real-world data or computationally generated datasets to train data-driven models (DDMs) that adequately approximate the underlying system behavior. In many cases, the DDMs primarily serve to reduce intractable models into forms that allow for subsequent analysis, such as process design, sensitivity analysis, or an assessment of controllability. Optimization methods are often embedded in each of these tasks which makes the deterministic optimization of nonconvex models which embed these DDMs, a pursuit of interest.

Numerous data-driven modeling approaches have been applied to engineered systems that include: artificial neural networks (ANNs) [129, 264], Gaussian (Kriging) process models [53, 267, 332], and support vector machines [268]. ANNs, in particular, have seen greatly increased usage with the advent of widely-accessible and highly-capable software tools, such as Tensorflow [1] and Pytorch [230]. These universal approximators represent one class of DDMs that has seen an abundance of usage in the recent decades with optimization-based applications

ranging from synthesis of biodiesel processes [97], selection of optimal of fermentation media [201], optimal control of pressure swing absorption processes [15], design of cross-flow filtration systems [82], and many others [142, 211, 227]. More recently, there has been an emergence of interest in applying deep ANNs (usually defined as ANNs with four or more hidden layers in contrast to shallow ANNs that have a single hidden layer) to process system engineering applications as well as standard classification and ranking tasks. One recent example consists of using *deep* ANNs to predict multiphase flow characteristics in a pipe [273]. Renewed interest in deep ANNs has resulted in the exploration of novel ANN structures that may provide better performing models (lower computational cost, improved robustness, and better predictive value). Accompanying these investigations are efforts [69, 197, 316, 333] to uncover superior activation functions to be used in the more complex structures that characterize deep ANNs that hold the potential to reduce computational time and improve robustness relative to the state-of-the-art ReLU activation function.

Approaches that solve optimization problems deterministically with trained ANNs embedded, have largely been limited to ReLU network-based models. Full-space formulations may exploit the equivalency of ReLU networks to mixed-integer linear programs (MILP) [12, 103, 158, 309] or adapt ReLU network representations of piecewise linear functions to perform adaptive partitioning and domain tightening [115]. The resulting problems are then solved using state-of-the-art MILP (or MINLP solvers if nonlinear terms are present). Despite recent developments enabling deterministic global optimization of certain ANNs for process systems engineering applications [264, 266], there still remains a need for theoretical developments that enable support for a broader library of activation functions and additional families of ANNs. Namely, trained neural networks incorporating these activation functions may be described by systems of equations and in turn embedded in a mathematical program. Unfortunately, most activation functions embedded in this manner are nonlinear and exhibit

significant nonconvexities that lead to difficulties in solving the resulting optimization formulation.

Explicit consideration of activation functions may lead to tighter relaxations of nonconvex optimization problems involving ANNs. The availability of tighter relaxations remains desirable as their use may greatly accelerate the convergence of the branch-and-bound (B&B) algorithm that underlies all commercially available state-of-the-art deterministic global optimizers. Support for more general ANNs is also desirable for two other important reasons. First, it increases the variety of ANN model forms that may be developed for use in global optimization applications. Second, it allows for the integration of a broader family of legacy models built for general predictive purposes. These may be difficult to adapt to alternative surrogate model formulations due to domain-specific modeling considerations or potential logistical hurdles such as the unavailability of legacy training data. As such, methods that are directly applicable to these models are desirable. In this chapter, we make the following novel contributions that serve to address these outstanding issues:

1. We discuss ANN structures of current research interest and reduced-space reformulations for specialized ANN structures that may participate in global optimization formulations (the reader is directed to Section 4.4 for a summary of reduced-space formulations). In particular, we highlight a collection of activation functions without standard factorable representations using software libraries and categorize these according to convexity properties.

2. We derive novel convex/concave envelopes for the increasingly popular implicitly regularizing activation functions sigmoid-weighted linear unit (SiLU) and Gaussian error linear unit (GELU).

3. We analyze the convexity properties of numerous common activation functions and

highlight how naïve McCormick relaxations lead to overestimation of convex/concave relaxations due to the dependency problem (i.e., overestimation inherent to set-valued arithmetic due to the inability to recognize multiple occurrences of the same variable in a given expression [193, 255]).

4. We illustrate that the use of envelopes for common/popular activation functions leads to increased performance relative to a naïve application of composition rules originating from Garth McCormick's foundational work [177] using a randomly generated benchmark set.

These contributions ultimately lead to faster solution times associated with reduced-space optimization methods for a wide variety of ANN-based machine learning models. Moreover, our contributions allow for an extended library of activation functions to be utilized in nonlinear programs that must be solved with a certificate of global optimality.

In this chapter, we present new developments on the relaxation of activation functions common in recent ANN-based models. In Section 4.2, we detail pertinent neural network preliminaries. Subsequently, in Section 4.3, we develop and analyze convex and concave relaxations of several activation functions that have become increasingly prevalent in broader machine learning applications. In Section 4.4, we describe full-space and reduced-space formulations for global optimization problems; we then proceed to detail how ANNs may readily be incorporated into either formulation. In Section 4.5, we present the numerical results arising from a randomly generated benchmark set that illustrate the performance improvements readily achievable using these novel envelopes. Lastly, in Section 4.6, we reflect on current technical challenges and suggest future directions for subsequent research.

4.2 Artificial Neural Networks Preliminaries

One of the most common ANN structures is that of the MLP. The MLP is a class of feed-forward ANN that consists of a directed acyclic graph (DAG) containing n layers enumerated $k = 1, \ldots, n$. The first layer consists of inputs to the MLP. The subsequent $k = 2, \ldots, n-1$ layers are the *hidden layers*, with $k = n$ the *output layer*. The number of neurons in layer k is denoted $m^{(k)}$. Let $\mathbf{a}^{(k)} \in \mathbb{R}^{m^{(k)}}$ be the output vector of layer k. Accordingly, $\mathbf{a}^{(1)}$ is the input vector and $\mathbf{a}^{(n)}$ is the output vector of the MLP. For layers $k \in \{2, \ldots, n\}$, the vector $\mathbf{a}^{(k)}$ is defined componentwise by

$$a_i^{(k)} = f^{(k)}\left(\left(\mathbf{w}_i^{(k-1)}\right)^{\mathrm{T}} \mathbf{a}^{(k-1)} + b_i^{(k-1)}\right), \quad i = 1, \ldots, m^{(k)}, \tag{4.2.1}$$

where $f^{(k)} : \mathbb{R} \to \mathbb{R}$ are activation functions, $\mathbf{W}^{(k-1)} = \begin{bmatrix} \mathbf{w}_1^{(k-1)} & \mathbf{w}_2^{(k-1)} & \cdots & \mathbf{w}_{m^{(k)}}^{(k-1)} \end{bmatrix} \in \mathbb{R}^{m^{(k)} \times m^{(k-1)}}$ is a *weight matrix*, and $\mathbf{b}^{(k-1)} \in \mathbb{R}^{m^{(k)}}$ is a *bias vector*. For ease of introduction, we define $\mathbf{o} : \mathbb{R}^{m^{(1)}} \to \mathbb{R}^{m^{(n)}}$ to represent the input-output function for a generic DDM. In the case of an MLP, we have $\mathbf{a}^{(n)} = \mathbf{o}(\mathbf{a}^{(1)})$. A depiction of this type of network is provided in Figure 4.2.1. When a feedforward ANN is trained, the weight matrices and bias vectors become optimization variables while the values of the input vector $a_i^{(1)}$ for $i = 1, \ldots, m^{(1)}$, are treated as parameters. When a trained feedforward ANN is embedded in an optimization problem, the weight matrices and bias vectors are fixed to constant parameter values.

It should be noted that while the MLP structure may be readily decomposed into a factorable representation, many ANNs of active interest may not. For instance, residual networks and recurrent neural networks have analogous continuous-time representations [108, 118, 166, 250]. These continuous-time representations are often desirable as they may be evaluated using state-of-the-art ODE integrators and in turn circumvent the need to search for an optimal number of hidden layers. This has motivated recent interest in neural-ordinary differential equations

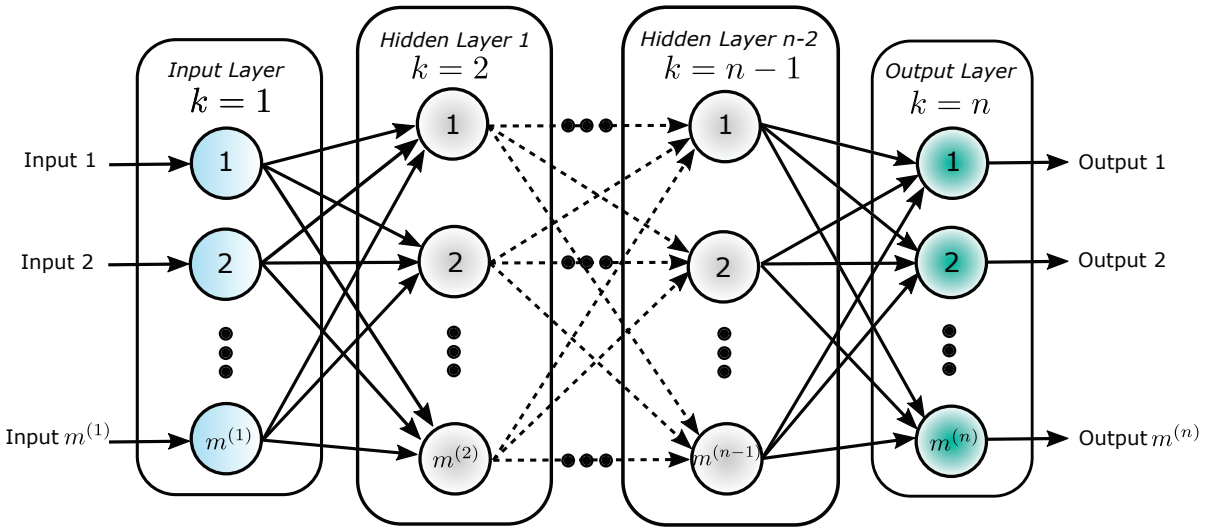

FIGURE 4.2.1: The directed acyclic graph representation of a multilayer perceptron with n layers. The input and output layers are the first ($k = 1$) and n-th ($k = n$) layers, respectively. The hidden layers are labeled $k = 2$ to $k = n - 1$. Note that the multilayer perceptron is a fully-connected network in which each neuron in layer $k - 1$ is connected to all neurons in the subsequent layer k.

[60, 238]. These models may be addressed by exploiting specialized continuous-time relaxation methods for ODEs [270, 272, 293, 328]. Alternatively, deep ANNs with an implicit representation [88] may require the solution of nonlinear systems via fixed-point methods in order to evaluate the neural network outputs and specialized relaxation methods for fixed-point methods need to be applied [300]. However, in both these cases, using tighter relaxations of the activation functions participating in the overall network will result in tighter relaxations of the overall network. Nonetheless, the methods described herein will be applicable to generalized feed forward neural networks [57], including deep feedforward networks [112], extreme learning machines [139], discrete recurrent neural networks [178], and deep residual networks [126] as activation functions represent a key component of each of these networks.

4.3 Relaxations of Activation Functions

To solve nonlinear programs with a certificate of global optimality, the use of a deterministic global optimization method (e.g., B&B) is required [136]. These methods typically require that convex (and concave) relaxations of the participating nonconvex expressions may be readily computed. After the seminal work of McCormick [177], there has been a significant effort to develop libraries of relaxations of common mathematical expressions and intrinsic functions that are often encountered in common mathematical models and relevant optimization problems. In this section, we further contribute to these efforts by developing convex and concave relaxations of common activation functions encountered in ANNs.

Previous work has focused on ANNs with a hyperbolic tangent [264, 265, 266] activation function. It was noted that increasing ANN depth is undesirable when holding the number of neurons fixed, due to the overestimation of relaxations [264]. In this section, we review relaxations of several common activation functions that have been developed more recently, with special attention paid to the Gaussian error linear unit (GELU) and the sigmoid-weighted linear unit (SiLU) functions. Many of these activation functions are known to perform better in a deep network configuration, specifically by either reducing the training time for equivalent network structures or by enabling networks with fewer terms that yield a more accurate fit to data. As a consequence of using these activation functions, the number of nonconvex expressions participating in a DDM may be reduced while maintaining the desired accuracy. This can lead to more tractable formulations of otherwise prohibitively expensive optimization problems.

We also note that the construction of relaxations of activation functions using standard McCormick libraries (e.g., [58, 329]) may often be inadequate. For example, in some cases, the piecewise definitions cannot be readily implemented via overloading approaches, and in other cases, the direct application of overloading approaches leads to weak relaxations. Note that this

applicability to broader classes of ANNs is desirable as it allows legacy models, built for long-term prediction, to be readily embedded in process simulations for optimization-based design tasks. We begin this section with a discussion of common families of activation functions and their associated convex/concave envelopes. Finally, we derive the envelopes for the recent SiLU and GELU activation functions that will be utilized in the case studies in this chapter.

4.3.1 Convex Activation Functions

One of the most ubiquitous activation functions currently used in machine learning is that of the Rectified Linear Unit (ReLU), $f : \mathbb{R} \to \mathbb{R} : x \mapsto \max(x, 0)$. The development of the ReLU represented a significant breakthrough in the supervised training of deep ANNs [155], and has since become the standard activation function used in deep learning. The simplicity of the ReLU in concert with the lack of any vanishing gradient issue, wherein nonlinearities lead to near-singular values of the input-output Jacobian, have often been noted as distinct advantages of this activation function. Additionally, networks comprised of ReLU functions may be readily modeled as MILPs [103] and solved using well-established MILP solvers. One proposed alteration to the ReLU resulted in the development of the exponential linear unit (ELU) which allows for negative outputs and avoids the "dying" ReLU problem — when ReLU neurons participating in a network only output 0 for any input during network training [165] — at the cost of significantly more complex arithmetic operations [62]. A scaled ELU was also proposed that avoids both vanishing and exploding gradient problems, and incorporates an internal normalization routine [153]. Many of these newly proposed activation functions belong to a family of monotonically increasing convex functions.

The convex envelope of a univariate scalar-valued convex function f on a compact set $[a, b]$ is simply the function itself. Whereas, its concave envelope is the affine function joining the

endpoints $f(a)$ and $f(b)$ [177]. Relaxations of compositions involving these terms can be computed using the Univariate McCormick Composition Theorem, Proposition 2.3.6 herein. While some activation functions such as the ReLU have exact convex/concave relaxations when computed via naïve McCormick composition [177, 191], others are defined as compositions of multiple algebraic expressions that lead to overestimation due to the well-known dependency problem. Functions such as ReLU consist of only a single expression and are already included in McCormick relaxation software libraries [58, 327, 329]. Other activation functions such as Maxsig, Maxtanh, and Softplus have relaxations that are weaker than their envelopes when computed using the algebraic expressions listed in Table 4.3.1, as illustrated in Figure 4.3.1, due to the dependency problem associated with the computation of relaxations of composite functions. Other convex activation functions, namely, ELU, SELU, and parametric ReLU, cannot be easily implemented using the frameworks of [177] and [191], as the conditional statement in the activation function definitions break the factorable function assumption inherent to these relaxation algorithms.

Two other considerations serve to motivate interest in alternatives to the ReLU function. First, the use of a twice-differentiable activation function is desirable as twice-differentiability is generally sufficient to ensure the second-order pointwise convergence of relaxations of ANNs [44]; a key consideration in mitigating the clustering problem present in deterministic global optimization algorithms [146]. While no function other than softplus listed in Table 4.3.1 is twice differentiable, one can reasonably expect a more significant degree of nondifferentiability to generally occur within nonsmooth activation functions as the nonsmooth behavior may arise from both the mid$(\cdot)$ operator present in the McCormick composition rule and the envelope itself. Second, the time spent calculating relaxations of the activation function may be minor when compared with the time spent in solving linear, mixed-integer, and convex nonlinear problems in any given iteration of the B&B algorithm. As such, it may be beneficial to compute tighter

Activation Function	Form $f(x)$	Source
ReLU	$\max\{0, x\}$	[202]
Parametric ReLU $0 < \alpha < 1$	$\begin{cases} x, & \text{if } x > 0 \\ \alpha x, & \text{otherwise} \end{cases}$	[125]
Maxsig (Let a be a root of $1 + \exp(-a) - a^{-1}$)	$\max(x, 1/(1 + \exp(-x)))$	[85]
Maxtanh	$\max(x, \tanh(x))$	[85]
Softplus	$\log(1 + \exp(x))$	[339]
Exponential Linear Unit (ELU) $\alpha > 0$	$\begin{cases} x, & x > 0 \\ \alpha(\exp(x) - 1), & x \leq 0 \end{cases}$	[62]
Scaled Exponential Linear Unit (SELU) $\lambda = 1.0507, \alpha = 1.67326$	$\lambda \begin{cases} x, & x > 0 \\ \alpha(\exp(x) - 1), & x \leq 0 \end{cases}$	[153]

TABLE 4.3.1: Convex activation functions and their first derivatives are defined in this table.

relaxations of slightly more computationally expensive terms if one may substantially reduce the overall number of iterations performed by the global solver and gain additional benefits associated with bounds tightening algorithms or other key heuristics [252, 305]. These considerations remain important for sigmoidal (convexoconcave) activation functions and novel self-gating activation functions.

4.3.2 Convexoconcave Activation Functions

Some of the earliest-used activation functions for ANNs are convexoconcave — a univariate function consisting of a convex region followed by a concave region. The preliminary use cases for convexoconcave activation functions primarily involve sigmoid and hyperbolic tangent

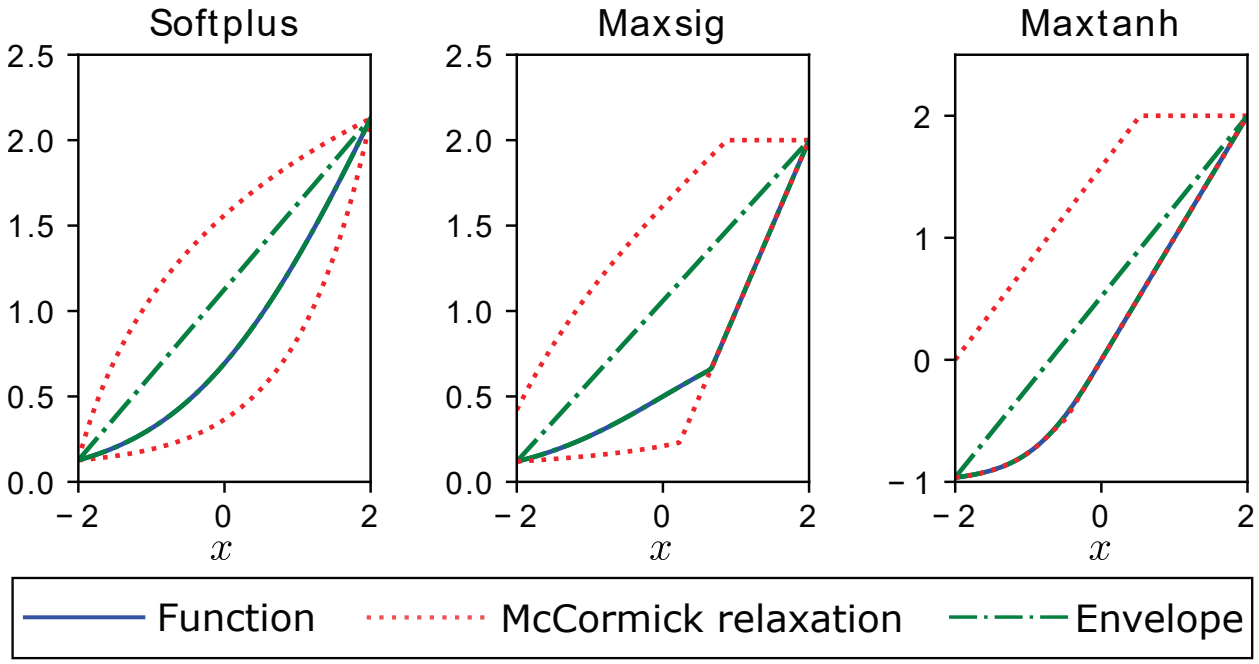

FIGURE 4.3.1: Relaxations of some common rectifier-like activation functions are weaker than their corresponding convex/concave envelopes when the relaxations are computed using the rules presented in [177]. The activation function, the relaxations of form $f(x)$ from Table 4.3.1 computed according to the rules presented in [177], and convex/concave envelope of the activation function, are shown in each subplot on the domain $x \in [-2, 2]$. These plots illustrate the overestimation of the classical McCormick relaxation approach compared to the envelopes for (**Left**) Softplus, (**Middle**) Maxsig, and (**Right**) Maxtanh functions.

activation functions [135] participating in shallow ANNs used for regression or classification tasks. As the sigmoid activation functions are bounded, inclusion of sigmoid layers may be used to constrain the range of ANN predictions. Continued investigations into sigmoid-shaped ANNs have focused on reducing the necessary computational time and minimizing the vanishing gradient problem, leading to forms such as the Softsign (i.e., ElliotSig) [91, 112]. Several equivalent algebraic forms of the hyperbolic tangent were examined by [264]. The authors noted that the direct application of McCormick composition rules leads to substantially weaker relaxations than the envelope for the majority of the alternative algebraic forms. We demonstrate here that these results hold for many other convexoconcave activation functions. While astute

Activation Function	Form $f(x)$	Derivative $f'(x)$
Softsign	$x/(1+\|x\|)$	$(1+\|x\|)^{-2}$
Hyperbolic tangent	$\tanh(x)$	$\operatorname{sech}^2(x)$
Penalized hyperbolic tangent	$\begin{cases}\tanh(x), & x > 0\\ \tanh(\alpha x), & x \le 0\end{cases}$	$\begin{cases}\operatorname{sech}^2(x), & x > 0\\ \alpha(\operatorname{sech}^2(\alpha x)), & x \le 0\end{cases}$
Sigmoid	$(1+\exp(-x))^{-1}$	$\exp(-x)(1+\exp(-x))^{-2}$
Bipolar sigmoid	$(1-\exp(-x))/(1+\exp(-x))$	$2\exp(x)(1+\exp(x))^{-2}$

TABLE 4.3.2: Convexoconcave activation functions and their first derivatives are defined in this table [85, 225].

usage of algebraic rearrangements may improve the quality of relaxations, such as the conversion of the bipolar sigmoid function to the equivalent $\tanh(x/2)$ form, this is not possible for all activation functions.

Convex/concave envelopes of convexoconcave activation functions can be computed using the rules described by [177] and [259]. A tie point x_m^{cv} is computed at which the function's derivative (provided the function is differentiable at x_m^{cv}) equals the slope of the secant line between $(f(x_m^{cv}), x_m^{cv})$ and $(f(x^U), x^U)$. Similarly, a tie point x_m^{cc} is computed at which the function's derivative (provided the function is differentiable at x_m^{cc}) equals the slope of the secant line between $(f(x_m^{cc}), x_m^{cc})$ and $(f(x^L), x^L)$. That is,

$$f^{cv,env}(x) = \begin{cases} f(x^U) + \frac{f(x^U)-f(x_m^{cv})}{x^U - x_m^{cv}}(x - x^U), & x \ge x_m^{cv} \\ f(x), & \text{otherwise} \end{cases} \tag{4.3.1}$$

$$f^{cc,env}(x) = \begin{cases} f(x^L) + \frac{f(x^L)-f(x_m^{cc})}{x^L - x_m^{cc}}(x - x^L), & x \le x_m^{cc} \\ f(x) & \text{otherwise.} \end{cases} \tag{4.3.2}$$

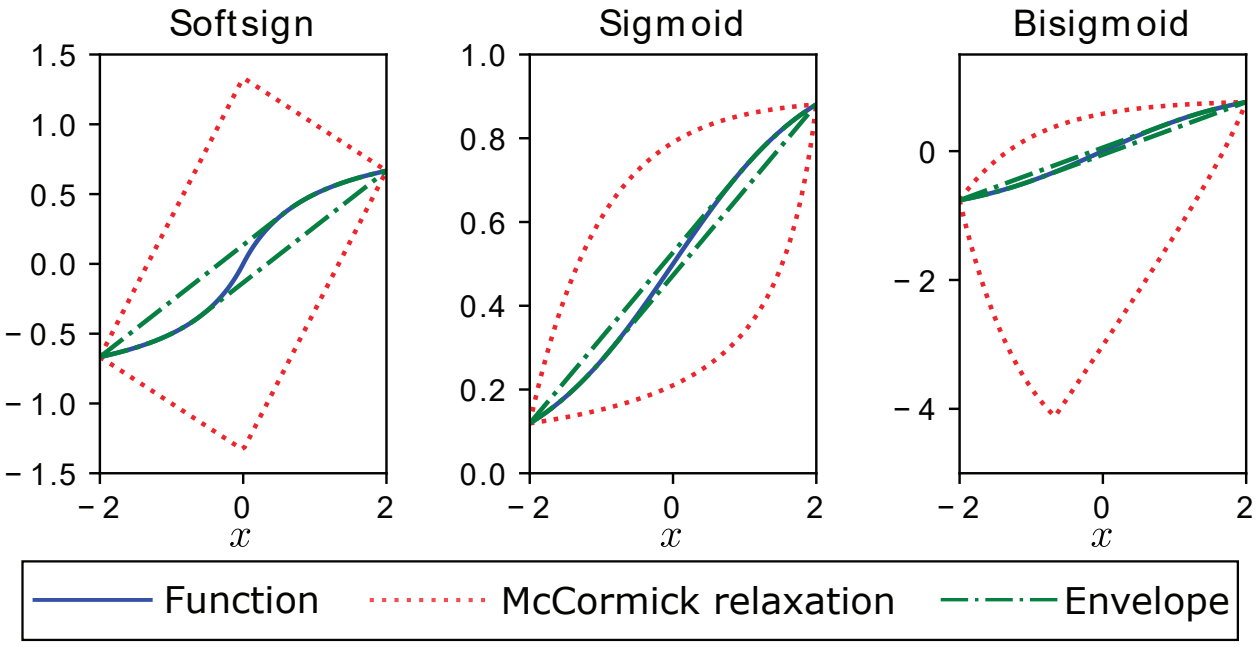

FIGURE 4.3.2: Many sigmoidal activation functions have relaxations that are weaker than their envelopes when computed using the rules presented in [177]. The activation function, the relaxations of form $f(x)$ from Table 4.3.2 computed according to the rules presented in [177], and convex/concave envelopes of the activation function are shown in each subplot on the domain $x \in [-2, 2]$. These plots illustrate the overestimation of the classical McCormick relaxation approach compared to the envelopes for (**Left**) Softsign, (**Middle**) Sigmoid, and (**Right**) Bisigmoid functions.

As illustrated in Figure 4.3.2, the envelopes yield substantially tighter relaxations than the direct application of McCormick composition rules [191] to the arithmetic expressions presented in Table 4.3.2.

4.3.3 Other Activation Functions

Models involving the ReLU function, as well as many of the convex activation functions presented here, lead to nondifferentiability, which may present issues for subsequent optimization and analysis. Moreover, sigmoid activation functions often suffer from a vanishing gradient issue when applied in deep ANNs. Two other activation functions have recently garnered significant

interest that are differentiable and avoid the vanishing gradient problem. The GELU function, denoted as gelu : $\mathbb{R} \to \mathbb{R}$, was developed as a means of merging stochastic dropout and zoneout procedures naïvely with the ReLU activation function [130]. The cumulative distribution function arising from the stochastic zero or identity transformation yields the gelu(·) function, defined as:

$$\text{gelu}(x) = \frac{x}{2}\left(1 + \text{erf}\left(\frac{x}{\sqrt{2}}\right)\right). \tag{4.3.3}$$

Another activation function of particular interest in the past few years that was originally purposed for use in reinforcement learning by [90], is the SiLU function, herein denoted silu : $\mathbb{R} \to \mathbb{R}$, and defined as:

$$\text{silu}(x) = \frac{x}{1 + \exp(-x)}. \tag{4.3.4}$$

It was noted that the silu(·) function exhibits a basic self-stabilizing property [89]. Akin to the gelu(·) function, the global minimum of the silu(·) function serves as an implicit regularizer that inhibits learning of large magnitude weights. Subsequent exploratory work by [243] used a reinforcement learning approach with a recurrent ANN to identify several potentially useful activation functions. They found the silu(·) activation function and provided strong numerical evidence that it outperforms a myriad of alternative activation functions.

Figure 4.3.3 illustrates the gelu(·) and silu(·) activation functions and their respective inflection points. We note that the convexity of silu(·) parallels that of gelu(·). Namely, each function has inflection points $x_{r,1}$ and $x_{r,2}$ that bound a region where they are convex. Outside of this region, the functions are concave. These inflection points occur at $\pm\sqrt{2}$ and approximately ± 2.39935 for gelu(·) and silu(·), respectively. Given a domain X, if $x_{r,1}, x_{r_2} \notin X$, then the function is either concave or convex on X and the envelope may be constructed as described in Section 4.3.1. If $x_{r,1} \notin X, x_{r,2} \in X$, then $f \in \{\text{gelu}, \text{silu}\}$ is convexoconcave on X and convex and concave

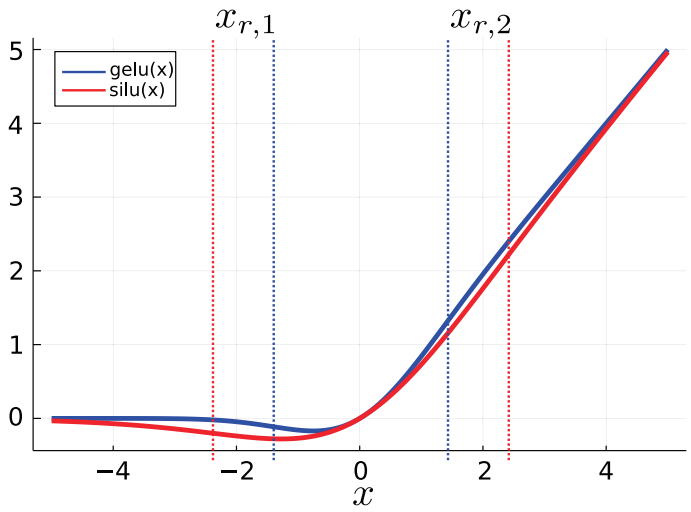

FIGURE 4.3.3: The gelu(·) and silu(·) activation functions are plotted on the domain $X = [-5, 5]$. The left and right inflection points, $x_{r,1}$ and $x_{r,2}$, respectively, are marked by the dashed vertical lines with the color corresponding to the respective function. Each function is concave on the domain $(-\infty, x_{r,1}]$, convex on the domain $[x_{r,1}, x_{r,2}]$, and concave on the domain $[x_{r,2}, +\infty)$.

envelopes may be computed from (4.3.1) and (4.3.2), respectively. If $x_{r,1} \in X, x_{r,2} \notin X$, then $g = -f$, with $f \in \{\text{gelu}, \text{silu}\}$, is convexoconcave and its envelope may be computed using (4.3.1) and (4.3.2), and then by applying the identity $(f^{cv,env}, f^{cc,env}) = (-g^{cc,env} - g^{cv,env})$. We now proceed to derive the convex and concave envelopes of gelu(·) and silu(·) on X, with $x_{r,1}, x_{r,2} \in X$ in the following Theorem 4.3.1.

Theorem 4.3.1. (Convex/Concave Envelopes of gelu(·), silu(·)) Let $f : \mathbb{R} \to \mathbb{R}$ be defined as either gelu(·) or silu(·), as defined in (4.3.3) and (4.3.4), respectively. Let $x_{r,1}, x_{r,2} \in X \in \mathbb{R}$ be the inflection points of $f(\cdot)$ such that $x_{r,1} < x_{r,2}$ and let x_{min} be the point at which f attains its minimum on $\mathbb{R}$. Let $f^{cv}, f^{cc} : X \to \mathbb{R}$ denote convex and concave relaxations of f on X,

respectively. Then, for each $x \in X$, $f^{cv}(x)$ is given by

$$f^{cv}(x) = \begin{cases} f(x^L) + \frac{f(x^{cv}_{m,1}) - f(x^L)}{x^{cv}_{m,1} - x^L}(x - x^L) & x < x^{cv}_{m,1} \\ f(x) & x^{cv}_{m,1} \leq x < x^{cv}_{m,2} \\ f(x^{cv}_{m,2}) + \frac{f(x^U) - f(x^{cv}_{m,2})}{x^U - x^{cv}_{m,2}}(x - x^{cv}_{m,2}) & x^{cv}_{m,2} \leq x \end{cases} \tag{4.3.5}$$

where the tie points $x^{cv}_{m,1} \in [x_{r,1}, x_{min}]$ and $x^{cv}_{m,2} \in [x_{min}, x_{r,2}]$ are points that respectively satisfy the following:

$$f(x^{cv}_{m,1}) - f(x^L) - (x^{cv}_{m,1} - x^L) f'(x^{cv}_{m,1}) = 0, \tag{4.3.6}$$

$$f(x^U) - f(x^{cv}_{m,2}) - (x^U - x^{cv}_{m,2}) f'(x^{cv}_{m,2}) = 0. \tag{4.3.7}$$

Similarly, for each $x \in X$, $f^{cc}(x)$ is given by

$$f^{cc}(x) = f(x^L) + \frac{f(x^U) - f(x^L)}{x^U - x^L}(x - x^L). \tag{4.3.8}$$

Proof. Note that under the hypothesis $x_{r,1}, x_{r,2} \in X$ and the convexity/concavity properties of $f(\cdot) \in \{\text{gelu}(\cdot), \text{silu}(\cdot)\}$ on X, the envelopes of f are required on a domain consisting of three adjoining regions where f is concave ($R_1 = [x^L, x_{r,1}]$), convex ($R_2 = [x_{r,1}, x_{r,2}]$), and concave ($R_3 = [x_{r,2}, x^U]$), such that $X = \bigcup_i R_i$. First, we note that both gelu$(\cdot)$ and silu$(\cdot)$ are twice differentiable. Beginning with the convex envelope, we note that f is monotonically decreasing and since $x_{min} > x_{r,1}$, $f(x) > f(x_{min})$ for all $x \in R_1$. As a consequence, there exists a tie point $x^{cv}_{m,1} \in [x_{r,1}, x_{min}]$ satisfying (4.3.6) such that a secant line may be constructed between $(x^L, f(x^L))$ and the point $(x^{cv}_{m,1}, f(x^{cv}_{m,1}))$. This secant line is defined by the first case of (4.3.5) and is the convex envelope of f on R_1 and part of R_2. Similarly, since $x_{min} < x_{r,2}$ and f is monotonically

increasing on R_3, $f(x) > f(x_{min})$ for all $x \in R_3$. Thus, there exists a tie point $x^{cv}_{m,2} \in [x_{min}, x_{r,2}]$ satisfying (4.3.7), such that a secant line may be constructed between $(x^{cv}_{m,2}, f(x^{cv}_{m,2}))$ and the point $(x^U, f(x^U))$. This secant line is defined by the third case of (4.3.5) and is the convex envelope of f on R_3 and part of R_2. For $x \in [x^{cv}_{m,1}, x^{cv}_{m,2}]$, $f(x)$ is convex and therefore is trivially its own convex envelope on this subdomain. Thus, (4.3.5) defines the convex envelope of f on X. Next, we consider the concave envelope. Since f is monotonically decreasing on $[x^L, x_{min}]$ and monotonically increasing on $[x_{min}, x^U]$, the concave envelope is defined by the secant line connecting the endpoints $(x^L, f(x^L))$ and $(x^U, f(x^U))$, defined by (4.3.8). □

We now proceed to examine the convergence behavior of activation function envelopes and contrast these with naïve McCormick calculations.

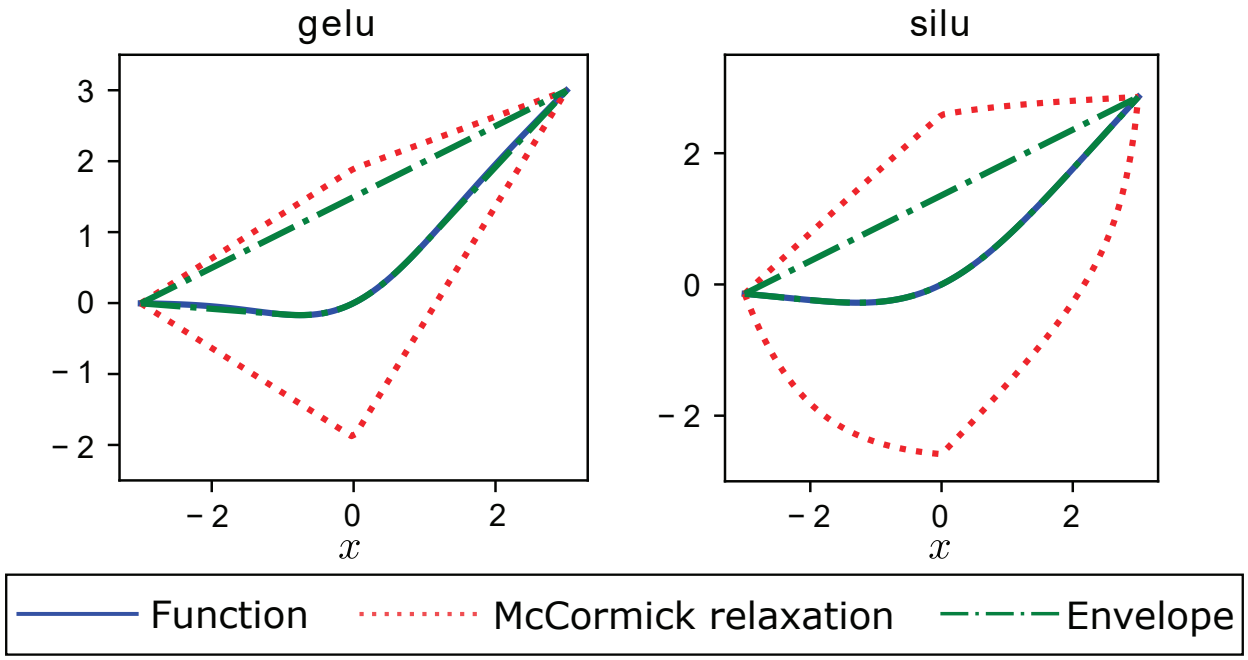

FIGURE 4.3.4: **Left:** The Gaussian error linear unit (gelu(·)) function , the convex/concave relaxations of $x\left(1 + \text{erf}\left(x/\sqrt{2}\right)\right)/2$ computed according to the rules presented in [177] and the convex/concave envelope of the gelu function are plotted on $x \in [-3, 3]$. **Right:** The sigmoid-weighted linear unit (silu(·)) function, the convex/concave relaxations of $x/(1 + \exp(-x))$ computed according to the rules presented in [177] and the convex/concave envelope of the silu function are plotted on $x \in [-3, 3]$.

4.3.4 Convergence Properties of Convex/Concave Relaxations of Activation Functions

We now compare the relative performance of naïve McCormick relaxations versus the newly defined envelopes of the library of activation functions: Softplus, Maxsig, Maxtanh, Softsign, Sigmoid, Bisigmoid, SiLU, and GELU. Figure 4.3.1, Figure 4.3.2, and Figure 4.3.4 visualize the relative tightness of McCormick relaxations versus the envelopes. Here, we quantify the performance of the proposed envelopes versus the naïve McCormick relaxations using two analyses. First, the relative computational times are measured for constructing convex and concave relaxations on a domain and evaluating these relaxations and subgradients at a single point. The second analysis compares the relaxations of prototypical ANNs of varying depth under a width metric. Relaxations of each activation function have been implemented in the McCormick.jl [329] subpackage of the author's optimization package EAGO.jl [327], and is openly available.

The computational time comparison was conducted using the BenchmarkTools.jl[59] package in Julia v1.6.2 with the default settings. For each benchmark run, 10^4 samples were used with an automatically-generated number of expression evaluations per sample (chosen by the package for accurate timings). Relaxations of each activation function on the domain $X = [-3, 3]$ were computed using the implemented envelopes as well as applying the standard McCormick rules to the algebraic forms of each activation function. This domain was chosen as it encloses the inflection points of all activation functions considered.

The timing results for calculating envelopes are recorded in Table 4.3.3 as percentages relative to naïve McCormick relaxations. In most cases, the envelopes are significantly more computationally expensive to calculate. This is because the tie points x_m^{cv}, x_m^{cc} must be calculated to construct the envelopes, which in turn requires the solution of nonlinear algebraic equations.

Function	Softplus	Maxsig	Maxtanh	Softsign	Sigmoid	SiLU	GELU
envelope (μs)	0.172	0.134	23.1	0.919	0.898	1.41	249
naïve (μs)	0.158	0.384	14.9	0.0199	0.221	0.189	135
τ (%)	8.74	-65.2	54.8	4519	306	643	83.9

TABLE 4.3.3: The costs of calculating convex and concave relaxations and corresponding subgradients of the considered activation functions are tabulated for the newly defined envelopes and naïve McCormick relaxations. The absolute CPU times (μ s) and relative times (%) τ are reported. For almost all activation functions in this table, the envelope calculations are more expensive (and sometimes significantly) due to the necessity of calculating the tie points.

Softplus and Maxsig are exceptions because they are convex on X, and thus their envelopes are trivial and less expensive to calculate than naïvely applying the McCormick composition rules. However, the increased CPU time required to compute envelopes is still considered small when compared to the CPU time requirements for the other subroutines encountered in global optimization, such as optimization-based bounds tightening, the exact solution of MILPs, and local solution of nonlinear programs, which are all bottlenecks in the solution of global optimization problems. Moreover, it is well-known that the use of tighter relaxations accelerates node fathoming, and therefore, reduces the number of subproblems that must be considered. For complex nested subexpressions, such as ANNs, the gains achieved may be substantial. We demonstrate the tightness conferred to the relaxations of ANNs by the use of envelopes using the illustrative numerical example below.

For the analysis of relaxations of ANNs on an input domain, a simple MLP is generated with two hidden layers, each with a single type of activation function $f^{(k)}(\cdot) \in \{\text{gelu}(\cdot), \text{silu}(\cdot)\}$, and a fully-connected affine layer defining a single output value (i.e., $f_{mlp} = a^{(4)} = o(a^{(1)})$). Ten neurons are included in each layer and the weights and bias values were randomly selected from a uniform distribution between 0 and 1. To simplify the calculations, we set the input variable $a^{(1)}$ to $p \in P = [-\delta p, \delta p]$ and allowed δp to vary by factors of 10 from 10^{-1} to 10^{-4}. We consider two

distinct metrics used to assess the tightness of relaxations. First, we consider the maximal pointwise distance from the convex relaxation to the concave relaxation, $\mathcal{P}(P) = \max_{p \in P} \left(f^{cc}_{mlp}(p) - f^{cv}_{mlp}(p) \right)$, which relates to the pointwise convergence properties of McCormick relaxations. Secondly, we consider a tightness metric for convex and concave relaxations, $\mathcal{H}(P) = \text{diam}\left(\left[\min_{p \in P} f^{cv}_{mlp}(p), \max_{p \in P} f^{cc}_{mlp}(p) \right] \right)$, which relates to the Hausdorff convergence properties of McCormick relaxations as discussed in [203]. We refer the reader to the original work for a full discussion of the underlying theory that motivates this metric.

The use of the envelopes improves the tightness of the derived relaxations of the MLP over naïve McCormick relaxations as illustrated in Figure 4.3.5. Under each distance metric, the relaxations of most activation functions exhibit quadratic convergence; a property required to avoid the clustering problem in spatial B&B [83, 323]. Further, as expected, the convex and concave envelopes of the activation functions result in a significant reduction in overestimation of the relaxations of the MLP. This improvement is most apparent under the $\mathcal{P}(P)$ metric. As previously hypothesized, despite the additional computational cost required to calculate the envelopes, it is expected that the reduction in overestimation will significantly speed up convergence of the B&B algorithm for deterministic global optimization of models containing ANNs.

4.4 Global Optimization of ANNs

Optimization problems with ANN models [264] are formalized in this section. The vector of input variables is defined as $\mathbf{x} \in X \subseteq \mathbb{R}^{n_x}$, and the vector of output variables of an ANN is defined as $\mathbf{z} \in Z \subset \mathbb{R}^{n_z}$. $\mathbf{h} : X \times Z \to \mathbb{R}^{n_z}$ represents the general nonlinear network equations governed by an ANN model. $\mathbf{g} : X \times Z \to \mathbb{R}^{n_z}$ represents the inequalities that constrain the feasible region.

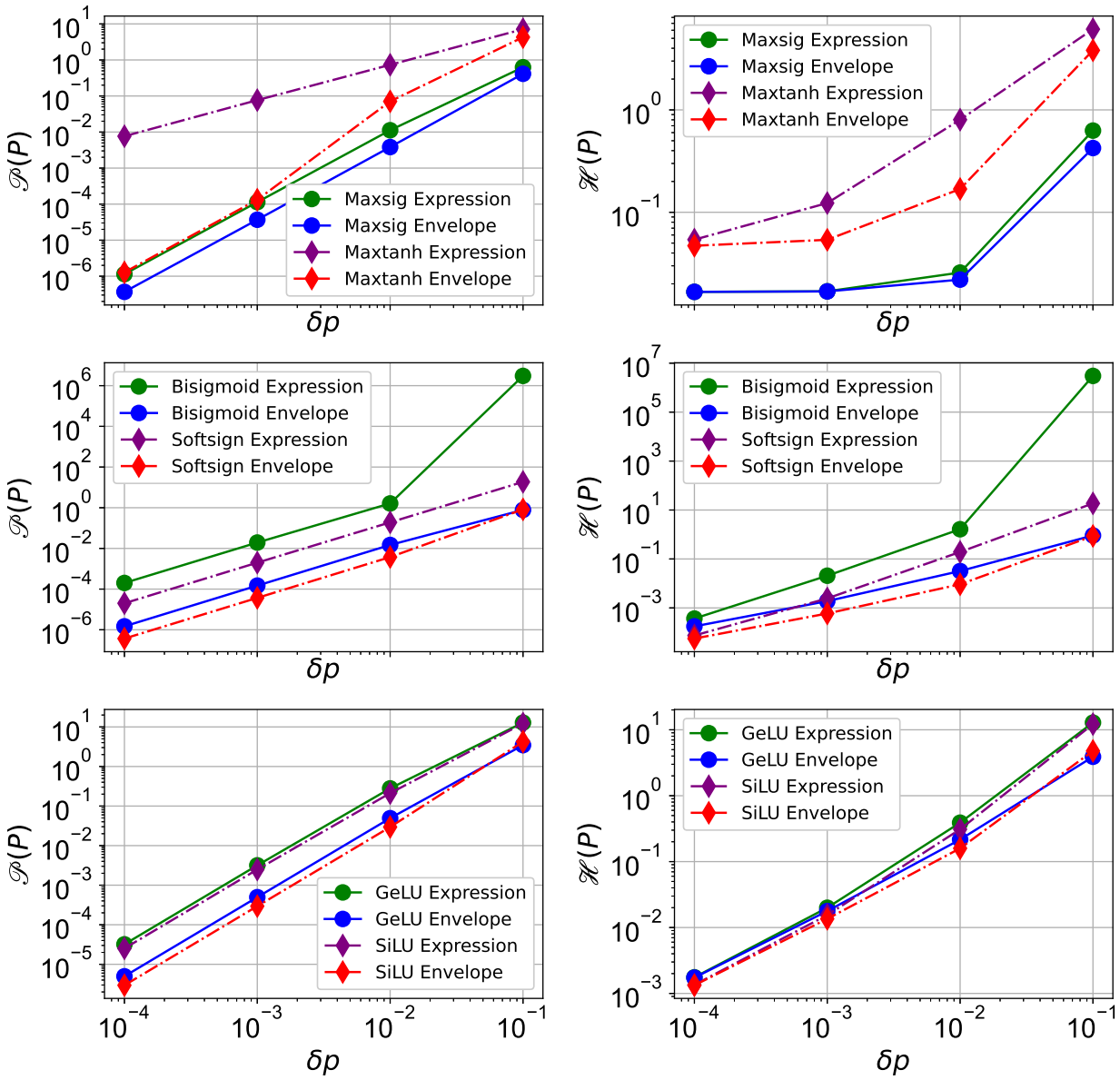

FIGURE 4.3.5: A comparison of the tightness of the envelopes of activation functions is made against relaxations derived using the naïve McCormick relaxation approach. **Left:** pointwise convergence behavior is illustrated using the $\mathcal{P}(P)$ metric and **right:** the behavior under the relaxation tightness metric $\mathcal{H}(P)$, is shown. Under each metric, the relaxations generated using the envelopes of the activation functions outperform naïve McCormick relaxations. This comparison is made for (**top**) characteristic convex, (**middle**) convexoconcave, and (**bottom**) for the newer SiLU and GELU activation functions with relaxations from Theorem 4.3.1.

$\phi : X \times Z \to \mathbb{R}$ represents the objective function. Generally, a *full-space* formulation denotes that the equality constraints governed by a system model are directly embedded. The formulation with ANNs embedded can be expressed as:

$$\begin{aligned} \min_{\mathbf{x}\in X, \mathbf{z}\in Z} \ & \phi(\mathbf{x}, \mathbf{z}) \\ \text{s.t.} \ & \mathbf{h}(\mathbf{x}, \mathbf{z}) = \mathbf{0} \\ & \mathbf{g}(\mathbf{x}, \mathbf{z}) \leq \mathbf{0}. \end{aligned} \tag{4.4.1}$$

As introduced in Section 4.2, in a MLP, the network governing equations $\mathbf{h}(\mathbf{x}, \mathbf{z}) = \mathbf{0}$ can be calculated as an explicit input-output form: $\mathbf{z} = \mathbf{o}(\mathbf{x})$. Thus, the equality constraints can be eliminated and (4.4.1) can be reformulated to a *reduced-space* form:

$$\begin{aligned} \min_{\mathbf{x}\in X} \ & \phi(\mathbf{x}, \mathbf{o}(\mathbf{x})) \\ \text{s.t.} \ & \mathbf{g}(\mathbf{x}, \mathbf{o}(\mathbf{x})) \leq \mathbf{0}. \end{aligned} \tag{4.4.2}$$

Reduced-space methods with respect to deterministic global optimization originated in [93], where details were introduced for a method using a B&B algorithm with only a subset of the decision variables being branched on. This approach was further generalized for broader classes of model structures (e.g., [48, 191, 296, 322]). The core idea of a *reduced-space* method is to treat the vector of independent input variables $\mathbf{x}$ as the only decision variables of the optimization problem by eliminating the equality constraints and therefore the explicit dependence of the problem on auxiliary variables through intermediate computation and compositions.

In deterministic global optimization, a variation of the spatial B&B algorithm can be applied to solve nonconvex problems with formulations of (4.4.1) and (4.4.2) [136, 252, 305]. The B&B

algorithm iteratively partitions the decision space into successively smaller subdomains and solves a sequence of upper-bounding and lower-bounding problems on each subdomain. Upper bounds are typically determined by solving nonconvex optimization problems in subdomains to either feasibility or local optimality. Lower bounds rely on convex and concave relaxations of the objective and constraints. As the algorithm proceeds, the best-found bounds are saved for comparison. By comparing the obtained upper and lower bounds, the algorithm converges to an ϵ-optimal global solution in finitely-many iterations, or a certification of infeasibility.

Many global optimizers, such as BARON [253, 305] and ANTIGONE [185], may introduce auxiliary variables to any of the formulations (4.4.1) when constructing subproblems, resulting in excessively large dimensionality subproblems with a high time cost due to the curse of dimensionality. This is especially true for ANN models as they include multiple layers, neurons, and network equations that inherently results in a large-scale optimization problem. In contrast, the EAGO [327] and MAiNGO [46] global solvers allow for the construction of relaxations directly from (4.4.2), which has been shown to reduce the computational complexity of solving subproblems and dramatically decreases solution time costs on several examples [48, 93, 148, 151, 152, 264, 300].

4.5 Numerical Experiments

We now illustrate how the use of envelopes described herein leads to a reduction in CPU run time and allows for a large variety of problems to be solved to deterministic global optimality. This is done by comparing the methods on a randomly generated benchmark library using the approach presented by Dolan and Moré [80]. The *performance* of a solver configuration s is taken to be the solution time $t_{p,s}$ in CPU seconds (single-threaded) for problem p. We consider the

performance ratio on problem p by solver s to be the ratio of solver s performance to the best solver performance in the set:

$$r_{p,s} = \frac{t_{p,s}}{\min\{t_{p,s} : s \in S\}}.$$

The *performance profile* of solver s on a particular benchmark set depicts the distribution function of the performance metric, $\rho_s(\tau)$; the probability that a performance ratio $r_{p,s}$ is within a factor $\tau \in \mathbb{R}$ of the best possible ratio

$$\rho_s(\tau) = \frac{1}{n_p}\text{size}\{p \in \mathcal{P} : r_{p,s} \leq \tau\},$$

where $\mathcal{P}$ is the set of problems with $n_p = \text{card}(\mathcal{P})$. A plot comparing r_s for each solver configuration $s \in S$ will then illustrate the relative performance. For problems that terminate due to the specified time limit, the relative gap remaining can be compared to assess solver performance. The relative gap remaining is given by (`U` - `L`)/ max(`U`, `L`) where `U` is the upper bound (best feasible objective value) and `L` is the lower bound.

As the focus of this chapter is the development of global optimization algorithms—and to the best of our knowledge, no benchmark library of ANN-embedded global optimization test problems exists— we choose to utilize a randomly generated library of 100 MLPs. Since the following analysis is motivated by a desire to compare the computational results between solvers and relaxations methods, we may conceptualize each random MLP as an ideally-trained MLP for some arbitrary function that serves to extricate the role of the optimization method from the confounding effects of surrogate model structure and training methodology. Weights and bias values are randomly assigned values sampled from a uniform distribution within $[-1, 1]$. The number of decision variables participating in the MLP, the number of layers, and the number of

Attribute	Values
Number of Input Variables	2 to 5
Number of Hidden Layers	1 to 4
Number of Neurons per Layer	2 to 5

TABLE 4.5.1: The range of values for each metaparameter used to generate the instances in the benchmark set are listed here.

neurons per layer for each instance are chosen randomly from a uniform distribution within the ranges listed in Table 4.5.1. The objective function considered is simple summation (4.5.1):

$$\phi(\mathbf{x}, \mathbf{o}(\mathbf{x})) = \sum_i \mathbf{o}_i(\mathbf{x}). \tag{4.5.1}$$

The only constraints present in each problem are taken to be the box constraints on the decision variables, x_i, such that $x_i \in [-1, 1]$ for $i = 1, \ldots, n$.

4.5.1 Implementation

All numerical experiments in this work were run on a single thread of an Intel Xeon E3-1270 v5 3.60/4.00GHz (base/turbo) processor with 16GB ECC RAM allocated to a virtual machine running the Ubuntu 18.04LTS operating system and Julia v1.6.2 [36]. Absolute and relative convergence tolerances for the B&B algorithm of 10^{-4} were specified for all example problems along with a maximum CPU time limit of 15 minutes (900 seconds). The EAGO.jl v0.7.0 package [327] was used to solve each optimization problem. Relaxations based on the envelopes of each activation function were implemented in the McCormick.jl [329] subpackage of EAGO.jl and are openly available. Validated interval arithmetic was computed using the package

IntervalArithmetic.jl [260]. The Intel MKL package (2019 Update 2) [95] was used to perform all LAPACK [11, 320] and BLAS [40] routines. BARON v21.1.13 [253, 305] and SCIP 7.0.3[313] were used for performance comparisons. The data used with and generated from the following numerical examples are openly available in the Git repository established at `https://github.com/PSORLab/RSActivationFunctions` along with the corresponding problem formulations.

4.5.2 Benchmark Results

We now examine the impact of the envelopes on the solution times of a reduced-space global optimizer for solving deterministic global optimization problems with ANN models. The algebraic expressions for all activation functions considered in this study can be found in Tables 4.3.1 and 4.3.2, and Equations 4.3.3 and 4.3.4. Computational experiments were then conducted in which global optimization problems were solved for the benchmark suite of ANNs. To compare the performance of using the developed library of envelopes, the optimization problems were solved using both naïve McCormick relaxations (EAGO - McCormick) as well as the envelopes (EAGO - Envelope). These configurations are then compared to a state-of-the-art open-source solver, SCIP [313], and the state of the art commercial solver BARON [253, 305]. This comparison was made as SCIP, like EAGO, is open-source but implements the auxiliary variable method to constructing polyhedral relaxations used to compute lower bounds and refine the problem domain [313]. As SCIP does not support nonlinear objectives directly, a variable q was introduced and the problem was recast via the epigraph reformulation:

$$\min_{\mathbf{x}\in X, q\in Q} \; q \tag{4.5.2}$$
$$\text{s.t.} \;\; \sum_i \mathbf{o}_i(\mathbf{x}) \leq q.$$

We note that EAGO performs this reformulation automatically as a preprocessing step. MLPs corresponding to sigmoid, softplus, silu, and gelu activation were considered. As illustrated by the performance profile plot depicted in Figure 4.5.1, EAGO generally outperforms SCIP on this limited benchmark set and the use of the activation function envelopes developed herein further improves computational performance, as evidenced by the the increased number of problem solved within the 15 minute limit as detailed in Table 4.5.2. As shown in Tables 4.5.3 and 4.5.4, the use of envelopes in EAGO categorically increases the number of problems solved within 15-minute for each activation function and further decreases the 15 minutes average relative gap remaining for unsolved problems. Moreover, the shifted geometric mean solve times shown in Table 4.5.5 are reduced by using the envelopes presented herein.

While EAGO outperforms BARON with the new envelope functions for sigmoid and silu, we see significantly higher shifted geometric mean solve times for EAGO than BARON for softplus. While BARON is not an open-source solver, we may speculate on the potential reasons. We note that the median solve time for softplus problems is 0.2 CPU seconds, indicating that the geometric mean solve time is highly influenced by high solution time instances. The softplus activation function (also termed a "logistic loss" function) is known to have an epigraph that may be represented by the exponential cone and a disciplined convex programming approach used in BARON's presolve phase indicates that softplus($\mathbf{a}^{\mathrm{T}}\mathbf{x} + b$) is itself convex [113, 149]. Taken together, these two factors indicate that BARON's presolve convexity detection may allow it to derive tighter bounds automatically. This highlights the potential that improved automatic

Solver	Solved	Unsolved
EAGO (Envelope)	280 (93.3%)	20 (6.7%)
EAGO (Naïve McCormick)	260 (86.7%)	40 (13.3%)
SCIP	240 (80.0%)	60 (20.0%)
BARON	273 (91.0%)	27 (9.0%)

TABLE 4.5.2: The number of problems solved within 15 minutes by solver configuration in the benchmark set, are tabulated. Only sigmoid, softplus, and silu functions are used in these calculations for fair comparison since gelu is unsupported by both SCIP and BARON. EAGO and the developed envelopes outperform all other configurations in total problems solved.

convexity detection may have to further mitigate the overestimation of activation function relaxations as compared to the naïve McCormick approach.

We note that for a select number of poorly scaled problems, either BARON, EAGO, or SCIP may incorrectly terminate with a certificate of infeasibility. This may be attributed to the rounding errors encountered when computing polyhedral relaxations from convex and concave relaxations and their associated subgradients. EAGO currently makes use of a heuristic approach to ensure only numerically-safe affine relaxations are added to subproblems. BARON v21.1.13 makes use of a state-of-the-art adaptive approach to resolve numerical difficulties. In either case, neither of these approaches are sufficient to fully resolve all numerical issues occurring in the benchmark set. Accordingly, additional work on resolving these numerical issues remains an active area of research. However, in spite of this observation, we can see that the use of envelopes in this context leads to categorically improved computational performance relative to the naïve McCormick calculations.

Activation Function	EAGO (Envelope)	EAGO (McCormick)	SCIP	BARON
sigmoid	96 (4)	94 (4)	91 (5)	94 (4)
softplus	93 (7)	88 (7)	85 (8)	92 (7)
silu	91 (0)	78 (0)	64 (1)	87 (0)
gelu	76 (0)	59 (0)	N/A	N/A

TABLE 4.5.3: The number of benchmark problems solved within the 15-minute time limit are listed by solver configuration and activation function. The number of problems returning an infeasible result are given in parentheses for each condition. EAGO and the developed envelopes outperform all other configurations in total problems solved for each activation function.

	EAGO (Envelope)	EAGO (McCormick)	SCIP	BARON
sigmoid	N/A	2.4×10^{-3}	5.2×10^{1}	2.5×10^{-2}
softplus	N/A	1.0×10^{-1}	7.3×10^{1}	8.9×10^{1}
silu	1.7×10^{-1}	9.9×10^{0}	2.1×10^{1}	4.2×10^{1}
gelu	9.5×10^{-1}	3.3×10^{1}	N/A	N/A

TABLE 4.5.4: The average relative gap remaining for any problems not solved within the 15-minute time limit are listed by solver configuration and activation function. For each activation function, EAGO with the developed envelopes have significantly smaller relative gaps at the 15-minute limit.

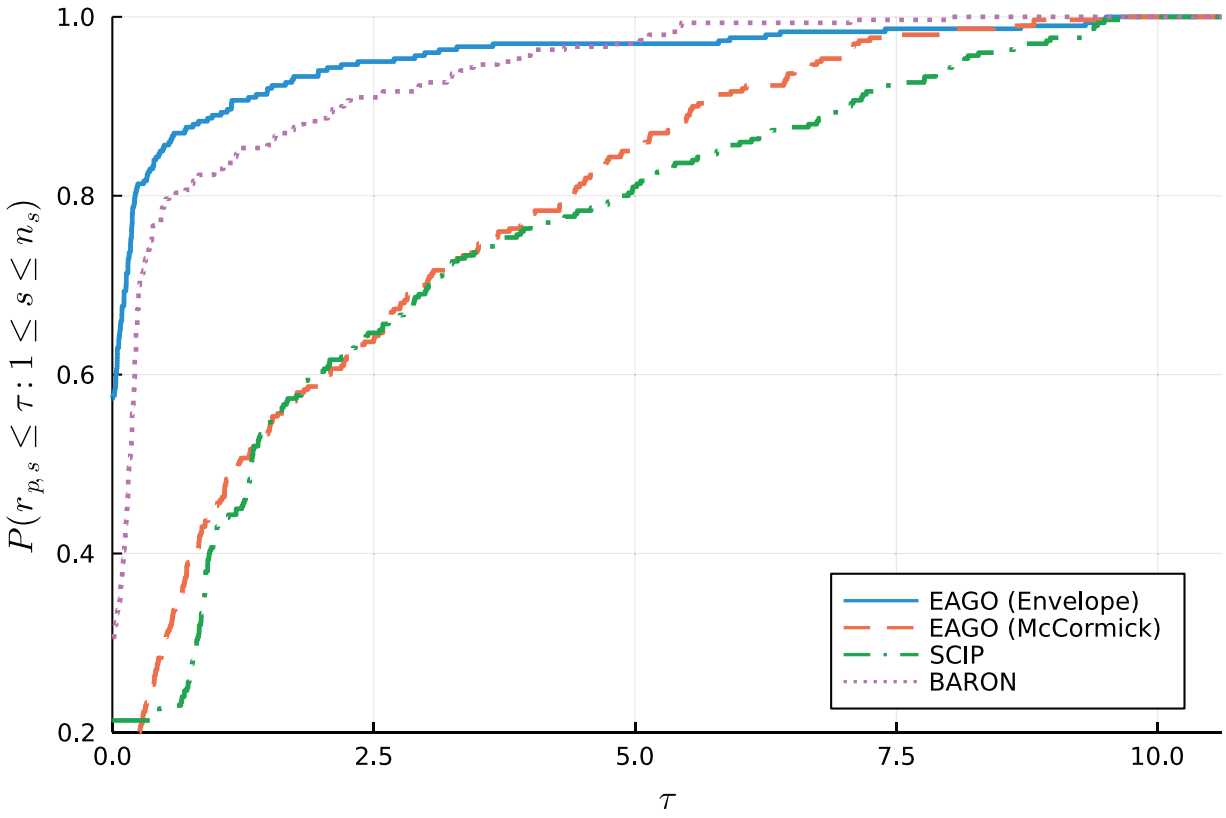

FIGURE 4.5.1: As illustrated by the performance profiles shown for each solver configuration, computing relaxations using the envelopes leads to a substantial decrease in CPU solution time for a typical problem within the benchmark set when compared to either SCIP or the naïve McCormick approach implemented in EAGO. The EAGO (envelope) calculations (blue-solid) also outperform BARON for many activation function types, which is evident from the superior performance for $\tau < 5$.

4.6 Concluding Remarks

In this chapter, the convex/concave envelopes were developed for a variety of activation functions commonly used in ANN surrogate models. These included an identification of several activation functions that have not been previously implemented in existing libraries of relaxations and lack a general factorable form. Moreover, the McCormick arithmetic approach was shown to lead to weak relaxations for several common activation functions that have factorable representations. Particular attention was paid to the development of envelopes for the novel SiLU and GELU

	EAGO (Envelope)	EAGO (McCormick)	SCIP	BARON
sigmoid	1.09	4.16	6.37	1.54
softplus	2.43	7.19	7.69	1.36
silu	5.04	36.77	43.11	9.58
gelu	19.62	82.69	N/A	N/A

TABLE 4.5.5: The shifted geometric mean of solve times $t_1, t_2, \ldots, t_n$ defined by $(\prod_{i=1}^{n}(t_i + s))^{1/n} - s$ are given by solver configuration and activation function with $s = 1$. EAGO using the envelopes developed herein outperforms naïve McCormick and SCIP on all activation functions examined. However, BARON outperforms all configurations examined for the Softplus function

activation functions that do not belong in standard convexity classes (convex, convexoconcave, etc.) that have been previously addressed. Further, we demonstrated that the use of these envelopes provides desirable Hausdorff and pointwise convergence properties for the relaxations of the underlying activation functions.

Lastly, we generated benchmark results with respect to the performance of these relaxations when incorporated into a reduced-space global optimization routine using the EAGO optimizer. Using a randomly generated benchmark set of MLPs, we illustrated that the use of envelopes leads to a substantial reduction in run time and this reduced-space approach outperforms the full-space approach implemented in SCIP, leading to solving 13.3% more benchmark problems solved within the specified time limit. Moreover, this approach is comparable to the performance of the state-of-the-art commercial solver BARON. As such, the use of these envelopes provides a unilateral improvement when computing relaxations using a reduced-space optimization approach.

Chapter 5

Improved Relaxation of Composite Bilinear Form

Deterministic nonconvex optimization solvers generate convex relaxations of nonconvex functions by making use of underlying factorable representations. One approach introduces auxiliary variables assigned to each factor that lifts the problem into a higher-dimensional decision space. In contrast, a generalized McCormick relaxation approach offers the significant advantage of constructing relaxations in the lower dimensionality space of the original problem without introducing auxiliary variables, often referred to as a "reduced-space" approach. Recent contributions illustrated how additional nontrivial inequality constraints may be used in factorable programming to tighten relaxations of the ubiquitous bilinear term. In this chapter, we exploit an analogous representation of McCormick relaxations and factorable programming to formulate tighter relaxations in the original decision space using *a priori* information. We develop the underlying theory to generate necessarily tighter reduced-space McCormick relaxations when *a priori* convex/concave relaxations of intermediate bilinear terms are known. We then show how these rules can be applied in a McCormick relaxation scheme via three different approaches: the

use of a McCormick relaxations coupled to affine arithmetic, the propagation of affine relaxations implied by subgradients, and an enumerative approach which directly uses relaxations of each factor. In the first example, we illustrate how these results may be applied in advanced manufacturing context to optimize supply chain quality metrics. New approaches are then demonstrated using the EAGO.jl optimizer on several of examples taken from the literature. We find that each method considered leads to an improvement to CPU time used during optimizing via the branch-and-bound algorithm.

5.1 Introduction

Deterministic global optimization is required by many routine process systems engineering (PSE) tasks due to the nonconvexity of underlying process models. Moreover, PSE applications routinely require the strict satisfaction of safety-critical and quality-critical constraints while accounting for complex physical processes and equipment design specifications. As a consequence, a certificate of global optimality is highly preferred or potentially required when such design decisions are made. This is particularly true when the consequences of a system failure are catastrophic as is the case of nuclear reactor design wherein the cooling apparatus should be sized to accommodate a wide process window by accounting for a true worst-case scenario. In this case, knowledge of worst-case performance obtained by a globally optimal solution provides an assurance that accounting for a particularly concerning case does not.

There are two main approaches to solving global optimization problems deterministically. The approach which is predominant in state-of-the-art commercial solvers is that of the auxilliary variable method, which exploits a factorable representation of the underlying problem by subsequently lifting the problem into a higher-dimensional space [136]. This higher-dimensional

representation simplifies the construction of convex/concave relaxations required to form subproblems, facilitates potentially tighter relaxations of mixed-integer expressions [185], and simplifies a number of important heuristics. For some problems, the introduction of these additional variables may be detrimental as the aforementioned advantages are counterbalanced by the need to branch on more variables and the well-known the ”curse of dimensionality” which is intrinsic to solving this NP-hard class of problems. An alternative to this is relaxations may be computed in the original functional space.

The eponymous McCormick relaxation of the bilinear function was first introduced in [177]. This relaxation bounds the bilinear term using a series of affine inequalities; an approach used by many commercially available optimizers, such as ANTIGONE [185] and BARON [254], and the nonconvex solver options of CPLEX [66] and Gurobi [117]. In the past decade, a significant effort has been made to further generalize this approach to arbitrary nonlinear functions. An operator-overloading scheme was detailed by Mitsos et al. [191] for constructing McCormick-based relaxations of functions described by a class of direct algorithms (i.e., algorithms with the number of steps/iterations known *a priori*). Variations on this manner of constructing relaxations in the original problem space through the application of composition rules have been termed *McCormick relaxations*; a convention we adopt herein to maintain consistency with the existing body of literature.

The use of McCormick relaxations [191] potentially offer a significant advantage by allowing for relaxations to be constructed in the reduced-dimension space of the original problem. Recent developments have dramatically broadened the scope and performance of this approach. Scott et al. [271] developed a generalized McCormick relaxation theory for constructing convex and concave composite relaxations using arbitrary convex and concave functions. Tighter composition rules for multiplication and maximum operators were presented in [207, 310].

Methods of generating relaxations of implicit functions were developed by Stuber et al. [300]. Wechsung et al. [324] developed a method of propagating McCormick relaxations backwards on a directed acyclic graph (DAG) representation of a problem. A method for tightening interval bounds was described in [206]. Alternative differentiable relaxations were developed and introduced in [151, 152].

Moreover, the theoretical underpinnings of McCormick relaxation performance have been recently explored. These works have illustrated that under mild assumptions, these McCormick relaxations exhibit quadratic point-wise convergence [44, 203, 205]; which may mitigate the clustering issues common to branch-and-bound algorithms[146]. Each of these aforementioned advances has been demonstrated to led to improved performance of global optimizers for specialized classes of simulation-inspired problems.

The benefits of using these reduced-space McCormick relaxation method have been found to span a number of application areas. These include the deterministic global optimization of process flowsheets [48, 49, 50], nonconvex optimization problems with embedded surrogate models (such as artificial neural networks and Gaussian process models) [264, 265, 266, 267, 330], dynamic optimization [270, 328], and reachability analysis [256]. Recently, McCormick relaxations have been implemented in two open-source global optimizers: the EAGO [327] toolkit in Julia [36], and the MAiNGO [46] software written in C++. In each implementation, the overloading approach taken to construct relaxations leads to the classic *dependency problem* inherent to set-valued arithmetics; wherein, the progressive application of bounding rules leads to expansive departures from convex/concave envelopes of complicated expressions.

To ameliorate the dependency problem, several efforts have been made to expand the typical library of intrinsic functions to include envelopes for common functional forms. These efforts include the development of relaxations of componentwise-convex functions by [208], the

construction of novel relaxations of cost and thermodynamic functions in [209], relaxations of activation functions appearing on artificial neural networks [330] and Gaussian processes [267], as well as special relaxations for the logarithmic mean temperature difference (LMTD) and its reciprocal [187, 204].

One expression of particular interest is that of the bilinear term. This term has been examined extensively within the deterministic nonconvex optimization community. Specialized approaches to treating these problem classes have led to numerous optimizers that initially focused on quadratic (and polynomial) problem formulations and were often subsequently extended to a number of preeminent optimizers including BARON [305], ANTIGONE [185], Gurobi [117], GLOMIQO [184], MOSEK [195], and ALPINE [198, 199].

Within the McCormick relaxations literature, the treatment of the composite bilinear term ($w(\mathbf{z}) = f(\mathbf{z})g(\mathbf{z})$ on $\mathbf{z} \in Z$) has been limited to three key theoretical contributions. The first is provided in the work of [191] that details composite relaxations derived from McCormick's original inequalities [177]. The second contribution lies in the analysis of multivariate composite relaxations [207, 310] that yield potentially tighter relaxations of the bilinear term under mild assumptions. Lastly, a differentiable relaxation of the bilinear term was detailed in [151, 152].

In the large context of full-space factorable programming, numerous approaches exist to address the bilinear relaxation, which do not yet directly have an analog in reduced-space factorable programming. One such notable work is that of He and Tawarmalani [127], which details how bilinear relaxations of composite factors can be improved when *a priori* over/underestimators (as well as associated bounds of said over/underestimators) of the bilinear factors, are available. A lifting approach is necessary when this strategy is applied withing factorable programming, which the authors resolve using a fast combinatorial algorithm to solve a simpler *separation problem*.

In this chapter, we build upon the recent work of He and Tawarmalani [127], detailing how reduced-space McCormick relaxations may be improved when *a priori* knowledge of intermediate convex/concave relaxations exists. We do this by generalizing the results of [127] for factorable programming to composite relaxations. Our approach does not rely on knowledge of *a priori* over/underestimators, rather, only relaxations of these functions (which may be simpler to obtain for intermediate terms) are required for computation. We subsequently discuss three algorithms used to refine convex/concave relaxations of functions for a broad class of nonlinear functions in the original problem space. In Section 5.2, we develop composition rules for generating convex/concave relaxations of intermediate bilinear terms when *a priori* relaxations are known along with associated subgradients. Subsequently, in Section 5.3, we detail three algorithms that employ this novel theoretical contribution to generate tight relaxations in the reduced-dimension problem space. We then explore a case study in Section 5.5 which demonstrate how improved composite bilinear relaxations may be applied for quality chain design. In Section 5.4, we provide numerical examples detailing the utility of each algorithm developed herein. Lastly, we conclude in Section 5.7 by highlighting potential areas for future research.

5.2 Tight Composite Relaxations of Bilinear Terms

We now describe two major theoretical contributions. First, we develop Theorem 5.2.1 as an extension of Theorem 1 and Theorem 5 from [127], to compute convex and concave relaxations of the bilinear term using convex and concave relaxations of its arguments and *a priori* convex underestimators. This new result differs from the preliminary work of [127] in that the introduction of auxiliary variables for intermediate bilinear terms, into the optimization formulation, is not required. Secondly, a corresponding approach that makes use of *a priori*

concave relaxations is detailed in Theorem 5.2.2. We combine these results in Theorem 5.2.3 to obtain tight relaxations of bilinear terms exploiting both *a priori* convex and concave relaxations, simultaneously. Finally, we develop subgradients of these relaxations in Theorem 5.2.6.

Theorem 5.2.1. Define $x_1 : Z \subset \mathbb{R}^n \to X_1 \subset \mathbb{R}$ and $x_2 : Z \subset \mathbb{R}^n \to X_2 \subset \mathbb{R}$ with corresponding convex/convex relaxations $x_1^{cv}, x_1^{cc}, x_2^{cv}, x_2^{cc} : Z \to \mathbb{R}$ on Z. Let $u_1, u_2 : Z \subset \mathbb{R}^n \to \mathbb{R}$ be underestimators of x_1, x_2 on Z, respectively, with associated $(a_1, a_2) \in X_1 \times X_2 \in \mathbb{IR}^2$ such that $x_1^L \leq u_1(\cdot) \leq \min\{x_1(\cdot), a_1\}$ and $x_2^L \leq u_2(\cdot) \leq \min\{x_2(\cdot), a_2\}$. Further, suppose that convex relaxations of u_1 and u_2 on Z are available. Let the following intermediate factors be defined as:

$$\begin{aligned}
\alpha_1(\cdot) &= \min\{a_2 x_1^{cv}(\cdot), a_2 x_1^{cc}(\cdot)\}, & \beta_1(\cdot) &= \max\{x_1^U x_2^{cv}(\cdot), x_1^U x_2^{cc}(\cdot)\},\\
\alpha_2(\cdot) &= \min\{a_1 x_2^{cv}(\cdot), a_1 x_2^{cc}(\cdot)\}, & \beta_2(\cdot) &= \max\{a_2 x_2^{cv}(\cdot), a_2 x_2^{cc}(\cdot)\},\\
\alpha_3(\cdot) &= \min\{x_2^L x_1^{cv}(\cdot), x_2^L x_1^{cc}(\cdot)\}, & \beta_3(\cdot) &= \max\{a_1 x_2^{cv}(\cdot), a_1 x_2^{cc}(\cdot)\},\\
\alpha_4(\cdot) &= \min\{x_1^L x_2^{cv}(\cdot), x_1^L x_2^{cc}(\cdot)\}, & \beta_4(\cdot) &= \max\{x_2^U x_2^{cv}(\cdot), x_2^U x_2^{cc}(\cdot)\},\\
\rho_1 &= a_1 a_2 - a_1 x_2^U - a_2 x_1^U.
\end{aligned}$$

Then, the following expressions:

$$w_1^{cv}(\cdot) = (x_2^U - a_2)u_1^{cv}(\cdot) + (x_1^U - a_1)u_2^{cv}(\cdot) + \alpha_1(\cdot) + \alpha_2(\cdot) + \rho_1 \tag{5.2.1}$$

$$w_2^{cv}(\cdot) = (x_2^U - x_2^L)u_1^{cv}(\cdot) + \alpha_2(\cdot) + \alpha_3(\cdot) - a_1 x_2^U \tag{5.2.2}$$

$$w_3^{cv}(\cdot) = (x_1^U - x_1^L)u_2^{cv}(\cdot) + \alpha_1(\cdot) + \alpha_4(\cdot) - a_2 x_1^U \tag{5.2.3}$$

$$w_4^{cv}(\cdot) = (a_2 - x_2^L)u_1^{cv}(\cdot) + (a_1 - x_1^L)u_2^{cv}(\cdot) + \alpha_3(\cdot) + \alpha_4(\cdot) - a_1 a_2 \tag{5.2.4}$$

are convex relaxations of $w(\cdot) = x_1(\cdot)x_2(\cdot)$ on Z. Moreover, the following expressions:

$$w_1^{cc}(\cdot) = (x_2^L - a_2)u_1^{cv}(\cdot) + (a_1 - x_1^U)u_2^{cv}(\cdot) + \beta_1(\cdot) + \beta_2(\cdot) - a_1x_2^L \tag{5.2.5}$$

$$w_2^{cc}(\cdot) = (x_2^L - x_2^U)u_1^{cv}(\cdot) + \beta_3(\cdot) + \beta_4(\cdot) - a_1x_2^L \tag{5.2.6}$$

$$w_3^{cc}(\cdot) = (x_1^L - x_1^U)u_2^{cv}(\cdot) + \beta_1(\cdot) + \beta_2(\cdot) - a_2x_1^L \tag{5.2.7}$$

$$w_4^{cc}(\cdot) = (a_2 - x_2^U)u_1^{cv}(\cdot) + (x_1^L - a_1)u_2^{cv}(\cdot) + \beta_3(\cdot) + \beta_4(\cdot) - a_2x_1^L, \tag{5.2.8}$$

are concave relaxations of $w(\cdot) = x_1(\cdot)x_2(\cdot)$ on Z. Lastly, the expressions

$$w_{\times,1}^{cv}(\cdot) = \max\left\{w_1^{cv}(\cdot), w_2^{cv}(\cdot), w_3^{cv}(\cdot), w_4^{cv}(\cdot)\right\} \tag{5.2.9}$$

$$w_{\times,1}^{cc}(\cdot) = \min\left\{w_1^{cc}(\cdot), w_2^{cc}(\cdot), w_3^{cc}(\cdot), w_4^{cc}(\cdot)\right\} \tag{5.2.10}$$

are convex and concave relaxations of $w(\cdot) = x_1(\cdot)x_2(\cdot)$ on Z, respectively.

Proof. He and Tawarmalani [127] define nontrival underestimators and overestimators of $w^*(\xi_1, \xi_2) = \xi_1\xi_2$ on $X_1 \times X_2$ derived from *a priori* underestimators and overestimators by the planes $e_i(\xi_1, \xi_2) = 0$ and $r_i(\xi_1, \xi_2) = 0$ for $i = 2, \ldots 5$, respectively. As we have $x_i : Z \to X_i$ for $i \in 1, 2$, we may write $\mathcal{E}_i(\cdot) = e_i(x_1(\cdot), x_2(\cdot)) = 0$ and $\mathcal{R}_i(\cdot) = r_i(x_1(\cdot), x_2(\cdot)) = 0$ for $i = 2, \ldots 5$,

respectively, which are under/overestimators of $w(\cdot)$ on Z. Namely,

$$\mathcal{E}_2(\cdot) = (x_2^U - a_2)u_1(\cdot) + (x_1^U - a_1)u_2(\cdot) + a_2x_1(\cdot) + a_1x_2(\cdot) + \rho_1 \tag{5.2.11}$$

$$\mathcal{E}_3(\cdot) = (x_2^U - x_2^L)u_1(\cdot) + x_2^Lx_1(\cdot) + a_1x_2(\cdot) - a_1x_2^U \tag{5.2.12}$$

$$\mathcal{E}_4(\cdot) = (x_1^U - x_1^L)u_2(\cdot) + a_2x_1(\cdot) + x_1^Lx_2(\cdot) - a_2x_1^U \tag{5.2.13}$$

$$\mathcal{E}_5(\cdot) = (a_2 - x_2^L)u_1(\cdot) + (a_1 - x_1^L)u_2(\cdot) + x_2^Lx_1(\cdot) + x_1^Lx_2(\cdot) - a_1a_2 \tag{5.2.14}$$

$$\mathcal{R}_2(\cdot) = (x_2^L - a_2)u_1(\cdot) + (a_1 - x_1^U)u_2(\cdot) + a_2x_1(\cdot) + x_1^Ux_2(\cdot) - a_1x_2^L \tag{5.2.15}$$

$$\mathcal{R}_3(\cdot) = (x_2^L - x_2^U)u_1(\cdot) + a_1x_2(\cdot) + x_2^Ux_1(\cdot) - a_1x_2^L \tag{5.2.16}$$

$$\mathcal{R}_4(\cdot) = (x_1^L - x_1^U)u_2(\cdot) + a_2x_1(\cdot) + x_1^Ux_2(\cdot) - a_2x_1^L \tag{5.2.17}$$

$$\mathcal{R}_5(\cdot) = (a_2 - x_2^U)u_1(\cdot) + (x_1^L - a_1)u_2(\cdot) + x_2^Ux_1(\cdot) + a_1x_2(\cdot) - a_2x_1^L \tag{5.2.18}$$

First, we note that the terms $(x_i^U - a_i)$, $(x_i^U - x_i^L)$, and $(a_i - x_i^L)$ are positive for $i \in \{1, 2\}$. As such, convex relaxations of the $\alpha u_i(\cdot)$ terms in (5.2.11)-(5.2.14) on Z are given by $\alpha u_i^{cv}(\cdot)$, for $i \in \{1, 2\}$. Next we note that $(a_i - x_i^U)$, $(x_i^L - x_i^U)$, and $(x_i^L - a_i)$ are negative by construction for $i \in \{1, 2\}$. As such, concave relaxations of the $\alpha u_i(\cdot)$ terms in (5.2.15)-(5.2.18) on Z are given by $\alpha u_i^{cv}(\cdot)$, for $i \in \{1, 2\}$. The remaining coefficients of $x_1(\cdot)$ and $x_2(\cdot)$ may be either positive or negative, and as such, a convex relaxation of $\alpha x_i(\cdot)$ on Z is given by $\min\left\{\alpha x_i^{cv}(\cdot), \alpha x_i^{cc}(\cdot)\right\}$, whereas a concave relaxation of $\alpha x_i(\cdot)$ on Z is given by $\max\left\{\alpha x_i^{cv}(\cdot), \alpha x_i^{cc}(\cdot)\right\}$ for $i \in \{1, 2\}$. The sum of convex functions is convex, and the sum of concave functions is concave. The pointwise maximum of convex underestimators is convex while the pointwise minimum of concave overestimators is concave. Thus, the expressions (5.2.1)-(5.2.10) hold. □

The above relaxations are derived from Theorems 1 and 5 presented in [127]. This requires knowledge of valid underestimating functions u_1 and u_2. When propagating relaxations through a composite function, it is quite common to have *a priori* knowledge of valid overestimating

functions as well. The following Theorem 5.2.2 is a new result that adapts the results of Theorem 5.2.1 to improve relaxations using valid overestimating functions known *a priori*.

Theorem 5.2.2. Define $x_1 : Z \subset \mathbb{R}^n \to X_1 \subset \mathbb{R}$ and $x_2 : Z \subset \mathbb{R}^n \to X_2 \subset \mathbb{R}$ with corresponding convex/convex relaxations $x_1^{cv}, x_1^{cc}, x_2^{cv}, x_2^{cc} : Z \to \mathbb{R}$ on Z. Let $o_1, o_2 : Z \to \mathbb{R}$ be overestimators of x_1, x_2 on Z, respectively, with associated $(b_1, b_2) \in X_1 \times X_2 \in \mathbb{IR}^2$ such that $x_1^U \geq o_1(\cdot) \geq \max\{x_1(\cdot), b_1\}$ and $x_2^U \geq o_2(\cdot) \geq \max\{x_2(\cdot), b_2\}$. Further, suppose that concave relaxations of o_1 and o_2 on Z are available. Let the following intermediate factors be defined as

$$\begin{aligned}
\delta_1(\cdot) &= \min\{b_2 x_1^{cv}(\cdot), b_2 x_1^{cc}(\cdot)\}, & \gamma_1(\cdot) &= \max\{b_2 x_1^{cv}(\cdot), b_2 x_1^{cc}(\cdot)\},\\
\delta_2(\cdot) &= \min\{b_1 x_2^{cv}(\cdot), b_1 x_2^{cc}(\cdot)\}, & \gamma_2(\cdot) &= \max\{x_1^L x_2^{cv}(\cdot), x_1^L x_2^{cc}(\cdot)\},\\
\delta_3(\cdot) &= \min\{x_2^U x_1^{cv}(\cdot), x_2^U x_1^{cc}(\cdot)\}, & \gamma_3(\cdot) &= \max\{b_1 x_2^{cv}(\cdot), b_1 x_2^{cc}(\cdot)\},\\
\delta_4(\cdot) &= \min\{x_1^U x_2^{cv}(\cdot), x_1^U x_2^{cc}(\cdot)\}, & \gamma_4(\cdot) &= \max\{x_2^L x_1^{cv}(\cdot), x_2^L x_1^{cc}(\cdot)\},\\
\rho_2 &= b_1 b_2 - b_1 x_2^L - b_2 x_1^L.
\end{aligned}$$

Then, the following expressions:

$$w_5^{cv}(\cdot) = (x_2^L - b_2) o_1^{cc}(\cdot) + (x_1^L - b_1) o_2^{cc}(\cdot) + \delta_1(\cdot) + \delta_2(\cdot) + \rho_2 \tag{5.2.19}$$

$$w_6^{cv}(\cdot) = (x_2^L - x_2^U) o_1^{cc}(\cdot) + \delta_2(\cdot) + \delta_3(\cdot) - b_1 x_2^L \tag{5.2.20}$$

$$w_7^{cv}(\cdot) = (x_1^L - x_1^U) o_2^{cc}(\cdot) + \delta_1(\cdot) + \delta_4(\cdot) - b_2 x_1^L \tag{5.2.21}$$

$$w_8^{cv}(\cdot) = (b_2 - x_2^U) o_1^{cc}(\cdot) + (b_1 - x_1^U) o_2^{cc}(\cdot) + \delta_3(\cdot) + \delta_4(\cdot) - b_1 b_2 \tag{5.2.22}$$

are convex relaxations of $w(\cdot) = x_1(\cdot)x_2(\cdot)$ on Z. Moreover, the following expressions:

$$w_5^{cc}(\cdot) = (x_2^U - b_2)o_1^{cc}(\cdot) + (b_1 - x_1^L)o_2^{cc}(\cdot) + \gamma_1(\cdot) + \gamma_2(\cdot) - b_1 x_2^U \tag{5.2.23}$$

$$w_6^{cc}(\cdot) = (x_2^U - x_2^L)o_1^{cc}(\cdot) + \gamma_3(\cdot) + \gamma_4(\cdot) - x_2^U b_1 \tag{5.2.24}$$

$$w_7^{cc}(\cdot) = (x_1^U - x_1^L)o_2^{cc}(\cdot) + \gamma_1(\cdot) + \gamma_2(\cdot) - x_1^U b_2 \tag{5.2.25}$$

$$w_8^{cc}(\cdot) = (b_2 - x_2^L)o_1^{cc}(\cdot) + (x_1^U - b_1)o_2^{cc}(\cdot) + \gamma_3(\cdot) + \gamma_4(\cdot) - x_1^U b_2 \tag{5.2.26}$$

are concave relaxations of $w(\cdot) = x_1(\cdot)x_2(\cdot)$ on Z. Lastly, the expressions

$$w_{\times,2}^{cv}(\cdot) = \max\left\{w_5^{cv}(\cdot), w_6^{cv}(\cdot), w_7^{cv}(\cdot), w_8^{cv}(\cdot)\right\}$$
$$w_{\times,2}^{cc}(\cdot) = \min\left\{w_5^{cc}(\cdot), w_6^{cc}(\cdot), w_7^{cc}(\cdot), w_8^{cc}(\cdot)\right\}$$

are convex and concave relaxations of $w(\cdot) = x_1(\cdot)x_2(\cdot)$ on Z, respectively.

Proof. First, we note that $x_1(\cdot)x_2(\cdot) = y_1(\cdot)y_2(\cdot)$, with $y_1(\cdot) = -x_1(\cdot)$ and $y_2(\cdot) = -x_2(\cdot)$. Define $(a_1, a_2) \in Y_1 \times Y_2$, where $Y_i = -X_i$ and $a_i = -b_i$. Let $u_1, u_2 : Z \to \mathbb{R}$ be underestimators of y_1, y_2 on Z, respectively, defined as $u_i(\cdot) = -o_i(\cdot)$. Then, we see that $y_1^L \leq u_1(\cdot) \leq \min\{y_1(\cdot), a_1\}$, $y_1(\cdot) \leq y_1^U$ and $y_2^L \leq u_2(\cdot) \leq \min\{y_2(\cdot), a_2\}$, $y_2(\cdot) \leq y_2^U$. Similarly, we have $y_i^L \leq u_i(\cdot) \leq \min\{y_i(\cdot), a_i\}$ and $-y_i^L \geq -u_i(\cdot) \geq -\min\{y_i(\cdot), a_i\}$, and therefore
$x_i^U \geq -u_i(\cdot) \geq -\min\{-x_i(\cdot), a_i\} = \max\{x_i(\cdot), -a_i\} = \max\{x_i(\cdot), b_i\}$. It remains to show that the facets defined by Equations (5.2.19) - (5.2.26) are valid. Again inspecting the underestimators

and overestimators of Theorems 1 and 5 of [127], we have

$$\mathcal{E}_2(\cdot) = (y_2^U - a_2)u_1(\cdot) + (y_1^U - a_1)u_2(\cdot) + a_2y_1(\cdot) + a_1y_2(\cdot) + a_1a_2 - a_1y_2^U - y_1^Ua_2$$

$$\mathcal{E}_3(\cdot) = (y_2^U - y_2^L)u_1(\cdot) + y_2^Ly_1(\cdot) + a_1y_2(\cdot) - a_1y_2^U$$

$$\mathcal{E}_4(\cdot) = (y_1^U - y_1^L)u_2(\cdot) + a_2y_1(\cdot) + y_1^Ly_2(\cdot) - a_2y_1^U$$

$$\mathcal{E}_5(\cdot) = (a_2 - y_2^L)u_1(\cdot) + (a_1 - y_1^L)u_2(\cdot) + y_2^Ly_1(\cdot) + y_1^Ly_2(\cdot) - a_1a_2$$

$$\mathcal{R}_2(\cdot) = (y_2^L - a_2)u_1(\cdot) + (a_1 - y_1^U)u_2(\cdot) + a_2y_1(\cdot) + y_1^Uy_2(\cdot) - a_1y_2^L$$

$$\mathcal{R}_3(\cdot) = (y_2^L - y_2^U)u_1(\cdot) + a_1y_2(\cdot) + y_2^Uy_1(\cdot) - a_1y_2^L$$

$$\mathcal{R}_4(\cdot) = (y_1^L - y_1^U)u_2(\cdot) + a_2y_1(\cdot) + y_1^Uy_2(\cdot) - a_2y_1^L$$

$$\mathcal{R}_5(\cdot) = (a_2 - y_2^U)u_1(\cdot) + (y_1^L - a_1)u_2(\cdot) + y_2^Uy_1(\cdot) + a_1y_2(\cdot) - a_2y_1^L.$$

Substituting in b_i for a_i, $x_i(\cdot)$ for $y_i(\cdot)$, and $o_i(\cdot)$ for $u_i(\cdot)$, we get

$$\mathcal{E}_2(\cdot) = (x_2^L - b_2)o_1(\cdot) + (x_1^L - b_1)o_2(\cdot) + b_2x_1(\cdot) + b_1x_2(\cdot) + b_1b_2 - b_1x_2^L - b_2x_1^L$$

$$\mathcal{E}_3(\cdot) = (x_2^L - x_2^U)o_1(\cdot) + x_2^Ux_1(\cdot) + b_1x_2(\cdot) - b_1x_2^L$$

$$\mathcal{E}_4(\cdot) = (x_1^L - x_1^U)o_2(\cdot) + b_2x_1(\cdot) + x_1^Ux_2(\cdot) - b_2x_1^L$$

$$\mathcal{E}_5(\cdot) = (b_2 - x_2^U)o_1(\cdot) + (b_1 - x_1^U)o_2(\cdot) + x_2^Ux_1(\cdot) + x_1^Ux_2(\cdot) - b_1b_2$$

$$\mathcal{R}_2(\cdot) = (x_2^U - b_2)o_1(\cdot) + (b_1 - x_1^L)o_2(\cdot) + b_2x_1(\cdot) + x_1^Lx_2(\cdot) - b_1x_2^U$$

$$\mathcal{R}_3(\cdot) = (x_2^U - x_2^L)o_1(\cdot) + b_1x_2(\cdot) + x_2^Lx_1(\cdot) - b_1x_2^U$$

$$\mathcal{R}_4(\cdot) = (x_1^U - x_1^L)o_2(\cdot) + b_2x_1(\cdot) + x_1^Lx_2(\cdot) - b_2x_1^U$$

$$\mathcal{R}_5(\cdot) = (b_2 - x_2^L)o_1(\cdot) + (x_1^U - b_1)o_2(\cdot) + x_2^Lx_1(\cdot) + b_1x_2(\cdot) - b_2x_1^U.$$

We then construct convex and concave relaxations of these expressions in a manner similar to that for Theorem 5.2.1. □

Theorem 5.2.3. Define $x_1 : Z \subset \mathbb{R}^n \to X_1 \subset \mathbb{R}$ and $x_2 : Z \subset \mathbb{R}^n \to X_2 \subset \mathbb{R}$ with corresponding convex/convex relaxations $x_1^{cv}, x_1^{cc}, x_2^{cv}, x_2^{cc} : Z \to \mathbb{R}$ on Z. Let $u_1, u_2 : Z \subset \mathbb{R}^n \to \mathbb{R}$ be underestimators and $o_1, o_2 : Z \to \mathbb{R}$ be overestimators of x_1, x_2 on Z, respectively. Let $(a_1, a_2), (b_1, b_2) \in X_1 \times X_2 \in \mathbb{IR}^2$ such that $x_1^L \leq u_1(\cdot) \leq \min\{x_1(\cdot), a_1\}$, $\max\{x_1(\cdot), b_1\} \leq o_1(\cdot) \leq x_1^U$, $x_2^L \leq u_2(\cdot) \leq \min\{x_2(\cdot), a_2\}$, and $\max\{x_2(\cdot), b_2\} \leq o_2(\cdot) \leq x_2^U$. Then, convex and concave relaxations of $w(\cdot) = x_1(\cdot)x_2(\cdot)$ are, respectively, given by

$$w^{cv}(\cdot) = \max\left\{w_{\times,0}^{cv}(\cdot), w_{\times,1}^{cv}(\cdot), w_{\times,2}^{cv}(\cdot), w^L\right\},$$
$$w^{cc}(\cdot) = \min\left\{w_{\times,0}^{cc}(\cdot), w_{\times,1}^{cc}(\cdot), w_{\times,2}^{cc}(\cdot), w^U\right\}.$$

Proof. This result follows directly from the application of Proposition 2.3.7, Theorem 5.2.1, and Theorem 5.2.2, and basic convexity/concavity properties. □

As discussed in [127], the relaxations from Theorem 5.2.1 and Theorem 5.2.2 reduce to the form given by Proposition 2.3.7 if $a_i, b_i \in \left\{x_i^L, x_i^U\right\}$ for $i \in \{1, 2\}$.

Proposition 5.2.4. Define $x_1 : Z \subset \mathbb{R}^n \to X_1 \subset \mathbb{R}$ and $x_2 : Z \subset \mathbb{R}^n \to X_2 \subset \mathbb{R}$ with corresponding convex/convex relaxations $x_1^{cv}, x_1^{cc}, x_2^{cv}, x_2^{cc} : Z \to \mathbb{R}$ on Z. Let $u_1, u_2 : Z \subset \mathbb{R}^n \to \mathbb{R}$ be underestimators of x_1 and x_2 on Z, respectively. Let $(a_1, a_2) \in X_1 \times X_2 \in \mathbb{IR}^2$ such that $x_1^L \leq u_1^{cv}(\cdot) \leq u_1(\cdot) \leq \min\{x_1(\cdot), a_1\}$, $x_2^L \leq u_2^{cv}(\cdot) \leq u_2(\cdot) \leq \min\{x_2(\cdot), a_2\}$. Further, suppose that $a_i \in \left\{x_i^L, x_i^U\right\}$ for $i \in \{1, 2\}$ then the convex relaxations presented in Theorems 5.2.1 and 5.2.2 reduce to $w_{\times,0}^{cv}(\cdot)$.

Proof. We proceed by enumeration of the possible cases to illustrate this result. For brevity, we observe that the relaxations obtained from $w(\cdot) = x_1(\cdot)x_2(\cdot)$ and $w(\cdot) = x_2(\cdot)x_1(\cdot)$ are equivalent. Moreover, $w(\cdot) = x_1(\cdot)x_2(\cdot)$ can be written as $w(\cdot) = (-x_1(\cdot))(-x_2(\cdot))$ and in doing so negates and

swaps the positions of the upper and lower interval bounds. Then without loss of generality, we restrict our consideration to two cases: $(a_1, a_2) = (x_1^L, x_2^L)$ and $(a_1, a_2) = (x_1^L, x_2^U)$.

For $(a_1, a_2) = (x_1^L, x_2^L)$, we have $u_i^{cv} = u_i = x_i^L$ which is identical to the $(a_1, a_2) = (x_1^L, x_2^L)$ case of He and Tawarmalani [127] (Theorem 1). For $(a_1, a_2) = (x_1^L, x_2^U)$, we have

$$\mathcal{E}_2(\cdot) = (x_1^U - x_1^L)u_2(\cdot) + x_2^U x_1(\cdot) + x_1^L x_2(\cdot) - x_1^U x_2^U$$
$$\mathcal{E}_3(\cdot) = x_2^L x_1(\cdot) + x_1^L x_2(\cdot) - x_2^L x_1^L$$

with $\mathcal{E}_4(\cdot) = \mathcal{E}_2(\cdot)$ and $\mathcal{E}_5(\cdot) = \mathcal{E}_3(\cdot) = \mathcal{E}_1(\cdot)$. Moreover, $(x_1^U - x_1^L)u_2(\cdot) \leq (x_1^U - x_1^L)x_2(\cdot)$ and $\mathcal{E}_2(\cdot) \leq x_1^U x_2(\cdot) + x_2^U x_1(\cdot) - x_1^U x_2^U = \mathcal{E}_6(\cdot)$. □

Clearly, nontrivial lower and upper bounds must be available if relaxations of this form are expected to improve on the McCormick composition approach. In the next section, we propose three computational approaches to achieving this. In the remainder of this section, we detail associated rules for propagating valid subgradients of the convex/concave relaxations that are necessary when forming affine relaxations or as an input to nonsmooth convex NLP solvers.

Definition 5.2.5 ($\omega, \boldsymbol{\Omega}$)**.** Let $a \in \mathbb{R}$, $\boldsymbol{\sigma}_1, \boldsymbol{\sigma}_2 \in \mathbb{R}^n$. The functions $\omega, \boldsymbol{\Omega} : \mathbb{R} \times \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}^n$ are defined as

$$\omega(a, \boldsymbol{\sigma}_1, \boldsymbol{\sigma}_2) \equiv \begin{cases} a\boldsymbol{\sigma}_1 & a \geq 0, \\ a\boldsymbol{\sigma}_2 & \text{otherwise,} \end{cases}$$
$$\boldsymbol{\Omega}(a, \boldsymbol{\sigma}_1, \boldsymbol{\sigma}_2) \equiv \begin{cases} a\boldsymbol{\sigma}_1 & a \leq 0, \\ a\boldsymbol{\sigma}_2 & \text{otherwise.} \end{cases}$$

Theorem 5.2.6. Let $Z \subset \mathbb{R}^n$ be a nonempty convex set and $w, x_1, x_2 : Z \to \mathbb{R}$ such that $w(\cdot) = x_1(\cdot)x_2(\cdot)$ with corresponding convex/convex relaxations $x_1^{cv}, x_1^{cc}, x_2^{cv}, x_2^{cc} : Z \to \mathbb{R}$ on Z. Let $u_1, u_2 : Z \subset \mathbb{R}^n \to \mathbb{R}$ be underestimators and $o_1, o_2 : Z \to \mathbb{R}$ be overestimators of x_1, x_2 on Z, respectively. Let $(a_1, a_2), (b_1, b_2) \in X_1 \times X_2 \in \mathbb{IR}^2$ such that $x_1^L \leq u_1(\cdot) \leq \min\{x_1(\cdot), a_1\}$, $\max\{x_1(\cdot), b_1\} \leq o_1(\cdot) \leq x_1^U$, $x_2^L \leq u_2(\cdot) \leq \min\{x_2(\cdot), a_2\}$, and $\max\{x_2(\cdot), b_2\} \leq o_2(\cdot) \leq x_2^U$. Let $w_i^{cv}(\cdot)$ and $w_i^{cc}(\cdot)$ on Z be defined as in Theorems 5.2.1 and 5.2.2. Then, subgradients $\mathbf{s}_{w_i}^{cv}(\bar{\mathbf{z}}), \mathbf{s}_{w_i}^{cc}(\bar{\mathbf{z}})$ of w_i^{cv} and w_i^{cc} on Z, evaluated at $\bar{\mathbf{z}} \in Z$, for $i = 1, \ldots, 8$, are given by

$$
\begin{aligned}
\mathbf{s}_{w_1}^{cv}(\bar{\mathbf{z}}) &= (x_2^U - a_2)\mathbf{s}_{u_1}^{cv}(\bar{\mathbf{z}}) + (x_1^U - a_1)\mathbf{s}_{u_2}^{cv}(\bar{\mathbf{z}}) + \boldsymbol{\omega}\left(a_2, \mathbf{s}_{x_1}^{cv}(\bar{\mathbf{z}}), \mathbf{s}_{x_1}^{cc}(\bar{\mathbf{z}})\right) \\
&\quad + \boldsymbol{\omega}\left(a_1, \mathbf{s}_{x_2}^{cv}(\bar{\mathbf{z}}), \mathbf{s}_{x_2}^{cc}(\bar{\mathbf{z}})\right), \\
\mathbf{s}_{w_2}^{cv}(\bar{\mathbf{z}}) &= (x_2^U - x_2^L)\mathbf{s}_{u_1}^{cv}(\bar{\mathbf{z}}) + \boldsymbol{\omega}\left(x_2^L, \mathbf{s}_{x_1}^{cv}(\bar{\mathbf{z}}), \mathbf{s}_{x_1}^{cc}(\bar{\mathbf{z}})\right) + \boldsymbol{\omega}\left(a_1, \mathbf{s}_{x_2}^{cv}(\bar{\mathbf{z}}), \mathbf{s}_{x_2}^{cc}(\bar{\mathbf{z}})\right), \\
\mathbf{s}_{w_3}^{cv}(\bar{\mathbf{z}}) &= (x_1^U - x_1^L)\mathbf{s}_{u_2}^{cv}(\bar{\mathbf{z}}) + \boldsymbol{\omega}\left(a_2, \mathbf{s}_{x_1}^{cv}(\bar{\mathbf{z}}), \mathbf{s}_{x_1}^{cv}(\bar{\mathbf{z}})\right) + \boldsymbol{\omega}\left(x_1^L, \mathbf{s}_{x_2}^{cv}(\bar{\mathbf{z}}), \mathbf{s}_{x_2}^{cv}(\bar{\mathbf{z}})\right), \\
\mathbf{s}_{w_4}^{cv}(\bar{\mathbf{z}}) &= (a_2 - x_2^L)\mathbf{s}_{u_1}^{cv}(\bar{\mathbf{z}}) + (a_1 - x_1^L)\mathbf{s}_{u_2}^{cv}(\bar{\mathbf{z}}) + \boldsymbol{\omega}\left(x_2^L, \mathbf{s}_{x_1}^{cv}(\bar{\mathbf{z}}), \mathbf{s}_{x_1}^{cc}(\bar{\mathbf{z}})\right) \\
&\quad + \boldsymbol{\omega}\left(x_1^L, \mathbf{s}_{x_2}^{cv}(\bar{\mathbf{z}}), \mathbf{s}_{x_2}^{cc}(\bar{\mathbf{z}})\right), \\
\mathbf{s}_{w_5}^{cv}(\bar{\mathbf{z}}) &= (x_2^L - b_2)\mathbf{s}_{o_1}^{cc}(\bar{\mathbf{z}}) + (x_1^L - b_1)\mathbf{s}_{o_2}^{cc}(\bar{\mathbf{z}}) + \boldsymbol{\omega}\left(b_2, \mathbf{s}_{x_1}^{cv}(\bar{\mathbf{z}}), \mathbf{s}_{x_1}^{cc}(\bar{\mathbf{z}})\right) \\
&\quad + \boldsymbol{\omega}\left(x_1^U, \mathbf{s}_{x_2}^{cv}(\bar{\mathbf{z}}), \mathbf{s}_{x_2}^{cc}(\bar{\mathbf{z}})\right), \\
\mathbf{s}_{w_6}^{cv}(\bar{\mathbf{z}}) &= (x_2^L - x_2^U)\mathbf{s}_{o_1}^{cc}(\bar{\mathbf{z}}) + \boldsymbol{\omega}\left(x_2^U, \mathbf{s}_{x_1}^{cv}(\bar{\mathbf{z}}), \mathbf{s}_{x_1}^{cc}(\bar{\mathbf{z}})\right) + \boldsymbol{\omega}\left(b_1, \mathbf{s}_{x_2}^{cv}(\bar{\mathbf{z}}), \mathbf{s}_{x_2}^{cc}(\bar{\mathbf{z}})\right), \\
\mathbf{s}_{w_7}^{cv}(\bar{\mathbf{z}}) &= (x_1^L - x_1^U)\mathbf{s}_{o_2}^{cc}(\bar{\mathbf{z}}) + \boldsymbol{\omega}\left(b_2, \mathbf{s}_{x_1}^{cv}(\bar{\mathbf{z}}), \mathbf{s}_{x_1}^{cc}(\bar{\mathbf{z}})\right) + \boldsymbol{\omega}\left(x_1^U, \mathbf{s}_{x_2}^{cv}(\bar{\mathbf{z}}), \mathbf{s}_{x_2}^{cc}(\bar{\mathbf{z}})\right), \\
\mathbf{s}_{w_8}^{cv}(\bar{\mathbf{z}}) &= (b_2 - x_2^U)\mathbf{s}_{o_1}^{cc}(\bar{\mathbf{z}}) + (b_1 - x_1^U)\mathbf{s}_{o_2}^{cc}(\bar{\mathbf{z}}) + \boldsymbol{\omega}\left(x_2^U, \mathbf{s}_{x_1}^{cv}(\bar{\mathbf{z}}), \mathbf{s}_{x_1}^{cc}(\bar{\mathbf{z}})\right) \\
&\quad + \boldsymbol{\omega}\left(x_1^U, \mathbf{s}_{x_2}^{cv}(\bar{\mathbf{z}}), \mathbf{s}_{x_2}^{cc}(\bar{\mathbf{z}})\right),
\end{aligned}
$$

and

$$
\begin{aligned}
\mathbf{s}^{cc}_{w_1}(\bar{\mathbf{z}}) &= (x_2^L - a_2)\mathbf{s}^{cv}_{u_1}(\bar{\mathbf{z}}) + (a_1 - x_1^U)\mathbf{s}^{cv}_{u_2}(\bar{\mathbf{z}}) + \mathbf{\Omega}\left(a_2, x_1^{cv}(\bar{\mathbf{z}}), x_1^{cc}(\bar{\mathbf{z}})\right) \\
&\quad + \mathbf{\Omega}\left(x_1^U, x_2^{cv}(\bar{\mathbf{z}}), x_2^{cc}(\bar{\mathbf{z}})\right), \\
\mathbf{s}^{cc}_{w_2}(\bar{\mathbf{z}}) &= (x_2^L - x_2^U)\mathbf{s}^{cv}_{u_1}(\bar{\mathbf{z}}) + \mathbf{\Omega}\left(a_1, \mathbf{s}^{cv}_{x_2}(\bar{\mathbf{z}}), \mathbf{s}^{cc}_{x_2}(\bar{\mathbf{z}})\right) + \mathbf{\Omega}\left(x_2^U, \mathbf{s}^{cv}_{x_1}(\bar{\mathbf{z}}), \mathbf{s}^{cc}_{x_1}(\bar{\mathbf{z}})\right), \\
\mathbf{s}^{cc}_{w_3}(\bar{\mathbf{z}}) &= (x_1^L - x_1^U)\mathbf{s}^{cv}_{u_2}(\bar{\mathbf{z}}) + \mathbf{\Omega}\left(a_2, \mathbf{s}^{cv}_{x_1}(\bar{\mathbf{z}}), \mathbf{s}^{cc}_{x_1}(\bar{\mathbf{z}})\right) + \mathbf{\Omega}\left(x_1^U, \mathbf{s}^{cv}_{x_2}(\bar{\mathbf{z}}), \mathbf{s}^{cc}_{x_2}(\bar{\mathbf{z}})\right), \\
\mathbf{s}^{cc}_{w_4}(\bar{\mathbf{z}}) &= (a_2 - x_2^U)\mathbf{s}^{cv}_{u_1}(\bar{\mathbf{z}}) + (x_1^L - a_1)\mathbf{s}^{cv}_{u_2}(\bar{\mathbf{z}}) + \mathbf{\Omega}\left(x_2^U, \mathbf{s}^{cv}_{x_1}(\bar{\mathbf{z}}), \mathbf{s}^{cc}_{x_1}(\bar{\mathbf{z}})\right) \\
&\quad + \mathbf{\Omega}\left(a_1, \mathbf{s}^{cv}_{x_2}(\bar{\mathbf{z}}), \mathbf{s}^{cc}_{x_2}(\bar{\mathbf{z}})\right), \\
\mathbf{s}^{cc}_{w_5}(\bar{\mathbf{z}}) &= (x_2^U - b_2)\mathbf{s}^{cc}_{o_1}(\bar{\mathbf{z}}) + (b_1 - x_1^L)\mathbf{s}^{cc}_{o_2}(\bar{\mathbf{z}}) + \mathbf{\Omega}\left(b_2, \mathbf{s}^{cv}_{x_1}(\bar{\mathbf{z}}), \mathbf{s}^{cc}_{x_1}(\bar{\mathbf{z}})\right) \\
&\quad + \mathbf{\Omega}\left(x_1^L, \mathbf{s}^{cv}_{x_2}(\bar{\mathbf{z}}), \mathbf{s}^{cc}_{x_2}(\bar{\mathbf{z}})\right), \\
\mathbf{s}^{cc}_{w_6}(\bar{\mathbf{z}}) &= (x_2^U - x_2^L)\mathbf{s}^{cc}_{o_1}(\bar{\mathbf{z}}) + \mathbf{\Omega}\left(b_1, \mathbf{s}^{cv}_{x_2}(\bar{\mathbf{z}}), \mathbf{s}^{cc}_{x_2}(\bar{\mathbf{z}})\right) + \mathbf{\Omega}\left(x_2^L, \mathbf{s}^{cv}_{x_1}(\bar{\mathbf{z}}), \mathbf{s}^{cc}_{x_1}(\bar{\mathbf{z}})\right), \\
\mathbf{s}^{cc}_{w_7}(\bar{\mathbf{z}}) &= (x_1^U - x_1^L)\mathbf{s}^{cc}_{o_2}(\bar{\mathbf{z}}) + \mathbf{\Omega}\left(b_2, \mathbf{s}^{cv}_{x_1}(\bar{\mathbf{z}}), \mathbf{s}^{cc}_{x_1}(\bar{\mathbf{z}})\right) + \mathbf{\Omega}\left(x_1^L, \mathbf{s}^{cv}_{x_2}(\bar{\mathbf{z}}), \mathbf{s}^{cc}_{x_2}(\bar{\mathbf{z}})\right), \\
\mathbf{s}^{cc}_{w_8}(\bar{\mathbf{z}}) &= (b_2 - x_2^L)\mathbf{s}^{cc}_{o_1}(\bar{\mathbf{z}}) + (x_1^U - b_1)\mathbf{s}^{cc}_{o_2}(\bar{\mathbf{z}}) + \mathbf{\Omega}\left(x_2^L, \mathbf{s}^{cv}_{x_1}(\bar{\mathbf{z}}), \mathbf{s}^{cc}_{x_1}(\bar{\mathbf{z}})\right) \\
&\quad + \mathbf{\Omega}\left(b_1, \mathbf{s}^{cv}_{x_2}(\bar{\mathbf{z}}), \mathbf{s}^{cc}_{x_2}(\bar{\mathbf{z}})\right),
\end{aligned}
$$

where $\mathbf{s}^{cv}_{x_1}$, $\mathbf{s}^{cc}_{x_1}$, $\mathbf{s}^{cv}_{x_2}$, $\mathbf{s}^{cc}_{x_2}$, $\mathbf{s}^{cv}_{u_1}$, $\mathbf{s}^{cc}_{u_1}$, $\mathbf{s}^{cv}_{u_2}$, $\mathbf{s}^{cc}_{u_2}$, $\mathbf{s}^{cv}_{o_1}$, $\mathbf{s}^{cc}_{o_1}$, $\mathbf{s}^{cv}_{o_2}$, $\mathbf{s}^{cc}_{o_2}$ are, respectively, subgradients of x_1^{cv}, x_1^{cc}, x_2^{cv}, x_2^{cc},u_1^{cv}, u_1^{cc}, u_2^{cv}, u_2^{cc}, o_1^{cv}, o_1^{cc}, o_2^{cv}, o_2^{cc} on Z. Further, let $q_{max} \in \operatorname{argmax}\left\{w_1^{cv}(\bar{\mathbf{z}}), \ldots, w_8^{cv}(\bar{\mathbf{z}}), w^U\right\}$ and $q_{min} \in \operatorname{argmin}\left\{w_1^{cc}(\bar{\mathbf{z}}), \ldots, w_8^{cc}(\bar{\mathbf{z}}), w^L\right\}$, then

$$
\mathbf{s}^{cv}_{w}(\bar{\mathbf{z}}) = \begin{cases} \mathbf{s}^{cv}_{w_{q_{min}}}(\bar{\mathbf{z}}), & \text{if} \quad 1 \leq q_{min} \leq 8, \\ \mathbf{0}, & \text{otherwise}, \end{cases}
$$

and

$$\mathbf{s}_w^{cc}(\bar{\mathbf{z}}) = \begin{cases} \mathbf{s}_{w_{q_{max}}}^{cc}(\bar{\mathbf{z}}), & \text{if} \quad 1 \leq q_{max} \leq 8, \\ \mathbf{0}, & \text{otherwise.} \end{cases}$$

Proof. The proof follows from the construction of the relaxations in Theorem 5.2.3. The functions defined in Definition 5.2.5 select subgradients that respect the rules for scalar multiplication of relaxations [191]. For each equation (5.2.19) - (5.2.26), the subgradients are then summed using the standard additive relationship [133, 191]. □

5.3 Computability of Tight Composite Relaxations of Bilinear Terms

In this section, we describe three approaches to compute the requisite *a priori* relaxations in a reduced-space McCormick relaxation context.

5.3.1 Composite Convex/Concave Relaxations based on Over/Underestimators

For low-dimensional expressions, we may exploit the properties of convex/concave functions. We may use convex/concave relaxations of the arguments of the bilinear expression as the valid *a priori* relaxations. As illustrated in a subsequent example, if the a_1, a_2, b_1, b_2 values are selected judiciously, then this may lead to nontrivial affine relaxations and, in turn, improved relaxations of the bilinear term in reduced-space. The constants a_1 and a_2 can be selected by maximizing convex functions $x_1^{cv}(\cdot)$ and $x_2^{cv}(\cdot)$ on a convex polyhedron $\mathcal{P} = \text{conv}(v_1, v_2, \ldots, v_k)$. It is well-known that

the extremal value will be achieved at a vertex, i.e., $\max_{z\in\mathcal{P}} f(z) = \max_i f(v_i)$. Similarly, we may compute b_1, b_2 by minimizing the concave functions $x_1^{cc}(\cdot)$ and $x_2^{cc}(\cdot)$. This is in itself a series of nonconvex optimization problems. While specialized algorithms may exist to address this class of problems (e.g., [30, 101]), it is reasonable to conclude that this approach is too computationally expensive to be practical, as two nonsmooth concave optimization problems must be solved for each intermediate bilinear term in order to evaluate relaxations of the nonlinear function.

In the case of low-dimensional expressions, we may simply compute convex/concave relaxations at each vertex and solve each optimization problem via enumeration. In many cases, the evaluation of relaxations is often much less time intensive than other routines, such as solving a series of linear programs in optimization-based bounds tightening [234], or solving a nonlinear program in order to furnish valid upper bounds within a branch-and-bound routine for deterministic global optimization [136]. One may intuit that this method will yield a tighter relaxation of the bilinear term than using a weaker *a priori* relaxation. As it turns out, this is in fact false, and the use of a weaker *a priori* relaxation may lead to tighter relaxations of the bilinear term owing to the dependence of the relaxations of Theorem 5.2.3 on a_1, a_2, b_1, and b_2. A counterexample is provided in Example 1 and illustrated by in Figure 5.3.1. In the subsequent section, a less computationally expensive method is developed that may provide comparably tight bounds.

5.3.2 Improved Relaxations Using Subgradient-Based Under/Overestimators

In addition to the composite bilinear relaxation theory outlined herein, the use of *a priori* relaxations to refine the relaxations of a univariate factor can be accomplished via Proposition 5.3.1.

Proposition 5.3.1. Let $v_k : Z \subset \mathbb{R}^n \to V$ be a cumulative mapping. Let $v_{k,a}^{cv}/v_{k,a}^{cc}$ be convex/concave relaxations of v_k on Z, and suppose we have additional convex/concave relaxations $v_{k,b}^{cv}/v_{k,b}^{cc}$ of v_k on Z. Then,

$$v_k^{cv}(\cdot) := \max\left\{v_{k,a}^{cv}(\cdot), v_{k,b}^{cv}(\cdot)\right\}, \tag{5.3.1}$$

$$v_k^{cc}(\cdot) := \min\left\{v_{k,a}^{cc}(\cdot), v_{k,b}^{cc}(\cdot)\right\}, \tag{5.3.2}$$

are convex and concave relaxations of v_k on Z, respectively.

The subgradients associated with Proposition 5.3.1 are then simply the subgradients of the argument returned. In Proposition 5.3.2, we note that, provided with convex/concave relaxations of a factor at a particular point $\bar{\mathbf{z}} \in Z$ along with associated subgradients, then new affine relaxations may be derived.

Proposition 5.3.2. Let $v_k : Z \subset \mathbb{R}^n \to V$ be a cumulative mapping. Let v_k^{cv}/v_k^{cc} be convex/concave relaxations of v_k on Z and their respective subgradients $\mathbf{s}_{v_k}^{cv}$, $\mathbf{s}_{v_k}^{cc}$ computed at $\mathbf{z} = \bar{\mathbf{z}} \in Z$. The functions $\xi, \zeta : Z \to \mathbb{R}$ are the affine relaxations of the convex and concave relaxations of v_k on Z, respectively:

$$\xi(\mathbf{z}) \equiv v_k^{cv}(\bar{\mathbf{z}}) + \mathbf{s}_{v_k}^{cv}(\bar{\mathbf{z}})^{\mathrm{T}}(\mathbf{z} - \bar{\mathbf{z}}) \tag{5.3.3}$$

$$\zeta(\mathbf{z}) \equiv v_k^{cc}(\bar{\mathbf{z}}) + \mathbf{s}_{v_k}^{cc}(\bar{\mathbf{z}})^{\mathrm{T}}(\mathbf{z} - \bar{\mathbf{z}}). \tag{5.3.4}$$

As noted in [206], interval extensions of (5.3.3) and (5.3.4) can be used to derive valid upper bounds of the factor and subsequently refine the associated interval bounds. These same interval extensions define valid a_1, a_2, b_1, b_2 terms for the affine relaxations and allow for the direct use of Theorem 5.2.3 to improve the convex/concave relaxations as well.

The following numerical example is provided to illustrate the results of applying Theorems 5.2.3 and 5.2.6 using the methods of Sections 5.3.1 and 5.3.2 versus the previously established McCormick-based approaches.

Example 1. For illustrative purposes, consider the function $f : Z \to \mathbb{R}$, with $Z = [-0.5, 1]$, defined as

$$f(z) = \left(z - z^2\right)\left(z^3 - \exp(z)\right). \tag{5.3.5}$$

As illustrated in Figure 5.3.1, *a priori* affine relaxations constructed at a single reference point $\bar{z} = 0.25$ yield similar relaxations to the direct enumeration approach. Note that in this example, neither approach yields relaxations that are a strict improvement over the other for the entire domain.

5.3.3 Affine Arithmetic

Affine arithmetic has been proposed as an alternative set-valued arithmetic to interval arithmetic. In this approach, an affine representation of a function is constructed. In the case of affine functions, this representation is exact. For nonlinear terms, the enclosure is linearized and some overestimation necessarily occurs. Two common choices of linearization techniques include minimizing the range of the enclosure or minimizing the maximum width of enclosure (Chebyshev). In the original description of affine arithmetic [64] and the later introductory papers [72, 295], computations began with an affine representation of each term and an additional noise term was added for every nonlinear term introduced. This approach introduced significant computational complexity. In this work, we will instead address the use of two simplified forms of affine arithmetic that were proposed by [179]. In each of these forms, the intermediate term

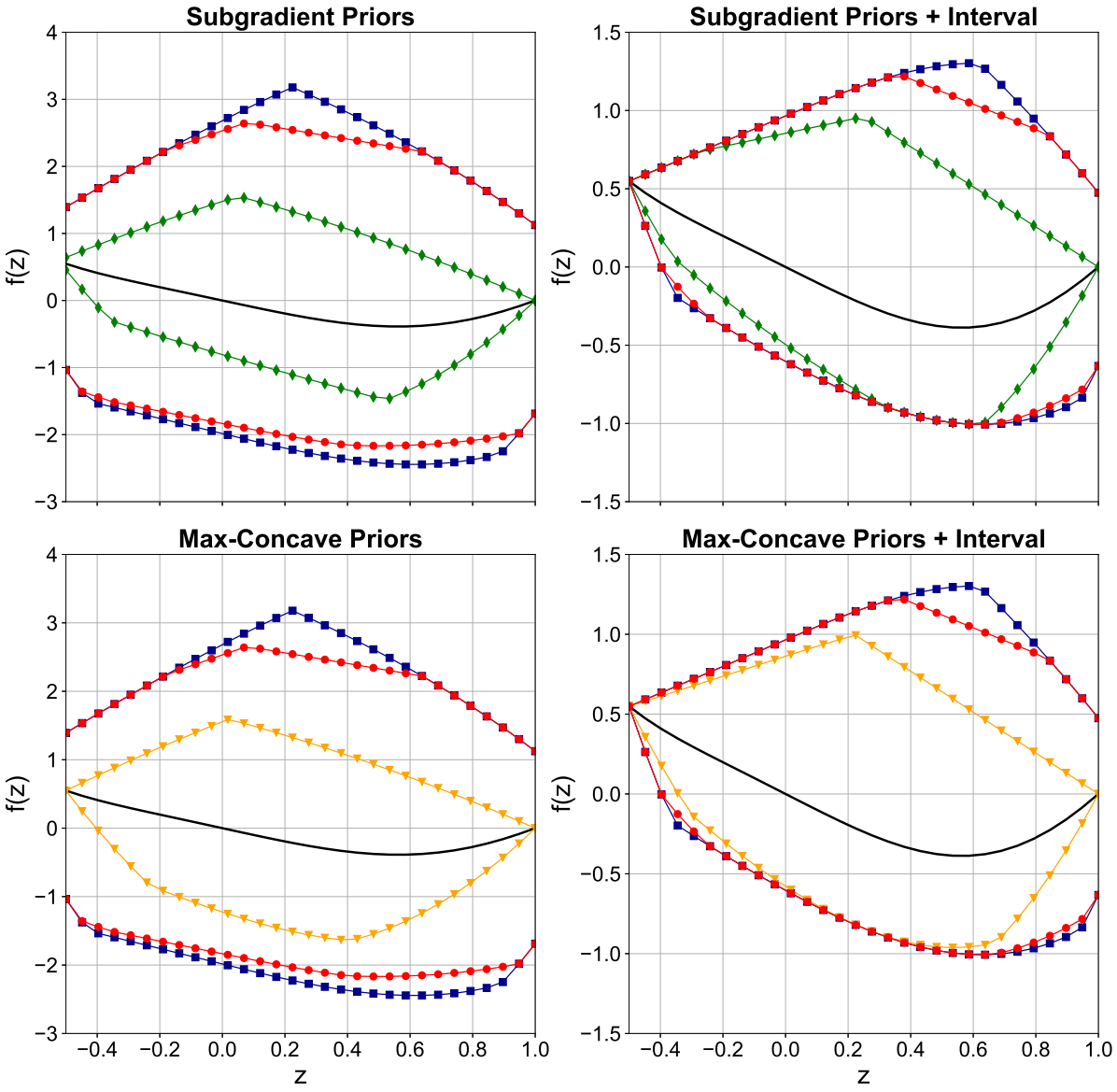

FIGURE 5.3.1: Relaxations of $f(z) = (z - z^2)(z^3 - \exp(z))$ (—) on $Z = [-0.5, 1]$, are constructed using existing approaches and compared with approaches developed in this chapter. Relaxations computed using *a priori* subgradients at $\bar{z} = 0.25$ (♦) lead to tighter relaxations than the use of standard (■), and multivariate (●) McCormick relaxation strategies. This occurs when subgradients are (**top-left**) only used as *a priori* relaxations, as well as when the subgradients are (**top-right**) used to refine the interval bounds of each factor [206]. (**bottom-left**) The *a priori* relaxations constructed by computing the maxima and minima of the operands' relaxations (▽) also lead to an improvement. (**bottom-right**) Minimal subsequent improvement is noted when using the subgradient method to refine interval bounds of each factor [206].

$v_k(\mathbf{z}) \in \tilde{v}_0 + \sum_{i=1}^{n} \tilde{v}_i \mathbf{z} + R_k$ is represented by a linear function such that with a small remainder R_k that encloses truncation error [216]. The first affine form, **AF1,** uses a single component to represent the R_k truncation error.

Definition 5.3.3. Affine Form of Type AF1

$$\hat{v} = v_0 + \sum_{i=1}^{n} v_i \epsilon_i + v_{n+1} \epsilon_{\pm} \tag{5.3.6}$$

where $\epsilon_i \in [-1, 1]$, $v_i \in \mathbb{R}$ for $i = 1, \ldots n + 1$, $\epsilon_{\pm} \in [-1, 1]$, and $v_0 \in \mathbb{R}$

In addition to this form, [179] discussed the use of a second affine form, **AF2,** which uses separate components to represent positive truncation error, negative truncation error, and mixed-sign truncation error. This distinction leads to tighter enclosures of certain operators.

Definition 5.3.4. Affine Form of Type AF2

$$\hat{v} = v_0 + \sum_{i=1}^{n} v_i \epsilon_i + v_{n+1} \epsilon_{\pm} + v_{n+2} \epsilon_{+} + v_{n+3} \epsilon_{-} \tag{5.3.7}$$

where $v_i \in \mathbb{R}$ for $i = 1, \ldots n + 3$, $\epsilon_i \in [-1, 1]$ for $i = 1, \ldots n$, $\epsilon_{\pm} \in [-1, 1]$, $\epsilon_{+} \in [0, 1]$, $\epsilon_{-} \in [-1, 0]$, and $v_0 \in \mathbb{R}$

Either of the preceding affine forms implies the existence of affine relaxations as described by [222]. We proceed to state the corresponding relaxations in Propositions 5.3.5 (adapted from [222, Prop. 2]) and 5.3.6 (adapted from [222, Prop. 3]). Next, note that components ϵ_i are simply nondimensionalized decision variables and can be converted to the dimensional form using the following equation

$$\epsilon_i = (z_i - \text{mid}(Z_i))/\text{rad}(Z_i) \tag{5.3.8}$$

Proposition 5.3.5 (Affine Relaxation from AF1 Form)**.** Let $v : Z \to V$ be a factor with an affine representation detailed in Definition 5.3.3. Then $v^{cv}_{AF1}, v^{cc}_{AF1} : Z \to V$ are affine relaxations of v at $\mathbf{z} \in Z$.

$$v^{cv}_{AF1}(\mathbf{z}) = v_0 + \sum_{i=1}^{n} v_i \epsilon_i - v_{n+1} \tag{5.3.9}$$

$$v^{cc}_{AF1}(\mathbf{z}) = v_0 + \sum_{i=1}^{n} v_i \epsilon_i + v_{n+1} \tag{5.3.10}$$

Proposition 5.3.6 (Affine Relaxation from AF2 Form)**.** Let $v : Z \to V$ be a factor with an affine representation detailed in 5.3.4 then $v^{cv}_{AF2}, v^{cc}_{AF2} : Z \to V$ are affine relaxations of v at $\mathbf{z} \in Z$.

$$v^{cv}_{AF2}(\mathbf{z}) = v_0 + \sum_{i=1}^{n} v_i \epsilon_i - v_{n+1} - v_{n+2} - v_{n+3} \tag{5.3.11}$$

$$v^{cc}_{AF2}(\mathbf{z}) = v_0 + \sum_{i=1}^{n} v_i \epsilon_i + v_{n+1} + v_{n+2} + v_{n+3} \tag{5.3.12}$$

The extrema of (5.3.9), (5.3.10), (5.3.11), and (5.3.12) on Z can then readily be computed via natural intervals extensions. We note that this operation is no more complicated than converting the affine representation to an interval form. As illustrated in Example 2, the use of apriori information propagated through affine forms may yield tighter relaxations than simply using interval bounds calculated via affine arithmetic.

Example 2. Consider the function $f : X \times Y \to \mathbb{R}$ on the domain $X \times Y = [0.1, 1.9]^2$, defined as

$$z = f(x, y) = (x^2 - x)(y^2 - y), \tag{5.3.13}$$

We compute convex and concave relaxations of this function using four distinct approaches: (1) standard McCormick arithmetic, (2) affine arithmetic of style AF1, (3) composite relaxations

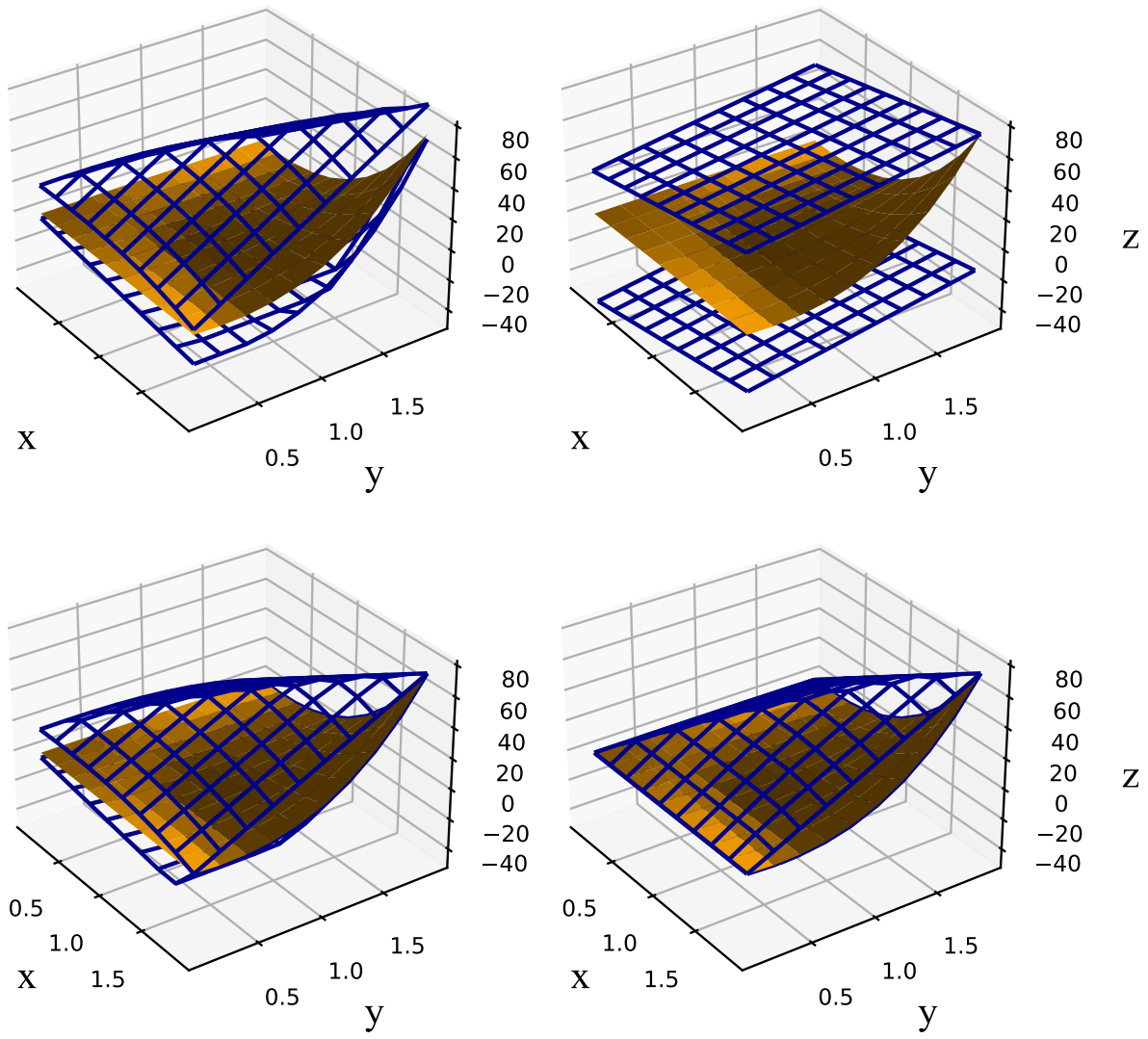

FIGURE 5.3.2: The function $z = (x^2 - x)(y^2 - y)$ (■) along with the corresponding convex and concave relaxations of (—) on $X \times Y = [0.1, 1.9]$ are presented. The McCormick relaxations approach (**top-left**) is contrasted to relaxations implied by affine arithmetic (AF1, **top-right**). The use of composite relaxations formed by intersecting affine enclosures with McCormick relaxations (**bottom-left**) is compared the relaxations implied by the use of apriori bounds per Theorems 5.2.1 - 5.2.3 (**bottom-right**).

taken by intersecting relaxations derived using (1) & (2) using standard multiplication rules, and (4) composite relaxations generate using (3) were the relaxations of the bilinear operator are computed using Theorems 5.2.1 - 5.2.3. The results illustrated in Figure 5.3.2. It is clear that the

relaxations computing using (3) outperform both (1) & (2) and (4) further tightens the relaxations obtained by (3).

5.4 Benchmark Results

All numerical experiments in this work were run on a single thread of an Intel Xeon E3-1270 v5 3.60/4.00GHz (base/turbo) processor with 16GB ECC RAM allocated to a virtual machine running the Ubuntu 18.04LTS operating system and Julia v1.6 [36]. Absolute and relative convergence tolerances for the B&B algorithm of 10^{-4} were specified for all example problems, unless otherwise noted. The EAGO.jl package [327] was used to solve each optimization problem. Relaxations of intrinsic functions have been implemented in the McCormick.jl [329] subpackage of EAGO.jl and is openly available. BARON v21.1.13 [253, 305] was used for performance comparisons. The Intel MKL (2019 Update 2) [95] was used to perform all LAPACK [11, 320] and BLAS [40] routines. The data used with and generated from the following numerical examples are openly available in the following Git repository `https://github.com/PSORLab/RSBilinear` along with the corresponding problem formulations. Let **EAGO** denote the use of relaxation-based *a priori* relaxations for nonlinear terms, **EAGO Sub** denote the use of subgradient-based *a priori* relaxations, and **EAGO Aff** denote the use of affine-arithmetic and associated *a priori* relaxations. A randomly generated benchmark library adapted from Example 5 of He work[127] was used to assess the performance of each approach. Each instances takes the form given by 5.4.1

$$
\begin{aligned}
&\min_{\mathbf{x}} \langle \mathbf{c}, \mathbf{x} \rangle + Q \circ Y \\
&\qquad s.t. \qquad \mathbf{x} \in [-1, 1]^n, \\
&\qquad\qquad \mathbf{y} = (x_1^2 - x_1, x_1^3 - x_1, x_1^4 - x_1, \ldots, x_n^2 - x_n, x_n^3 - x_n, x_n^4 - x_n), \\
&\qquad\qquad \mathbf{Y} = \mathbf{y}^T \mathbf{y}
\end{aligned} \tag{5.4.1}
$$

where $\circ$ denotes the entry-wise product, $Q \in \mathbb{R}^{n,n}$ is a strictly upper triangular matrix with density of nonzero elements in the upper triangular section given by ν. The $\mathbf{c} \in \mathbb{R}^n$ vector is in $[-512, -2]$. A set of 200 instances with randomly determined $\nu \in [0.3, 0.5, 0.7]$ and $n \in [10, 15, 20]$ was then generated and solved for each solver configuration. A 5 minute (300 second) CPU time limit was enforced for each instance. Solver performance was assessed using the shifted geometric mean time. A performance profile was generated for comparison using the methodology of Dolan and Moré [80]. The *performance* of a solver configuration s is set to the solution time $t_{p,s}$ in CPU seconds (single-threaded) for problem p. The *performance ratio* on problem p by solver s is then the ratio of solver s performance to the best solver performance in the set:

$$
r_{p,s} = \frac{t_{p,s}}{\min\{t_{p,s} : s \in S\}}.
$$

This *performance profile* of solver s on a benchmark set depicts the distribution function of the performance metric, $\rho_s(\tau)$; which is the probability that a performance ratio $r_{p,s}$ is within $\tau \in \mathbb{R}$ of the best possible ratio

$$
\rho_s(\tau) = \frac{1}{n_p} \text{size}\{p \in \mathcal{P} : r_{p,s} \leq \tau\},
$$

where $\mathcal{P}$ is the set of problems with $n_p = \text{card}(\mathcal{P})$. A plot comparing r_s for each configuration $s \in S$ then illustrates the relative performance.

First, we note that based on the performance data available in Tables 5.4.1 to 5.4.2 and Figure 5.4.1 all versions of EAGO significantly under-perform BARON in this benchmark. This result is to be expected as the benchmark set is a polynomial optimization problem which may be reformulated as a higher dimensional nonconvex quadratic program. Provided that this reformulation occurs BARON may implement a number of specialized approaches [20, 21, 223, 224, 340] which currently have no analog for reduced-space optimization. However, as we discussed previously these approaches cannot be readily applied to reduced-space applications in which the problem does not have a factorable representation.

Next, we observe that the **EAGO Sub** reduces the shifted geometric mean run time relative to **EAGO** by a factor of 3 increasing the number of problems solved within 5 minutes by 23.5% as illustrated by the Tables 5.4.1 to 5.4.2. Interestingly, **EAGO Aff** actually increases the mean solve time relative to **EAGO** as the increased time spent performing Affine Arithmetic calculations offsets any potential benefit from reducing over-estimation that occurs in the relaxed problem. As subgradients are already calculated when computing relaxations in **EAGO** the use of subgradients to tighten the composite relaxation of the bilinear term in **EAGO Sub** does not substantially increase calculation time.

5.5 Case Study: Process Optimization Through Sequential Response Surface Methodology (RSM)

The response surface model (RSM) are one of the most common statistical models used in industrial applications [73]. Models of this form consist of simple quadratic function which

Solver Configuration	Solved	Unsolved
BARON	200 (100.0%)	0 (0.0%)
EAGO	136 (68.0%)	64 (32.0%)
EAGO Aff	127 (63.5%)	73 (36.5%)
EAGO Sub	183 (91.5%)	17 (8.5%)

TABLE 5.4.1: The number of problems solved within 5 minutes by solver configuration in the benchmark set are tabulated.

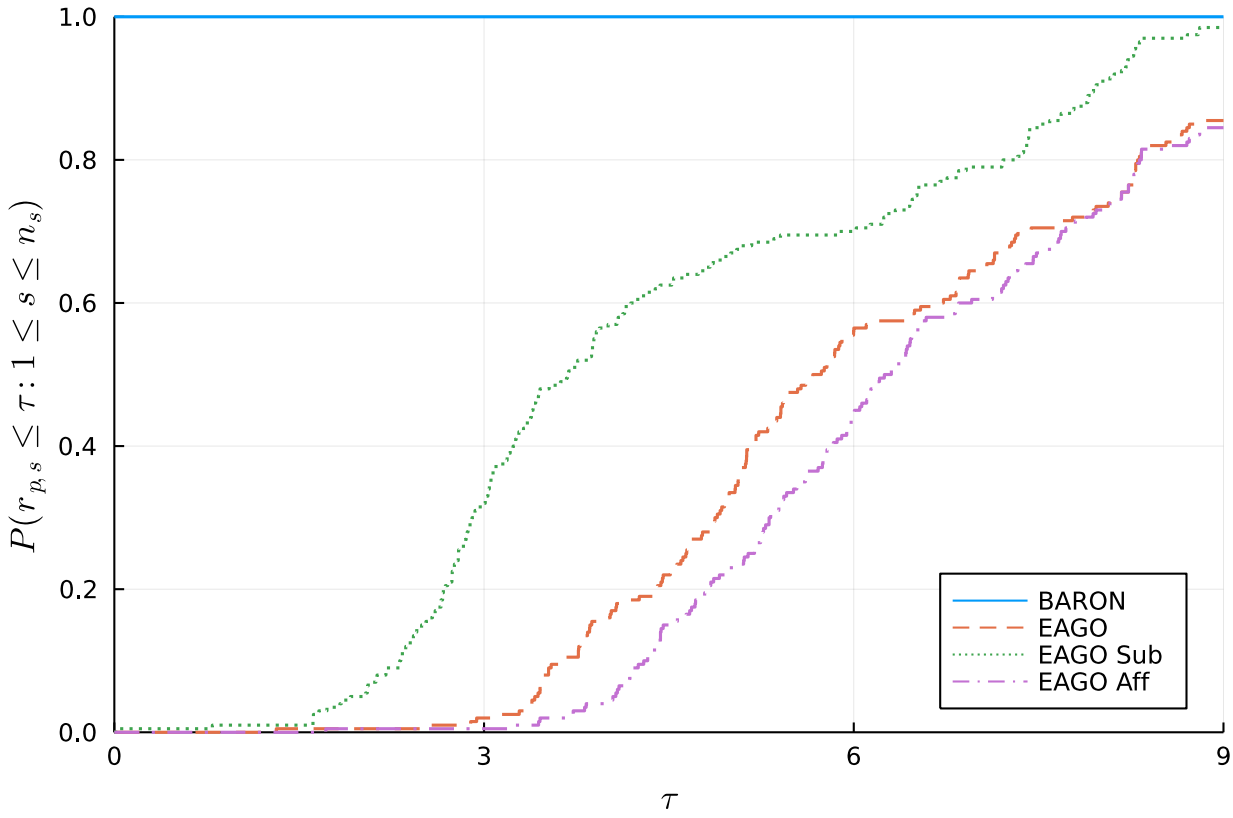

FIGURE 5.4.1: As illustrated by the performance profiles, computing relaxations using the apriori relaxation based on subgradients leads to a substantial decrease in CPU solution time for a typical problem within the benchmark set when compared the naïve McCormick approach implemented in EAGO. The horizontal line plot for BARON indicates that it uniformly provides substantially faster run times.

Solver Configuration	τ	δ_{rel}
BARON	0.69	N/A
EAGO	40.7	3.8×10^{-2}
EAGO Aff	52.4	4.6×10^{-2}
EAGO Sub	13.7	1.2×10^{-2}

TABLE 5.4.2: The shifted geometric mean, τ, of solve times $t_1, t_2, \ldots, t_n$ defined by $\tau = (\prod_{i=1}^{n}(t_i + s))^{1/n} - s$ are given by solver configuration with $s = 1$ along with the average relative gap remaining for any problems not solved within the 5-minute time limit. For problems that terminate due to the specified time limit, the relative gap remaining can be compared to assess solver performance. The relative gap remaining is given by $\delta_{rel} = (—U| - —L|)/\max(—U|, —L|)$ where $—U|$ is the upper bound (best feasible objective value) and $—L|$ is the lower bound.

approximates an underlying function:

$$y = \mathbf{a} + \mathbf{c}^T\mathbf{x} + \tfrac{1}{2}\mathbf{x}^T Q\mathbf{x} \tag{5.5.1}$$

These models are particularly appealing when first-principle process models are highly complex but variation in process outputs with respect to control variables behave in a predictable manner. Applications of RSMs span a wide range with examples including machining via abrasive waterjet turning [338], Nd:YAG laser drilling [241], electron beam welding [196] plasma spray coating [228], and diffusion bonding of alloys [240]. While these models are ubiquitous, the use of these models in optimal design problems readily leads to nonconvex problem formulations due to the presence of bilinear terms. One particularly interesting area of application for RSMs lies in the quality chain design of multistage manufacturing systems [128]. Here we revisit the numerical example which was previously addressed in [128] using local and stochastic methods. We will show that this model form may readily be addressed using reduced-space global optimization and solved to a certificate of global optimality using a simplified set of RSM models

generating from the original data provided in [128]. The manufacturing process consists of an initial shaft machining (termed stage 1) in which the output diameter ($y_1^{(1)}$), output roundness ($y_2^{(1)}$), and ($y_3^{(1)}$) process time are determined by feed rate (x_1), and cut depth (x_2) and input roundness (c_1) of the shaft bought from a supplier. This step is followed by a rough machining process (stage 2) wherein rougher rate (x_3), the part diameter, and machine type (z_1, z_2) determine the output diameter ($y_1^{(2)}$) and process time ($y_2^{(2)}$). The process concludes with finish machining (stage 3) in which finish rate (x_4) is adjusted to determine final diameter ($y_1^{(3)}$) and process time ($y_2^{(3)}$). We consider the case in which a nominal input diameter of 12 is sourced from the supplier. Machine operating parameters and a nominal roundness specification may then be varied to specify the process. Each process step is relates inputs and operating parameters to outputs by means of a response-surface model shown below:

$$\hat{y}_1^{(1)}(\mathbf{x}) = 3.55 + 0.27c_1 + 0.58c_2 + 60.6x_2 - 2.8c_1x_2 - 2.3c_2x_2 \tag{5.5.2}$$

$$\begin{aligned}\hat{y}_2^{(2)}(\mathbf{x},\mathbf{z}) = 126586.5 &- 21466.8\hat{y}_1^{(1)}(\mathbf{x}) + 520.43x_3 + 56.29z_1 + 315.95z_2 \\ &- 43.72x3\hat{y}_1^{(1)}(\mathbf{x}) + 3.74x3^2 + 910.1\hat{y}_1^{(1)}(\mathbf{x})^2 \\ &- 30.6\hat{y}_1^{(1)}(\mathbf{x})z_1 - 173.17\hat{y}_1^{(1)}(\mathbf{x})z_2\end{aligned} \tag{5.5.3}$$

$$\begin{aligned}\hat{y}_1^{(3)}(\mathbf{x},\mathbf{z}) = 9.16 &+ 0.092x_3 + 0.73x_4 + 0.64x_3x_4 - 0.49x_4^2 - 0.13x_4\hat{y}_2^{(2)}(\mathbf{x},\mathbf{z}) \\ &+ 0.0019\hat{y}_2^{(2)}(\mathbf{x},\mathbf{z})^2 + 0.018\hat{y}_2^{(2)}(\mathbf{x},\mathbf{z})\end{aligned} \tag{5.5.4}$$

The original work concerned itself principally with minimizing the variation in $\mathbf{y}^{(3)}$. We instead consider the problem of identifying a nominal process in which value of $\mathbf{y}^{(3)}$ which is close to the desired value of $v = 5$. This is motivated by balancing a maximal process capability index (CpK) with respect to the final diameter setpoint and the desire for just-in-time completion of the

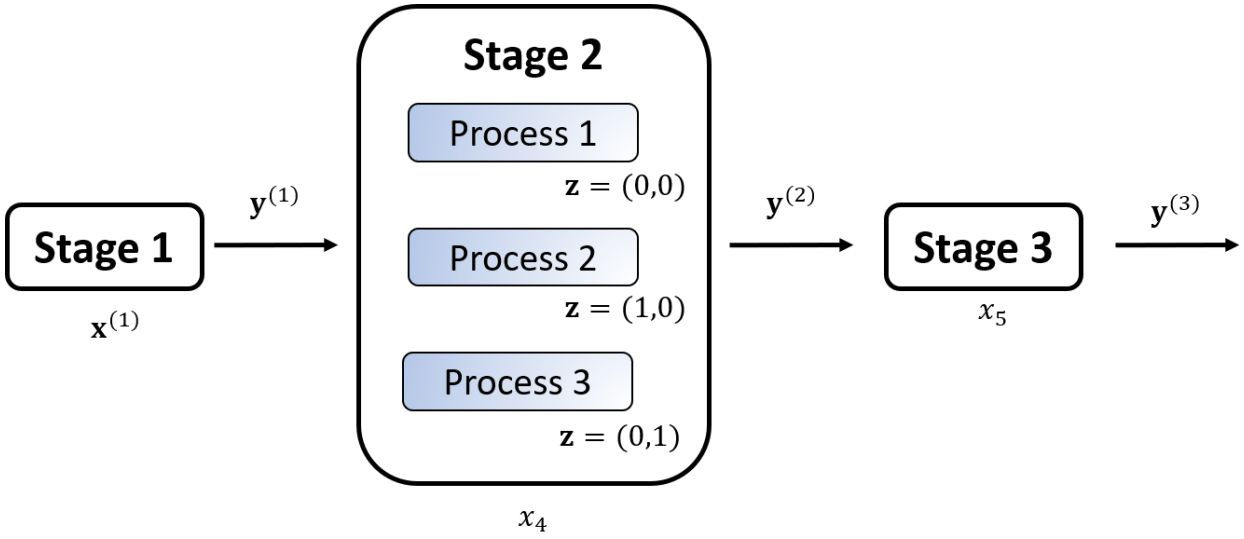

FIGURE 5.5.1: An illustration of the three stage machining process consisting of (1) initial shaft machining process, (2) followed by one of three roughing processes, and processed in a final (3) surface finishing step.

process. As such, the design decision may be formulated as the following optimization problem:

$$\begin{aligned} f^* = \min_{\mathbf{x}\in X, \mathbf{z}\in Z} & \left(\hat{y}_1^{(3)}(\mathbf{x},\mathbf{z}) - v\right)^2 \\ \text{s.t.} \;\; & z_1 + z_2 \leq 1, \end{aligned} \tag{5.5.5}$$

where $\mathbf{x} \in X = [7, 10] \times [0.1, 0.3] \times [-1.078, 1.078] \times [-1.078, 1.078]$, and $C = [0.001, 0.03]$.

We then proceed to solve this problem in EAGO as well as using relaxation-based *a priori* relaxations for nonlinear terms (**EAGO + Relax**), subgradient-based *a priori* relaxations (**EAGO + Sub**), and an affine arithmetic *a priori* relaxation (**EAGO + Aff**). A solve time of 26.3 seconds is required for EAGO to furnish a solution that is feasible in the original problem while a longer solver time of 47.6 seconds is required for the (**EAGO + Aff**) method. The relaxation-based *a priori* relaxations for nonlinear terms (**EAGO + Relax**) leads to a slightly faster run time of 25.5 seconds while (**EAGO + Sub**) yields a significantly faster run time of 7.1 seconds.

5.6 Case Study: Kinetic Parameter Estimation

The apriori relaxation methods presented here can readily be applied to dynamic optimization problems as well. We demonstrate this using an adaption of a kinetic parameter estimation problem [191]. The reaction mechanism is described by the initial value problem:

$$\begin{aligned}
\frac{dx_A}{dt} &= k_1 x_Z x_Y - c_{O_2}(k_{2f} + k_{3f})x_A + \frac{k_{2f}}{K_2}x_D + \frac{k_{3f}}{K_3}x_B - k_5 x_A^2 \\
\frac{dx_B}{dt} &= c_{O_2}k_{3f}x_A - \left(\frac{k_{3f}}{K_3} + k_4\right)x_B, \quad \frac{dx_Z}{dt} = -k_1 x_Z x_Y \\
\frac{dx_D}{dt} &= c_{O_2}k_{2f}x_A - \frac{k_{2f}}{K_2}x_D, \quad \frac{dx_Y}{dt} = -k_{1s} x_Z x_Y \\
x_A(0) &= 0, x_B(0) = 0, x_D(0) = 0, x_Y(0) = 0.4, x_Z(0) = 140
\end{aligned}$$

where x_j is the concentration of species $j \in \{A, B, D, Y, Z\}$ and the constants are given by $T = 273$, $K_2 = 46\exp(6500/T - 18)$, $K_3 = 2K_2$, $k_1 = 53$, $k_{1s} = k_1 \times 10^{-6}$, $k_5 = 1.2 \times 10^{-3}$, and $c_{O_2} = 2 \times 10^{-3}$. A least squares fit is sought to fit available intensity and time data which exhibit a known dependency on the concentration, that is, $I = x_A + \frac{2}{21}x_B + \frac{2}{21}x_D$ [283]. The reaction rate constants $k_{2f} \in [10, 1200]$, $k_{3f} \in [10, 1200]$, and $k_4 \in [0.001, 40]$ are the decision variables $\mathbf{p} = (k_{2f}, k_{3f}, k_4)$.

We consider an explicit Euler discretization of the problem [191] for simplicity sake. A semi-explicit approach is used in which the relaxations of state variables in the ODE, $\mathbf{x}$, are computed. A discretization consisting of 50 steps is sufficient for a high-degree of accuracy for this problem on the time domain $t \in [0, 0.5]$. The discretized model becomes:

$$
\begin{aligned}
x_A^{i+1} &= x_A^i + \Delta t\left(k_1 x_Y^i x_Z^i - c_{O_2}(k_{2f}+k_{3f})x_A^i + \frac{k_{2f}}{K_2}x_D^i + \frac{k_{3f}}{K_3}x_B^i - k_5(x_A^i)^2\right)\\
x_B^{i+1} &= x_B^i + \Delta t\left(k_{3f}c_{O_2}x_A^i - \left(\frac{k_{3f}}{K_3}+k_4\right)x_B^i\right)\\
x_D^{i+1} &= x_D^i + \Delta t\left(k_{2f}c_{O_2}x_A^i - \frac{k_{2f}}{K_2}x_D^i\right)\\
x_Y^{i+1} &= x_Y^i + \Delta t\left(-k_{1s}x_Y^i x_Z^i\right)\\
x_Z^{i+1} &= x_Z^i + \Delta t\left(-k_{1s}x_Y^i x_Z^i\right)
\end{aligned}
$$

where $i = 0, \ldots, 49$ and $\Delta t = 0.01$. The objective function is then given by

$$
f(\mathbf{p}) = \sum_{i=1}^{n}\left(I_i^c(\mathbf{p}) - I_i^d\right)^2 \tag{5.6.1}
$$

where I_i^c are the calculated intensity values at time step i from the model and I_i^d are experimental data. We solve this problem in **EAGO** as well as using subgradient-based *a priori* relaxations, **EAGO Sub**. The standard **EAGO** approach solves this problem in 450.0 seconds and 121,881 iterations whereas the **EAGO Sub** method solves this problem in only 94.5 seconds after 8,905 iterations as evidence from the relative gap convergence plot provided in Figure 5.6.1. This illustrates how dynamic problems structures that introduce a large number of composite bilinear terms may benefit substantially from the use of the a priori relaxation approach presented herein.

5.7 Concluding Remarks

A theorem was developed for computing improved relaxations of composite bilinear terms when relaxations of *a priori* underestimators and overestimators are available. A corresponding result

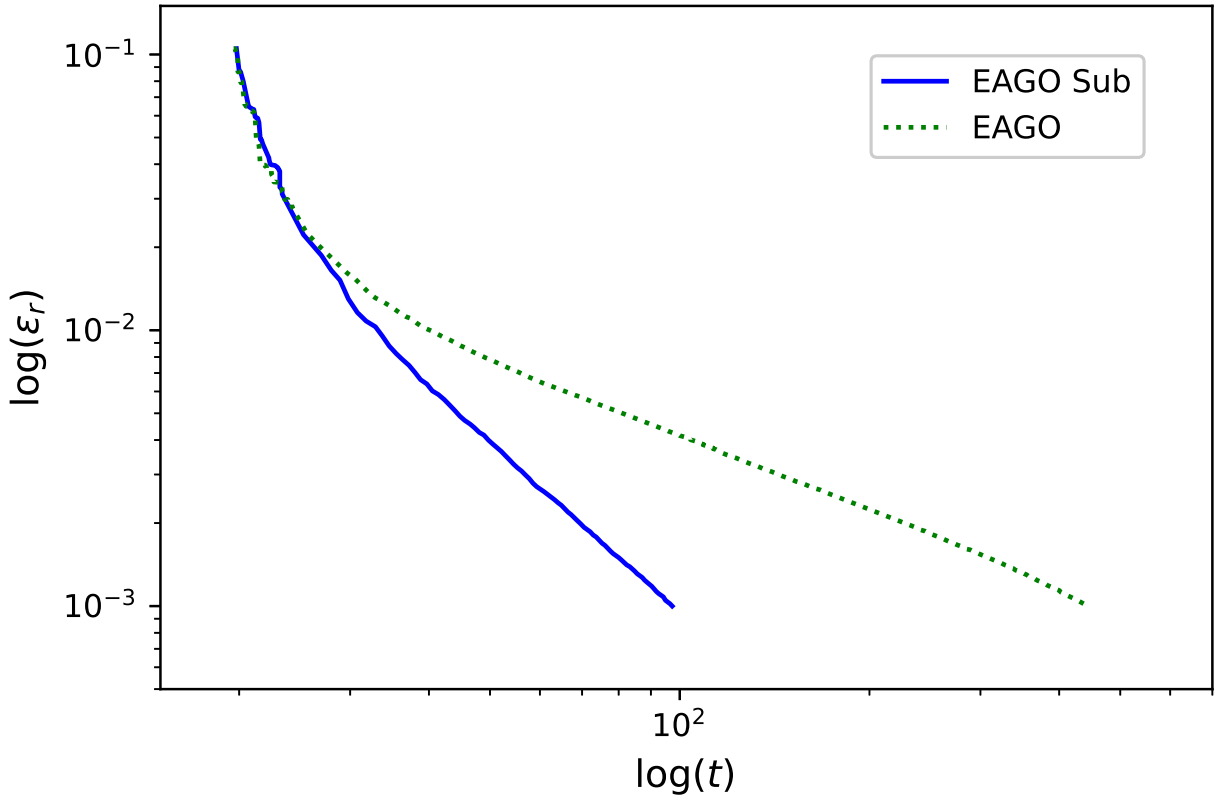

FIGURE 5.6.1: A log-log plot of relative gap remaining $\epsilon_r = (UBD - LBD)/\max(UBD, LBD)$ by solver configuration at time, t. **EAGO Sub** accelerates the rate convergence by a factor of 4.76.

for computing subgradient information was also detailed herein. Three distinct methods by which this new result may be used within a generalized McCormick relaxation framework were also described: an enumerative approach with McCormick relaxations, the use of intermediate affine relaxations define by subgradient expansions and, the use of affine relaxations defined by an affine arithmetic. A pair of simple examples were presented which illustrate how each method may lead to improved relaxations. Lastly, each method was incorporated into a version of the EAGO global optimizer and results from a small test set drawn from the global benchmark library were shown. In this benchmark set, the subgradient expansion reduced the computational time by a factor of three compared to the standard approach and each other method presented herein.

Chapter 6

Global Optimization of Stiff Dynamical Systems

6.1 Introduction

In this chapter, a deterministic global optimization method is presented, that is of special interest for nonlinear programming formulations constrained by stiff systems of ordinary differential equation (ODE) initial value problems (IVPs). The examples arise from dynamic optimization problems exhibiting both fast and slow transient phenomena commonly encountered in model-based systems engineering applications. The proposed approach utilizes unconditionally-stable implicit integration methods to reformulate the ODE-constrained problem into a nonconvex nonlinear program (NLP) with implicit functions embedded. This problem is then solved to global optimality in finite time using a spatial B&B framework utilizing convex/concave relaxations of implicit functions constructed by a method which fully exploits problem sparsity. The algorithms were implemented in the Julia programming language within

the EAGO.jl package and demonstrated on five illustrative examples with varying complexity relevant in process systems engineering. The developed methods enable the guaranteed global solution of dynamic optimization problems with stiff ODE-IVPs embedded.

Dynamic optimization problems of the form:

$$\begin{aligned} \phi^* = & \min_{\mathbf{p}\in P\subset\mathbb{R}^{np}} \phi(\mathbf{x}(\mathbf{p},t_f),\mathbf{p}) \\ \text{s.t. } & \dot{\mathbf{x}}(\mathbf{p},t) = \mathbf{f}(\mathbf{x}(\mathbf{p},t),\mathbf{p},t), \quad \forall t\in I=[t_0,t_f] \\ & \mathbf{x}(\mathbf{p},t_0) = \mathbf{x}_0(\mathbf{p}) \\ & \mathbf{g}(\mathbf{x}(\mathbf{p},t_f),\mathbf{p}) \le \mathbf{0} \end{aligned} \tag{6.1.1}$$

are of extreme importance to process systems engineers and the broader model-based systems engineering community as they can be formulated for a variety of systems whose transient behavior is of particular interest, from optimal control to mechanistic model validation. The first major complicating detail of the optimization formulation (6.1.1) is that it is constrained by a system of ODE-IVPs. Therefore, simply verifying a feasible point requires the solution of a system of ODE-IVPs. The second major complicating detail is that (6.1.1) is a nonconvex program, in general, and therefore verifying optimality requires deterministic global optimization. The focus of this chapter is on solving (6.1.1) to guaranteed global optimality (or declaration of infeasibility). The methods developed in this work are of specific importance when the ODE-IVP system is stiff.

Methods for solving (6.1.1) rigorously to global optimality rely on the spatial branch-and-bound (B&B) framework [99, 136] or some variant. The B&B algorithm requires the ability to calculate rigorous upper and lower bounds on the global optimal solution value. An upper bound can be calculated by simply evaluating $\phi(\mathbf{x}(\cdot\,,t_f),\cdot\,)$ at any feasible point. However,

calculating rigorous lower bounds poses significant challenges as this step requires that rigorous and accurate global bounds are known or are readily calculable for all variables and functions of (6.1.1). For standard nonconvex NLPs (i.e., without dynamical systems constraints), rigorous lower bounds on the optimal solution value are obtained by calculating convex and concave relaxations of the functions and solving a corresponding convex lower-bounding problem. Applying this approach to a dynamic optimization problem (6.1.1) requires rigorous bounds and relaxations of the solution of the parametric ODE-IVPs $\mathbf{x}(\mathbf{p}, t)$ are calculable which are valid for all parameter values $\mathbf{p} \in P$ at every instance in time $t \in I$. Bounding solutions of parametric ODE-IVPs is still an open and active area of research.

The first rigorous methods for solving (6.1.1) to global optimality were introduced by Papamichail and Adjiman [229] which utilized the αBB convex relaxations[3, 4], and by Singer and Barton [282, 283] utilizing McCormick-based relaxations [177]. These approaches summarily utilized auxiliary ODE-IVP systems based on differential inequalities whose solutions were theoretically guaranteed to provide rigorous bounds on the set of parametric solutions (i.e., the *reachable set*) of the original ODE-IVP system. This *relax-then-discretize* approach has been the basis of much of the advances that followed [269, 270, 272, 276, 278, 284, 287] which have focused heavily on advancing theory for improving the tightness and computational efficiency of calculating bounds and relaxations on solutions of parametric ODE-IVPs. In the differential inequalities approach, implicit integration routines may be employed to solve the auxiliary ODE-IVP system using either in-house (e.g., GDOC [286]) or state-of-the-art software packages (e.g., CVODES [274]) [272, 283, 287]. Implicit integration approaches have typically been chosen as the bounding ODE-IVP systems may themselves be stiff and stepsizes may be chosen in an adaptive fashion to achieve the specified accuracy for these auxiliary equations. In the case of CVODES, variable order Adams-Moulton or backward difference formula (BDF) methods are used by default.

Within the past 10 years, other theoretical developments have been made which enable the calculation of rigorous bounds and relaxations of the reachable set using an alternative approach referred to as *discretize-then-relax*. This method makes use of a two-step bounding approach to construct relaxations utilizing an explicit integration scheme. First, valid interval bounds over each timestep are determined via the application of validated interval Taylor models. In a second step, relaxations at specific pointwise-in-time values are refined using interval bounds tightening based on McCormick relaxations [255, 256, 257, 258]. In this approach, the fixed-point interval inclusion tests for existence and uniqueness of solutions limit stepsizes which can be used. The ultimate result is a method which provides valid bounds for the entire exact parametric solution set of the ODE-IVP and at every pointwise-in-time value queried.

An alternative approach to the relax-then-discretize and discretize-then-relax methods discussed above is the calculation of bounds and relaxations of discrete-time approximations. Minimal work has been done on constructing rigorous bounds and relaxations of numerical solutions of parametric ODE-IVPs. This strategy was first demonstrated using McCormick-based relaxations on a dynamic kinetic parameter estimation problem discretized using the explicit (forward) Euler scheme [191]. More recently, a discrete-time differential inequalities approach was developed utilizing the explicit Euler scheme [335]. However, stiff systems present significant computational and numerical challenges to explicit integration methods as stepsizes need to be dramatically reduced to avoid spurious oscillations in the solution trajectories, which in turn dramatically increases the total computational cost of the numerical integration procedure or causes it to fail as stepsizes become infinitesimally small. Recently, a theory for calculating relaxations of implicit functions was developed [300] which enabled the calculation of rigorous relaxations of solutions of parametric linear and nonlinear algebraic systems of equations. These methods were demonstrated for the first time by solving the dynamic kinetic parameter estimation problem of Singer [283] and Mitsos et al. [191] to global optimality using the implicit (backward)

Euler integration scheme [300].

In this work, a new method for rigorously solving nonconvex optimization problems with stiff ODE-IVP constraints is presented, which extends the initial work of Stuber et al. [300] to more accurate higher-order numerically-stable implicit integration schemes. The proposed approach reformulates the dynamic optimization problem (6.1.1) into a nonconvex NLP with equality constraints that are constructed by applying a numerically-stable implicit integration scheme to the ODE-IVP system. Our approach differs from discretizing the dynamic problem (6.1.1) using a chosen integration scheme and solving the resulting NLP via a global optimizer. In the latter case, many of the resulting NLPs will contain hundreds if not hundreds of thousands of nonlinear equality constraints with multiple nonlinear terms. Moreover, the discretized problem must include all the discrete state variables in the decision space.

Due to the curse-of-dimensionality, even the best-in-class commercial global optimizers (e.g. BARON [254], ANTIGONE [185]) have difficulty solving such high-dimensional problems. In our *reduced-space* approach, equality constraints are subsequently eliminated from the final formulation by utilizing the method of Stuber et al. [300] and embedding within the objective and inequality constraint functions the state variables as an implicit function of the parameters. This implicit function is the numerical approximation of the exact parametric solution of the ODE-IVP system. Further, the theory of bounds and relaxations of implicit functions [300] is employed within a B&B framework to solve the NLP with implicit functions embedded to guaranteed global optimality. No assumption is made about the existence of an explicit closed-form solution of the ODE-IVP system and therefore this approach can be applied to arbitrarily complex nonlinear models.

This chapter is arranged as follows. In the next section, the mathematical preliminaries and background are introduced. In the subsequent section, the optimization problem will be

formulated, which is followed by the presentation of algorithms for calculating relaxations of implicit functions that are the numerical approximations of parametric solutions of stiff ODE-IVPs. Lastly, the proposed approach is demonstrated on five relevant numerical examples, which is followed by a discussion and conclusion section.

6.2 Dynamic Optimization Formulation

In this section, we formalize the dynamic optimization problem within the context of the implicit function approach detailed in this work. The dynamic optimization problem (6.1.1) is generalized as:

$$\begin{aligned}
\phi^* = \min_{\mathbf{p}\in P\subset\mathbb{R}^{n_p}} \; & \phi(\mathbf{x}(\mathbf{p},\tau_0),\mathbf{x}(\mathbf{p},\tau_1),\ldots,\mathbf{x}(\mathbf{p},\tau_q),\ldots,\mathbf{x}(\mathbf{p},\tau_Q),\mathbf{p}) \\
\text{s.t.} \; & \dot{\mathbf{x}}(\mathbf{p},t) = \mathbf{f}(\mathbf{x}(\mathbf{p},t),\mathbf{p},t), \quad \forall t\in I=[t_0,t_f] \\
& \mathbf{x}(\mathbf{p},t_0) = \mathbf{x}_0(\mathbf{p}) \\
& \mathbf{g}(\mathbf{x}(\mathbf{p},\tau_0),\mathbf{x}(\mathbf{p},\tau_1),\ldots,\mathbf{x}(\mathbf{p},\tau_q),\ldots,\mathbf{x}(\mathbf{p},\tau_Q),\mathbf{p}) \leq \mathbf{0}
\end{aligned} \tag{6.2.1}$$

where $\phi : D\times D\times\cdots\times D\times\Pi\to\mathbb{R}$ and $\mathbf{g} : D\times\cdots\times D\times\cdots\times D\times\Pi\to\mathbb{R}^{n_g}$ are continuously differentiable on their domains. The formulation (6.2.1) is the most practical general dynamic optimization formulation which explicitly accounts for the objective function and inequality constraints having dependence on specific discrete time points τ_q with $q\in\{0,1,\ldots,Q\}$. Note, the objective function and constraints don't necessarily need to reference the same discrete time points t_q but are represented here as such for notational convenience and simplicity.

We consider discretizing the time domain I into $K=(t_f-t_0)/\Delta t$ timesteps, where $\Delta t>0$ is the stepsize. The differential constraint is then discretized at discrete time points t_k with

$k \in \{0, 1, \ldots, K\}$ where $t_K = t_f$, which are potentially distinct from the discrete time points τ_q. This can be viewed as a generalization of (6.1.1) to a larger number of discrete points. We assume that a factorable mapping $\kappa : \mathbb{R}^{n_x K} \to \mathbb{R}^{n_x Q}$ exists which computes the elements of $\{\mathbf{x}(\mathbf{p}, \tau_q)\}_{q=1}^{Q}$ from the elements of $\{\mathbf{x}(\mathbf{p}, t_k)\}_{k=1}^{K}$. In many of the cases subsequently addressed, an injective mapping of elements of $\{\mathbf{x}(\mathbf{p}, \tau_q)\}_{q=1}^{Q}$ into $\{\mathbf{x}(\mathbf{p}, t_k)\}_{k=1}^{K}$ exists. That is to say, the discretization points τ_q used to evaluate the objective and inequality constraint functions are simply a subset of the discretization points t_k used to approximate the continuous-time system as a discrete-time system.

When disparate indexes are desired for discretization and interpolation is used to compute the remaining state variables. Linear interpolation may be written as a linear combination of two state variables with nonnegative coefficients as shown in (6.2.2). As a result, the interpolated relaxation is subject to order 2 interpolation error and the McCormick relaxation is generally nonexpansive. For $t_{k-1} \leq \tau_q \leq t_k$, the interpolated state variable is given by:

$$\begin{aligned} \lambda &:= \frac{\tau_q - t_{k-1}}{t_k - t_{k-1}} \\ \mathbf{x}(\mathbf{p}, \tau_q) &:= \lambda \mathbf{x}(\mathbf{p}, t_k) + (1 - \lambda)\mathbf{x}(\mathbf{p}, t_{k-1}). \end{aligned} \tag{6.2.2}$$

Note that composite relaxation of $\mathbf{g}$ and ϕ can be defined to contain any factorable interpolation function. As such, we can assume without loss of generality that the problem may be formulated with respect to only t_k discrete time points.

Discretizing the system of ODE-IVPs and applying an implicit integration scheme effectively reformulates the system of n_x ODE-IVPs into a system of $n_x \times K$ (nonlinear) algebraic equations. Therefore, (6.2.1) is reformulated from a dynamic optimization problem to a standard

(nonconvex) NLP as:

$$\begin{aligned} \phi^* = & \min_{(\hat{\mathbf{z}},\mathbf{p})\in Z\times P} \phi(\hat{\mathbf{z}},\mathbf{p}) \\ \text{s.t. } & \mathbf{h}(\hat{\mathbf{z}},\mathbf{p}) = \mathbf{0} \\ & \hat{\mathbf{z}}_0 = \mathbf{x}_0(\mathbf{p}) \\ & \mathbf{g}(\hat{\mathbf{z}},\mathbf{p}) \leq \mathbf{0} \end{aligned} \tag{6.2.3}$$

where $\hat{\mathbf{z}} \in Z \in \mathbb{I}D^{k+1}$ with $\hat{\mathbf{z}} = (\hat{\mathbf{z}}_0, \hat{\mathbf{z}}_1, \ldots, \hat{\mathbf{z}}_K)$ as the vector of state variables for each discrete time point $t_k \in \{0, \ldots, K\}$ with $\hat{\mathbf{z}}_0$ specified by the initial condition $\mathbf{x}_0(\mathbf{p})$, and $\mathbf{h} : D \times \cdots \times D \times P \to \mathbb{R}^{n_x K}$ is the system of (nonlinear) algebraic equations formed by applying the implicit integration scheme. Note, the exact procedure (i.e., integration scheme) employed to formulate the discretized system $\mathbf{h}$ determines the accuracy of the numerical solution of the system of ODE-IVPs versus the *true* solution.

Assumption 6.2.1. There exists a unique implicit function $\mathbf{z}_k : P \to D$ for $k = 0, 1, \ldots, K$ such that $\mathbf{h}(\mathbf{z}(\mathbf{p}), \mathbf{p}) = \mathbf{0}$ holds for all $\mathbf{p} \in P$ with $\mathbf{z} = (\mathbf{x}_0, \mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_K)$.

No discretization scheme can ensure that the above assumption holds for all possible ODE-IVPs. Failure of a traditional implicit integrator may occur around singular points. Most integrators make an adaptive choice of stepsizes to ensure that the resulting system of equations is nonsingular. If a stepsize is selected that is near machine precision, the integrator will typically throw an error. We will present sufficient conditions to ensure that no singular system is contained in the domain of interest in the Relaxation Algorithm section. By making use of Assumption

6.2.1, we can reformulate the discretized problem (6.2.3) into the implicit form as:

$$\begin{aligned} \phi^* &= \min_{\mathbf{p}\in P} \phi(\mathbf{z}(\mathbf{p}), \mathbf{p}) \\ \text{s.t.} \quad & \mathbf{g}(\mathbf{z}(\mathbf{p}), \mathbf{p}) \leq \mathbf{0}. \end{aligned} \tag{6.2.4}$$

It is worth noting that the full-space equality-constrained formulation (6.2.3) has $n_x \times (K+1) + n_p$ decision variables, whereas the reduced-space formulation (6.2.4) has just n_p. This reduction in dimensionality does come at the cost of increased computational complexity associated with the calculation of bounds and relaxations of implicit functions[300]. However, the benefit of this approach is the dramatically reduced number of timesteps needed to evaluate numerically (with potentially improved accuracy) the solution of the stiff ODE-IVP system over explicit integration approaches requiring very small stepsizes for maintaining numerical stability.

It is worth mentioning that the reformulation (6.2.3) is very closely related to that of the collocation approaches[38, 308]. However, since this work is motivated by solving (6.1.1) to guaranteed global optimality, the focus moving forward is on providing rigorous bounds on the states (and the functions that are composed with them), which will be used by the B&B algorithm for deterministic search. Thus, the implicit formulation (6.2.4) is presented, which is also referred to as a "feasible-path method" since the implicit function $\mathbf{z}(\mathbf{p})$ (i.e., the discrete-time solution of the ODE-IVPs) is a feasible point with respect to the equality constraints $\mathbf{h}$ evaluated at $\mathbf{p} \in P$.

6.3 Relaxation Algorithm

In this section, the methods are developed for calculating convex and concave relaxations of implicit functions that are numerical solutions of parametric ODE-IVPs (2.4.2). The

conceptualization of these relaxations is illustrated in Figure 6.3.1. Specifically, in this section we develop methods for relaxing implicit functions via the relaxation of second-order implicit numerical integration schemes.

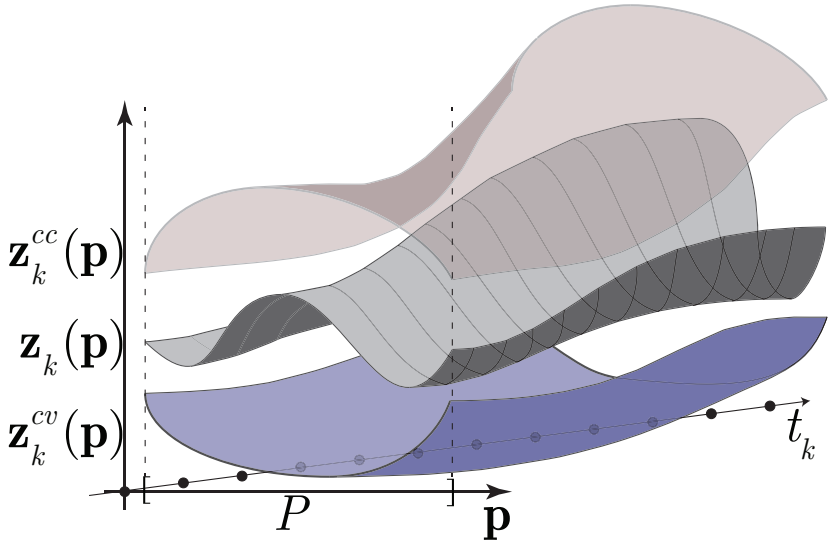

FIGURE 6.3.1: The implicit function $\mathbf{z}_k(\mathbf{p})$ is the approximation of the parametric solution to the initial value problem: $\dot{\mathbf{x}}(\mathbf{p},t) = \mathbf{f}(\mathbf{x}(\mathbf{p},t),\mathbf{p},t)$ at the discrete time points t_k illustrated by the curve segments on the center surface. For s-step methods, each $\mathbf{z}_k(\mathbf{p})$ approximates the actual solution $\mathbf{x}(\mathbf{p},t_k)$ with $O(\Delta t^{s+1})$ error, for each k. The center surface is plotted using the standard approach by connecting each $\mathbf{z}_k(\mathbf{p})$ using affine interpolation between adjacent time nodes (black dots) for each $\mathbf{p} \in P$. Convex and concave relaxations of $\mathbf{z}_k$ on P are illustrated as $\mathbf{z}_k^{cv}$ and $\mathbf{z}_k^{cc}$, respectively. Similarly, these surfaces are plotted using affine interpolation of each discrete-time relaxation of $\mathbf{z}_k$ on P between adjacent time nodes.

In order to relax solutions of an IVP with high accuracy, multiple different relaxation approaches must be used concurrently. This is because relaxations of the state variables $\mathbf{z}_k$ at the k^{th} timestep depend on the relaxations of the state variables at the previous s timesteps for an s-step (or s-order for the methods considered herein) parametric implicit linear multistep (PILMS) method. This is implemented as follows. The state relaxations of $\mathbf{z}_1$ are determined by solving the (nonlinear) algebraic equations formed by an implicit Euler integration step in which the relaxations of the $\mathbf{z}_0$ states take the values of the initial condition $\mathbf{x}_0$ (if the initial conditions do not depend on the parameters $\mathbf{p}$), or they are set equal to the respective relaxations of $\mathbf{x}_0(\mathbf{p})$ on P

(known explicitly in a closed form). For the $\mathbf{z}_2$ state relaxations, a second-order implicit method is used with the relaxations of $\mathbf{z}_0$ given by initial conditions and the $\mathbf{z}_1$ relaxations determined in the prior step. This continues through to $\mathbf{z}_s$ after which the defined s-step PILMS method can be used directly. Any subsequent timestep $k > s$ then makes use of the relaxations of the prior s states. The analog with real vector-valued PILMS integration schemes is that the s-step implicit integration approach requires the solution of an n_x-dimensional system of algebraic equations at each timestep. However, for timesteps $k > s$, we utilize the values calculated for the previous s states. Thus, the higher-order implicit integration schemes preserve the "sequential block-solve" property that only a system of n_x equations needs to be solved simultaneously at each timestep, while also improving accuracy by utilizing previously calculated information.

Note that this method relies on the *a priori* determination of stepsizes which must be the same for all values of $\mathbf{p} \in P$. As such, choosing the largest stepsize (smallest K) may itself be an NP-hard problem and the development of an adaptive stepsize selection routine for all values of $\mathbf{p} \in P$ is left for future research. This contrasts the differential inequalities approach in which state-of-the-art integration schemes can be used that determine stepsizes in an adaptive fashion as a non-parametric ODE-IVP system bounds the solution set for each box $P^l \subset P$.

6.3.1 Numerical Integration of Parametric ODE-IVPs

After obtaining $\hat{\mathbf{z}}_k$ for $k = 1, \ldots, s-1$, an s-step PILMS method can be defined generally by the following formula:

$$\hat{\mathbf{z}}_{k+s} + \sum_{i=0}^{s-1} a_i \hat{\mathbf{z}}_{k+i} = \Delta t \sum_{j=0}^{s} b_j \mathbf{f}(\hat{\mathbf{z}}_{k+j}, \mathbf{p}, t_{k+j}). \tag{6.3.1}$$

where the index $k + s$ is the current timestep we're calculating the states with respect to. For a fixed value of $\mathbf{p}$, this discretization reduces to its non-parametric version. We will consider two

generally applicable PILMS methods: the parametric Adams-Moulton (AM) methods and the backward-difference formula (BDF) methods. For Adams-Moulton methods, $a_{s-1} = -1$, and $a_{s-2} = \cdots = a_0 = 0$ and the b_j coefficients are chosen such that the s-step method has order s. In contrast, the BDF methods [70] set $b_j = 0$ for every j and determine the remaining coefficients to achieve an order of s. We can then write the s-step parametric Adams-Moulton method as the following residual:

$$\boldsymbol{\zeta}_k^s(\hat{\mathbf{z}}_{k+s}, \ldots, \hat{\mathbf{z}}_k, \mathbf{p}) = \hat{\mathbf{z}}_{k+s} - \hat{\mathbf{z}}_{k+s-1} - \Delta t \sum_{j=0}^{s} b_j \mathbf{f}(\hat{\mathbf{z}}_{k+j}, \mathbf{p}, t_{k+j}) = \mathbf{0} \tag{6.3.2}$$

and we can write the s-step parametric BDF method as the following residual:

$$\boldsymbol{\xi}_k^s(\hat{\mathbf{z}}_{k+s}, \ldots, \hat{\mathbf{z}}_k, \mathbf{p}) = \hat{\mathbf{z}}_{k+s} + \sum_{i=0}^{s-1} a_i \hat{\mathbf{z}}_{k+i} - \Delta t b_s \mathbf{f}(\hat{\mathbf{z}}_{k+s}, \mathbf{p}, t_{k+s}) = \mathbf{0} \tag{6.3.3}$$

where $\boldsymbol{\zeta}_k^s, \boldsymbol{\xi}_k^s : D \times \cdots \times D \times P \to D$. Note that in the above equations (6.3.2) and (6.3.3), the values $\hat{\mathbf{z}}_{k+i}$ are known for $i = 0, \ldots, s-1$. Therefore, (6.3.2) and (6.3.3) form a system of n_x algebraic equations with n_x unknowns. In order to solve each of these algebraic systems, the Jacobian matrices of (6.3.2) and (6.3.3) with respect to the $\hat{\mathbf{z}}_{k+s}$ variables are required. The respective Jacobian matrix of either formulation is given below by:

$$\mathbf{J}_k^s(\hat{\mathbf{z}}_{k+s}, \mathbf{p}) = \mathbf{I}_{n_x} - \Delta t b_s \mathbf{J}_{\mathbf{x}}(\hat{\mathbf{z}}_{k+s}, \mathbf{p}) \tag{6.3.4}$$

where $\mathbf{I}_{n_x} \in \mathbb{R}^{n_x \times n_x}$ is the identity matrix, $\mathbf{J}_{\mathbf{x}}$ is the Jacobian of the right-hand side system $\mathbf{f}$ with respect to the state vector, and the b_s values are determined by the method of choice. Note that in either case, this Jacobian depends only on the state variables of the current block $\hat{\mathbf{z}}_{k+s}$ and the parameter values $\mathbf{p}$.

The following theorem details conditions under which a unique implicit function exists

satisfying Assumption 6.2.1.

Theorem 6.3.1. Suppose the system has been discretized using an s-step parametric BDF or AM method. Let $b := \min(s, k)$ and $\boldsymbol{\theta}^b_{k-b}(\hat{\mathbf{z}}_k, \hat{\mathbf{z}}_{k-1}, \ldots, \hat{\mathbf{z}}_{k-b}, \mathbf{p}) = \mathbf{0}$ for $1 \leq k \leq K$ with $\boldsymbol{\theta} \in \{\boldsymbol{\zeta}, \boldsymbol{\xi}\}$. Let $J^v_k(X, P) \in \mathbb{R}^{n_x \times n_x}$ be nonsingular (i.e., contains no singular matrices) with $X \in \mathbb{I}D$ for $1 \leq v \leq s$ computed from (6.3.4). Suppose for $k = 1, \ldots, K$ there exists $(\hat{\mathbf{z}}^*_k, \hat{\mathbf{z}}^*_{k-1}, \ldots, \hat{\mathbf{z}}^*_{k-b}, \mathbf{p}^*) \in X^{b+1} \times P$ such that $\boldsymbol{\theta}^b_{k-b}(\hat{\mathbf{z}}^*_k, \hat{\mathbf{z}}^*_{k-1}, \ldots, \hat{\mathbf{z}}^*_{k-b}, \mathbf{p}^*) = \mathbf{0}$. Further, suppose that for every $\hat{\mathbf{z}}_k \in X$ such that $\hat{\mathbf{z}}_k \notin \mathit{int}(X)$, we have $\boldsymbol{\theta}^b_{k-b}(\hat{\mathbf{z}}_k, \hat{\mathbf{z}}_{k-1} \ldots, \hat{\mathbf{z}}_{k-b}, \mathbf{p}) \neq \mathbf{0}$ for every $(\hat{\mathbf{z}}_{k-1} \ldots, \hat{\mathbf{z}}_{k-b}, \mathbf{p}) \in X^b \times P$. Then Assumption 6.2.1 holds with $\mathbf{h}_k(\mathbf{z}_k(\mathbf{p}), \mathbf{p}) = \boldsymbol{\theta}^b_{k-b}(\mathbf{z}_k(\mathbf{p}), \mathbf{z}_{k-1}(\mathbf{p}), \ldots, \mathbf{z}_{k-b}(\mathbf{p}), \mathbf{p}) = \mathbf{0}$ for $1 \leq k \leq K$.

Proof. This result follows directly from the semilocal implicit function theorem (Thm. 5.1.3 of Neumaier[219]) applied sequentially to each block of equations $k \in \{1, \ldots, K\}$. We proceed by strong induction. For $k = 1$, note that $\mathbf{z}_0$ is a function of $\mathbf{p}$, namely $\mathbf{z}_0 = \mathbf{x}_0(\mathbf{p})$, therefore $\boldsymbol{\theta}^1_0(\hat{\mathbf{z}}_1, \hat{\mathbf{z}}_0, \mathbf{p}) = \boldsymbol{\theta}^1_0(\hat{\mathbf{z}}_1, \mathbf{x}_0(\mathbf{p}), \mathbf{p})$. Continuous differentiability of $\boldsymbol{\xi}^1_0$ with respect to $\hat{\mathbf{z}}_1 \in X$ is ensured directly by the continuous differentiability of $\mathbf{f}$ over $D \times \Pi \times T$ and Corollary 5.1.5 of Neumaier[219] is satisfied. By assumption, there exists $(\hat{\mathbf{z}}^*_1, \hat{\mathbf{z}}^*_0, \mathbf{p}^*) \in X^2 \times P$ such that $\boldsymbol{\theta}^1_0(\hat{\mathbf{z}}^*_1, \hat{\mathbf{z}}^*_0, \mathbf{p}^*) = \mathbf{0}$ which may be restated as $\boldsymbol{\theta}^1_0(\hat{\mathbf{z}}^*_1, \mathbf{x}_0(\mathbf{p}^*), \mathbf{p}^*) = \mathbf{0}$. Further, for every $\hat{\mathbf{z}}_1 \in X$ such that $\hat{\mathbf{z}}_1 \notin \mathrm{int}(X)$, we have $\boldsymbol{\theta}^1_0(\hat{\mathbf{z}}_1, \mathbf{x}_0(\mathbf{p}), \mathbf{p}) \neq \mathbf{0}$ for all $\mathbf{p} \in P$ since $\mathbf{x}_0(P) \subset D$. As such, Theorem 5.1.3 of Neumaier[219] is satisfied with $\mathbf{J}^1_1(X, P)$ ensuring a unique implicit function $\mathbf{z}_1(\mathbf{p}) \in X \in \mathbb{I}D$ exists for every $\mathbf{p} \in P$ satisfying $\mathbf{h}_1(\mathbf{z}_1(\mathbf{p}), \mathbf{p}) = \boldsymbol{\theta}^1_0(\mathbf{z}_1(\mathbf{p}), \mathbf{x}_0(\mathbf{p}), \mathbf{p}) = \mathbf{0}$.

Now suppose that for $1 < k \leq K$, there exist unique implicit functions $\mathbf{z}_1(\mathbf{p}), \mathbf{z}_2(\mathbf{p}), \ldots, \mathbf{z}_{k-1}(\mathbf{p}) \in X \in \mathbb{I}D$ for every $\mathbf{p} \in P$. Then, we have $\boldsymbol{\theta}^b_{k-b}(\hat{\mathbf{z}}_k, \hat{\mathbf{z}}_{k-1}, \ldots, \hat{\mathbf{z}}_{k-b}, \mathbf{p}) = \boldsymbol{\theta}^b_{k-b}(\hat{\mathbf{z}}_k, \mathbf{z}_{k-1}(\mathbf{p}), \ldots, \mathbf{z}_{k-b}(\mathbf{p}), \mathbf{p})$. Continuous differentiability of $\boldsymbol{\theta}^b_{k-b}$ with respect to $\hat{\mathbf{z}}_k \in X$ is ensured directly by the continuous differentiability of $\mathbf{f}$ over $D \times \Pi \times T$ and Corollary 5.1.5 of Neumaier[219] is satisfied. By assumption, there exists $(\hat{\mathbf{z}}^*_k, \ldots, \hat{\mathbf{z}}^*_{k-b}, \mathbf{p}^*) \in X^{b+1} \times P$ such that $\boldsymbol{\theta}^b_{k-b}(\hat{\mathbf{z}}^*_k, \hat{\mathbf{z}}^*_{k-1}, \ldots, \hat{\mathbf{z}}^*_{k-b}, \mathbf{p}^*) = \mathbf{0}$. As a result, there exists

$\mathbf{p}^* \in P$ such that $\boldsymbol{\theta}^b_{k-b}(\hat{\mathbf{z}}^*_k, \mathbf{z}_{k-1}(\mathbf{p}^*), \ldots, \mathbf{z}_{k-b}(\mathbf{p}^*), \mathbf{p}^*) = \mathbf{0}$. Additionally, for each $\hat{\mathbf{z}}_k \in X$ such that $\hat{\mathbf{z}}_k \notin \text{int}(X)$, we have $\boldsymbol{\theta}^b_{k-b}(\hat{\mathbf{z}}_k, \mathbf{z}_{k-1}(\mathbf{p}) \ldots, \mathbf{z}_{k-b}(\mathbf{p}), \mathbf{p}) \neq \mathbf{0}$ for every $\mathbf{p} \in P$ and as such Theorem 5.1.3 of Neumaier[219] is satisfied with $\mathbf{J}^s_k(X, P)$ ensuring existence of a unique implicit function $\mathbf{z}_k(\mathbf{p}) \in X \in \mathbb{I}D$ for every $\mathbf{p} \in P$. This implicit function satisfies $\mathbf{h}_k(\mathbf{z}_k(\mathbf{p}), \mathbf{p}) = \boldsymbol{\theta}^b_{k-b}(\mathbf{z}_k(\mathbf{p}), \mathbf{z}_{k-1}(\mathbf{p}) \ldots, \mathbf{z}_{k-b}(\mathbf{p}), \mathbf{p}) = \mathbf{0}$. This completes the proof by strong induction. As a consequence, there exists a unique implicit function $\mathbf{z}_k : P \to D$ for $k = 0, 1, \ldots, K$ such that $\mathbf{h}(\mathbf{z}(\mathbf{p}), \mathbf{p}) = \mathbf{0}$ holds for all $\mathbf{p} \in P$ with $\mathbf{z} = (\mathbf{x}_0, \mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_K)$. □

In addition to the theorem outlined above, parametric interval iterations (e.g., interval Newton[219], Krawcyzk[154], Hansen-Sengupta[219], etc.) have associated existence and uniqueness conditions which may be computationally verified at each iteration. As such, these methods can serve as a useful preprocessing step which may contract the X interval and furnish a guarantee of existence and uniqueness of an enclosed implicit function solution branch for every $\mathbf{p} \in P$. Further a condition for nonsingularity satisfying Theorem 6.3.1 is given by Theorem 6.3.2 below.

Theorem 6.3.2. Let $Z_{k+s} \in \mathbb{I}X$ and let $\mathbf{J}_c \in \mathbb{R}^{n_x \times n_x}$ be nonsingular defined as $\mathbf{J}_c \equiv mid(J^s_k(Z_{k+s}, P))$ (i.e., the elementwise midpoint of the interval matrix), and $\Delta^* \equiv (J^s_k(Z_{k+s}, P))^U - \mathbf{J}_c$ (i.e., the elementwise radius of J^s_k). Further, let $\mathbf{A} \equiv |\mathbf{J}_c^{-1}|\Delta^*$, where $|\mathbf{J}_c^{-1}|$ is the elementwise absolute value of $\mathbf{J}_c^{-1}$, and let $\lambda_{\max} = \max_i\{|\lambda_i|\}$ be the magnitude of the extremal eigenvalue(s) of $\mathbf{A}$. If $\lambda_{\max} < 1$, then $J^s_k(Z_{k+s}, P)$ contains no singular matrices.

Proof. This directly follows from Proposition 4.1.1 in Neumaier[219]. Note that b_s is a positive constant which depends on the method of choice and $\Delta t > 0$ by definition. □

Remark 6.3.1. Theorem 6.3.2 provides a computational test for aiding users in choosing appropriate discretizations to avoid violating the hypotheses of Theorem 6.3.1 and ensuring

satisfaction of Assumption 6.2.1.

Second-order PILMS methods considered herein (both the trapezoidal method, i.e., two-step Adams-Moulton method, and the two-step BDF) exhibit A-stability (unconditional stability) while first-order methods exhibit absolute A-stability (L-stability) [109, 119]. Higher-order PILMS methods often exhibit better stability than explicit methods but they do not exhibit A-stability [119]. However, these methods exhibit $O(\Delta t^{s+1})$ local truncation error and the superior stability of lower-order methods must be balanced against the superior numerical accuracy of higher-order methods. When used to solve ODE-IVPs, lower-order methods are commonly used for timesteps involving the initial condition and s-order methods are used once $s-1$ preceding points haven been calculated. Since the focus of this work is on the application to stiff systems, numerical stability is emphasized when choosing an appropriate implicit integration form. Therefore, in this chapter we consider only first- and second-order methods which are given by the residual equations:

$$\boldsymbol{\zeta}_k^2(\hat{\mathbf{z}}_{k+2}, \hat{\mathbf{z}}_{k+1}, \hat{\mathbf{z}}_k, \mathbf{p}) = \hat{\mathbf{z}}_{k+2} - \hat{\mathbf{z}}_{k+1} - (\Delta t/2)\left(\mathbf{f}(\hat{\mathbf{z}}_{k+2}, \mathbf{p}, t_{k+2}) + \mathbf{f}(\hat{\mathbf{z}}_{k+1}, t_{k+1})\right) \tag{6.3.5}$$

$$\boldsymbol{\xi}_k^1(\hat{\mathbf{z}}_{k+1}, \hat{\mathbf{z}}_k, \mathbf{p}) = \hat{\mathbf{z}}_{k+1} - \hat{\mathbf{z}}_k - \Delta t\mathbf{f}(\hat{\mathbf{z}}_{k+1}, \mathbf{p}, t_{k+1}) \tag{6.3.6}$$

$$\boldsymbol{\xi}_k^2(\hat{\mathbf{z}}_{k+2}, \hat{\mathbf{z}}_{k+1}, \hat{\mathbf{z}}_k, \mathbf{p}) = \hat{\mathbf{z}}_{k+2} - \tfrac{4}{3}\hat{\mathbf{z}}_{k+1} + \tfrac{1}{3}\hat{\mathbf{z}}_k - \tfrac{2}{3}\Delta t\mathbf{f}(\hat{\mathbf{z}}_{k+2}, \mathbf{p}, t_{k+2}) \tag{6.3.7}$$

where (6.3.5) is the second-order/two-step parametric Adams-Moulton method (i.e., trapezoidal method), (6.3.6) is the first-order parametric BDF method (i.e., backward/implicit Euler), and (6.3.7) is the second-order/two-step parametric BDF method. With respect to the equality constraints of (6.2.3), the equations given by (6.3.5)-(6.3.7) would form the K blocks of n_x

equations of $\mathbf{h}$. This is expressed formally as:

$$\mathbf{h}(\hat{\mathbf{z}}_0, \hat{\mathbf{z}}_1, \ldots, \hat{\mathbf{z}}_K, \mathbf{p}) = \begin{pmatrix} \boldsymbol{\xi}_0^1(\hat{\mathbf{z}}_1, \hat{\mathbf{z}}_0, \mathbf{p}) \\ \boldsymbol{\theta}_0^2(\hat{\mathbf{z}}_2, \hat{\mathbf{z}}_1, \hat{\mathbf{z}}_0, \mathbf{p}) \\ \vdots \\ \boldsymbol{\theta}_{K-2}^2(\hat{\mathbf{z}}_K, \hat{\mathbf{z}}_{K-1}, \hat{\mathbf{z}}_{K-2}, \mathbf{p}) \end{pmatrix} = \mathbf{0} \tag{6.3.8}$$

where $\boldsymbol{\theta} \in \{\boldsymbol{\xi}, \boldsymbol{\zeta}\}$. The reader is reminded that if the parameters $\mathbf{p}$ are specified, the number of unknown variables is $n_x K$ since $\hat{\mathbf{z}}_0$ is fully-determined by the initial conditions. From a numerical algebraic equation-solving perspective, one would not solve $\mathbf{h} = \mathbf{0}$ (i.e., the full $n_x K$-dimensional system) simultaneously, but instead in a sequential block-solve fashion where the n_x-dimensional system formed by the equations h_1 through h_{n_x} are solved simultaneously followed by equations h_{n_x+1} through h_{2n_x}, continuing sequentially all the way to the K^{th} system of equations $h_{n_x(K-1)+1}$ through $h_{n_x K}$.

The computational performance benefit of the sequential block-solve approach is clear as the full-scale (dense) Newton-Raphson with Gauss-Seidel algorithm exhibits $O(K^2 n_x^2 \kappa_{\text{GS}} \kappa_{\text{NR}})$ time complexity with κ_{GS} representing the number of Gauss-Seidel iterations and κ_{NR} the number of Newton-Raphson iterations required for convergence. In the worst-case, $\kappa_{\text{GS}} = K n_x$, and so a dense solve would have $O(K^3 n_x^3 \kappa_{\text{NR}})$ time complexity. In contrast, a conventional banded solver would exhibit $O(K n_x^2 \kappa_{\text{GS}} \kappa_{\text{NR}}) = O(K^2 n_x^3 \kappa_{\text{NR}})$ time complexity (for strongly-coupled right-hand side functions). The sequential-block approach outlined here exhibits $O(K n_x^2 \kappa_{\text{GS}} \kappa_{\text{NR}})$ time complexity, except $\kappa_{GS} = n_x$ in the worst-case. Thus, the sequential-block approach exhibits $O(K n_x^3 \kappa_{\text{NR}})$ worst-case time complexity. Example sparsity patterns of the occurrence matrices

corresponding to the systems formed for each method with $n_x = 5$ are shown in Figure 6.3.2. The corresponding directed graph for the considered two-step methods is shown in Figure 6.3.3.

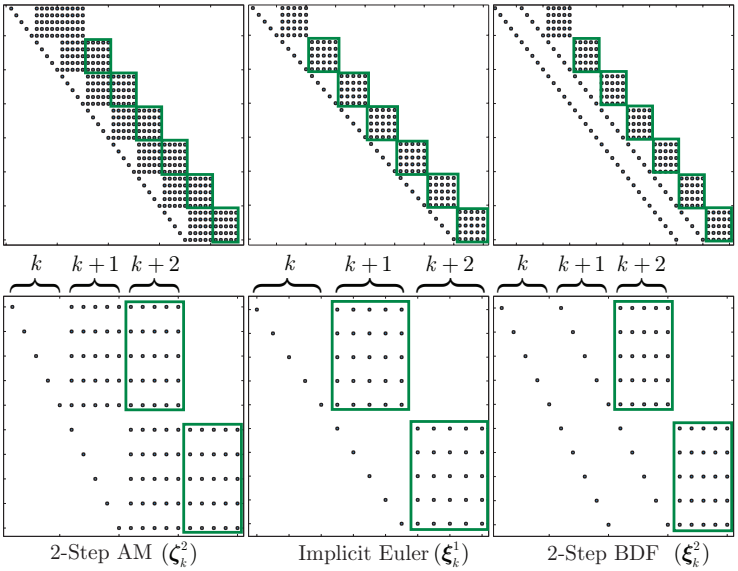

FIGURE 6.3.2: The sparsity patterns of the occurrence matrices of the systems of equations are illustrated for each of the three implicit integration schemes with $n_x = 5$. The sparsity patterns exhibit the block-diagonal structure corresponding to the timestep k which is exploited by the numerical equation solver and relaxation algorithms.

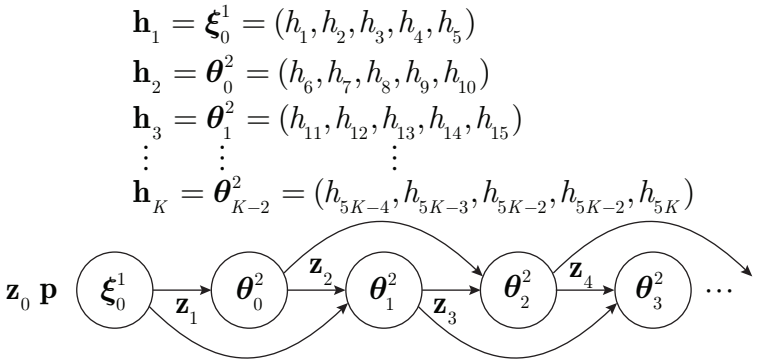

FIGURE 6.3.3: The directed graph corresponding to the occurrence matrices in Fig. 6.3.2 for the considered 2-step PILMS methods (with $n_x = 5$, $\boldsymbol{\theta} \in \{\boldsymbol{\xi}, \boldsymbol{\zeta}\}$) is depicted illustrating the sequential-block structure exploited when solving numerically the discretized system and constructing bounds and relaxations of the implicit functions $\mathbf{z}_k$ on P.

6.3.2 Relaxations of Parametric Implicit Linear Multistep Methods

In this section, we present the notation and results for constructing relaxations of implicit functions via a sequential-block procedure ideal for implicit integration schemes (6.3.5)-(6.3.7), which will also be extensible to general s-step PILMS methods as in (6.3.1). The first construction of relaxations and subgradients of implicit functions via an implicit integration form was demonstrated in Stuber et al.[300]. However, in that work, only the implicit (backward) Euler scheme (6.3.6) was presented, and higher-order methods such as (6.3.5) and (6.3.7) were not considered. Further, although that work constructed relaxations in a sequential-block procedure, the notation was not generalized beyond the newly-developed notation for the standard relaxations of implicit functions. We begin with the following assumption, which is analogous to Assumption 3.14 in Stuber et al.[300] for ensuring that relaxations of implicit functions are computable.

Assumption 6.3.2. We assume that the following holds:

1. There exists a function $\mathbf{z} : P \to D^{K+1}$ with $\mathbf{h}(\mathbf{z}(\mathbf{p}), \mathbf{p}) = \mathbf{0}$, and an interval $X \in \mathbb{I}D$ such that $\mathbf{z}(P) \subset X^{K+1}$ and $\mathbf{z}(\mathbf{p})$ is unique in X^{K+1} for all $\mathbf{p} \in P$.

2. Derivative information $\nabla h_i, \quad i = 1, \ldots, n_x K$ is available and is factorable, say by automatic differentiation.

3. A matrix $\mathbf{Y}_k \in \mathbb{R}^{n_x \times n_x}$ is known such that $M_k = \mathbf{Y} J_k^s(X, P)$ satisfies $0 \notin M_{k,ii}$ for all $i \in \{1, \ldots, n_x\}$ and for all k, where $J_k^s(X, P)$ is an inclusion monotonic interval extension of $\mathbf{J}_k^s$ (given by (6.3.4)) on $X \times P$.

Note that for consistency with (6.2.3), Assumption 6.2.1, and (6.3.8), we have defined $\mathbf{z}$ to have $n_x(K + 1)$ dimensionality to account for the initial condition $\mathbf{z}_0(\mathbf{p}) = \mathbf{x}_0(\mathbf{p})$. Theorems 6.3.1 and 6.3.2 together provide a method for determining an appropriate $X \in \mathbb{I}D$ satisfying

Assumption 6.3.2.1. Parametric interval methods may be used to automatically compute and verify an appropriate X interval satisfying Assumption 6.3.2.1 utilizing Theorems 6.3.1 and 6.3.2. In general, domain specific knowledge is required to furnish a valid initial guess for the interval X such that Assumption 6.3.2.1 holds. The difficulty of this will vary between applications. In some cases, for example, it may be sufficient to recognize that mass fractions must be nonnegative and less than one. In other cases, more specialized knowledge of the system may be required.

To develop the relaxations of the parametric state trajectories, we rely on the construction of relaxations of parametric solutions of nonlinear algebraic systems formed by the PILMS methods. Therefore, the core theory relies on the construction of composite relaxations of fixed-point iterations in a special way such that the computed relaxations are not only valid for the numerical approximations of parametric solutions of nonlinear algebraic systems, but are valid for the *true* solutions themselves. In this work, we will utilize an analog to the Newton-Raphson fixed-point iteration with Gauss-Seidel for constructing valid relaxations of implicit function solutions of stiff ODE-IVPs approximated using two-step PILMS methods. This approach is very closely related to the Hansen-Sengupta interval method [219] for constructing interval bounds of solutions of nonlinear algebraic systems. The development of this approach starts with the parametric mean value theorem [300].

The parametric mean value theorem applied to the j^{th} dimension of the full system of equations results in the following equation:

$$\nabla_{\mathbf{x}} h_j(\mathbf{y}^j(\mathbf{p}), \mathbf{p})^{\mathrm{T}}(\mathbf{z}_k(\mathbf{p}) - \boldsymbol{\gamma}(\mathbf{p})) = -h_j(\boldsymbol{\gamma}(\mathbf{p}), \mathbf{p}),\ \ j = (k-1)n_x + 1, \ldots, kn_x \tag{6.3.9}$$

where $\boldsymbol{\gamma} : P \to D$ and $\mathbf{y}^j : P \to D$ satisfies $\mathbf{y}^j(\mathbf{p}) = \lambda(\mathbf{p})\mathbf{z}_k(\mathbf{p}) + (1 - \lambda(\mathbf{p}))\boldsymbol{\gamma}(\mathbf{p})$ for all $\mathbf{p} \in P$ for some $\lambda : P \to (0, 1)$. In a somewhat similar way, the construction of relaxations of implicit functions as solutions of nonlinear algebraic systems is computed as composite relaxations of the

system of equations formed by the application of the parametric mean value theorem to $\mathbf{h}_k$. To form this system of equations, we must first define $\mathbf{M}_k$ (and $\mathbf{B}_k$) matrix-valued functions. The $\mathbf{M}_k$ (and $\mathbf{B}_k$) matrix-valued functions exist by differentiability of $\mathbf{h}_k$ and the parametric mean value theorem for parametric multivariate vector-valued functions (see Lemma 3.15 of Stuber et al.[300]).

Definition 6.3.3 ($\mathbf{M}_k, \mathbf{B}_k$)**.** The matrix-valued functions $\mathbf{M}_k : P \to M_k$ and $\mathbf{B}_k : X \times \cdots \times X \times P \to M_k$, with $k \in \{1, \ldots, K\}$ corresponding to each timestep, are defined as

$$\begin{aligned}\mathbf{M}_k(\cdot) &= \mathbf{B}_k(\mathbf{y}^{(k-1)n_x+1}(\cdot), \mathbf{y}^{(k-1)n_x+2}(\cdot), \ldots, \mathbf{y}^{kn_x}(\cdot), \cdot) \\ &\equiv \mathbf{Y}_k \begin{pmatrix} \nabla_{\mathbf{x}} h_{(k-1)n_x+1}(\mathbf{y}^{(k-1)n_x+1}(\cdot), \cdot)^{\mathrm{T}} \\ \nabla_{\mathbf{x}} h_{(k-1)n_x+2}(\mathbf{y}^{(k-1)n_x+2}(\cdot), \cdot)^{\mathrm{T}} \\ \vdots \\ \nabla_{\mathbf{x}} h_{kn_x}(\mathbf{y}^{kn_x}(\cdot), \cdot)^{\mathrm{T}} \end{pmatrix}. \end{aligned} \tag{6.3.10}$$

Using the timestep/block indexing, the vector-valued functions are defined as $\mathbf{h}_1(\hat{\mathbf{z}}, \mathbf{p}) = (h_1, h_2, \ldots, h_{n_x}) = \boldsymbol{\xi}_0^1(\hat{\mathbf{z}}_1, \mathbf{z}_0(\mathbf{p}), \mathbf{p})$ and $\mathbf{h}_k(\hat{\mathbf{z}}_k, \mathbf{p}) = (h_{(k-1)n_x+1}, h_{(k-1)n_x+2}, \ldots, h_{kn_x}) = \boldsymbol{\theta}_{k-2}^2(\hat{\mathbf{z}}_k, \mathbf{z}_{k-1}(\mathbf{p}), \mathbf{z}_{k-2}(\mathbf{p}), \mathbf{p})$ for $k \geq 2$ with $\boldsymbol{\theta} \in \{\boldsymbol{\zeta}, \boldsymbol{\xi}\}$. The parametric functions $\mathbf{y}^j : P \to X$ satisfy the parametric mean value theorem (Cor. 2.5 in Stuber et al.[300]) when applied to $h_j : X \times P \to X$, as illustrated in (6.3.9), with $j = (k-1)n_x + 1, \ldots, kn_x$ for each timestep $k \in \{1, \ldots, K\}$. In other words, $(\mathbf{y}^j(\mathbf{p}), \mathbf{p}) \in X \times P$ are the points at which the gradients $\nabla_{\mathbf{x}} h_j(\cdot, \cdot)$ are evaluated at. Further, the matrix $\mathbf{Y}_k \in \mathbb{R}^{n_x \times n_x}$ is chosen such that it satisfies Assumption 6.3.2.3.

Remark 6.3.4. Note that the definition of $\mathbf{M}_k$ (and $\mathbf{B}_k$) satisfies Lemma 3.15 of Stuber et al.[300].

We include it here for completeness and for easy reference in the following theorem. We note that $\mathbf{M}_k$ cannot be very easily calculated since the $\mathbf{y}^j$ functions are not known. Fortunately, we do not need to calculate $\mathbf{M}_k$, but we must able to calculate relaxations of $\mathbf{M}_k$, which is in fact much easier than calculating $\mathbf{M}_k$ itself.

These matrix-valued functions resemble the Jacobian matrix since they are matrices of partial derivatives of $\mathbf{h}_k$ with respect to the state variables $\mathbf{x}$, for the current timestep approximated as $\hat{\mathbf{z}}_k$. However, it is worth highlighting that these are not actually the Jacobian matrix since each transposed gradient (matrix row) is not evaluated at necessarily the same point, but at the point $\mathbf{y}^j$ from the k^{th} block of equations according to the parametric mean value theorem. From (6.3.9) and Definition 6.3.3, we can now form the n_x-dimensional system of equations for the k^{th} block as:

$$\mathbf{M}_k(\mathbf{p})(\mathbf{z}_k(\mathbf{p}) - \boldsymbol{\gamma}(\mathbf{p})) = -\mathbf{Y}_k\mathbf{h}_k(\boldsymbol{\gamma}(\mathbf{p}), \mathbf{p}) = -\mathbf{Y}_k\boldsymbol{\theta}^2_{k-2}(\boldsymbol{\gamma}(\mathbf{p}), \mathbf{z}_{k-1}(\mathbf{p}), \mathbf{z}_{k-2}(\mathbf{p}), \mathbf{p}),\ \forall \mathbf{p} \in P. \tag{6.3.11}$$

From this mean value form, we can now define the function $\boldsymbol{\psi}_k$ as an analog to the Newton-Raphson fixed-point iteration with Gauss-Seidel for approximating $\mathbf{z}_k$ (i.e., the solution of the nonlinear system). The form of this iteration is defined for the k^{th} timestep of two-step PILMS methods in the following.

Definition 6.3.5 ($\boldsymbol{\psi}_k$)**.** Let $\mathbf{b}_k : X \times X \times X \times P \to \mathbb{R}^{n_x}$ such that $\mathbf{b}_k = \mathbf{Y}_k\boldsymbol{\theta}^s_{k-2}$ with $\boldsymbol{\theta} \in \{\boldsymbol{\xi}, \boldsymbol{\zeta}\}$, $s \in \{1, 2\}$, and $2 \leq k \leq K$. Let $\mathbf{Y}_k$ and M_k be defined as in Assumption 6.3.2.3. Define the function $\boldsymbol{\psi}_k : X \times M_k \times X \times X \times X \times P \to \mathbb{R}^{n_x}$ such that $\forall(\tilde{\boldsymbol{\gamma}}, \tilde{\mathbf{M}}, \tilde{\mathbf{z}}_k, \tilde{\mathbf{z}}_{k-1}, \tilde{\mathbf{z}}_{k-2}, \mathbf{p}) \in X \times M \times X \times X \times X \times P$,

$\boldsymbol{\psi}_k(\tilde{\boldsymbol{\gamma}}, \tilde{\mathbf{M}}, \tilde{\mathbf{z}}_k, \tilde{\mathbf{z}}_{k-1}, \tilde{\mathbf{z}}_{k-2}, \mathbf{p}) = \tilde{\mathbf{z}}_k^*$, where the i^{th} component of $\tilde{\mathbf{z}}_k^*$ is given by the loop:

$$\begin{aligned}
&\textbf{\textit{for}} \;\; i = 1, \ldots, n_x \;\; \textbf{\textit{do}} \\
&\quad \tilde{z}_{k,i}^* := \tilde{\gamma}_i - \frac{b_{k,i}(\tilde{\boldsymbol{\gamma}}, \mathbf{z}_{k-1}(\mathbf{p}), \mathbf{z}_{k-2}(\mathbf{p}), \mathbf{p}) + \sum_{j<i} \tilde{m}_{ij}(\tilde{z}_{k,j}^* - \tilde{\gamma}_j) + \sum_{j>i} \tilde{m}_{ij}(\tilde{z}_{k,j} - \tilde{\gamma}_j)}{\tilde{m}_{ii}} \\
&\textbf{\textit{end}}.
\end{aligned} \tag{6.3.12}$$

Remark 6.3.6. Note that this definition is analogous to Definition 3.17 in Stuber et al.[300] with the modification that the $\mathbf{b}$ function is indexed by the timestep k and is dependent on the implicit functions of the two previous timesteps ($k-1$ and $k-2$) to account for the dependence on the two prior timesteps in two-step PILMS methods. For the first block of equations where we utilize the implicit Euler form: $\mathbf{h}_1 = \boldsymbol{\xi}_0^1$, we will still utilize this definition of $\boldsymbol{\psi}_1$ with $\mathbf{b}_1$ as $\mathbf{b}_1(\tilde{\boldsymbol{\gamma}}, \mathbf{z}_0(\mathbf{p}), \mathbf{z}_0(\mathbf{p}), \mathbf{p}) = \mathbf{Y}_1 \boldsymbol{\xi}_0^1(\tilde{\boldsymbol{\gamma}}, \mathbf{z}_0(\mathbf{p}), \mathbf{p})$.

If we were somehow capable of easily calculating $\mathbf{M}_k$ (and thus $\mathbf{B}_k$) for each k, and select $\tilde{\mathbf{M}} = \mathbf{M}_k$, then Definition 6.3.5 would define a semi-explicit representation of the implicit function solution $\mathbf{z}_k(\cdot)$ through its mean value form. This is important as we use this property to calculate valid convex and concave relaxations of $\mathbf{z}_k(\cdot)$ via relaxations of its mean value form componentwise as composite relaxations of $\boldsymbol{\psi}_k$. Stuber et al.[300] demonstrated exactly why relaxing the Newton-Raphson fixed-point iteration doesn't work as intended for general systems when approached in a naive way. That is, even though convex/concave relaxations are calculable by simply applying the rules for constructing McCormick relaxations and composite relaxations, they will not converge when used within a B&B algorithm for global optimization. Rules for circumventing this were developed for general nonlinear systems[300], which rely on relaxing $\mathbf{z}_k$ through the mean value form approximated using $\boldsymbol{\psi}_k$. We follow this approach in this work.

With these definitions of $\boldsymbol{\psi}_k$, $\mathbf{M}_k$, and $\mathbf{B}_k$, we can state the following theorem which provides

the result that convex and concave relaxations of the implicit functions $\mathbf{z}_k : P \to X$ for $k = 1, \dots K$, and their subgradient information, can be calculated in a practical way. This result follows directly from Thm. 3.25 of Stuber et al.[300] with the notational and functional modifications made here specifically for the implicit integration forms and a corresponding sequential-block relaxation construction procedure accounted for in the definition of $\boldsymbol{\psi}_k$ (Def. 6.3.5) and $\mathbf{M}_k$ (Def. 6.3.3).

Theorem 6.3.3. Let $\mathbf{h}_1(\hat{\mathbf{z}}_1, \mathbf{p}) = \boldsymbol{\xi}_0^1(\hat{\mathbf{z}}_1, \mathbf{z}_0(\mathbf{p}), \mathbf{p})$ and let $\mathbf{h}_k(\hat{\mathbf{z}}_k, \mathbf{p}) = \boldsymbol{\theta}_{k-2}^2(\hat{\mathbf{z}}_k, \mathbf{z}_{k-1}(\mathbf{p}), \mathbf{z}_{k-2}(\mathbf{p}), \mathbf{p})$ with $k \geq 2$ and $\boldsymbol{\theta} \in \{\boldsymbol{\zeta}, \boldsymbol{\xi}\}$. Let $\mathbf{z}_i^{cv}, \mathbf{z}_i^{cc} : P \to X$ be known convex and concave relaxations of $\mathbf{z}_i$ on P, respectively, for $i = 0$ and $i \in \{k-1, k-2\}$ for $k \geq 2$ and let $\mathbf{s}_{\mathbf{z}_i}^{cv}, \mathbf{s}_{\mathbf{z}_i}^{cc} : P \to \mathbb{R}^{n_p \times n_x}$ be known subgradients of those relaxations on P. Let $\lambda \in [0, 1]$ and $\bar{\mathbf{p}} \in P$. Let $\mathbf{u}_{\mathbf{B}k}, \mathbf{o}_{\mathbf{B}k}$ be composite relaxations of $\mathbf{B}_k$ on $X \times X \times \cdots \times X \times P$ and $\mathcal{S}_{\mathbf{u}_{Bk}}, \mathcal{S}_{\mathbf{o}_{Bk}}$ be composite subgradients of $\mathbf{u}_{\mathbf{B}k}, \mathbf{o}_{\mathbf{B}k}$, respectively. Let $\bar{\mathbf{u}}_{\psi_k}, \bar{\mathbf{o}}_{\psi_k}$ be composite relaxations of $\boldsymbol{\psi}_k$ on $X \times M \times X \times X \times X \times P$ and $\mathcal{S}_{\bar{\mathbf{u}}_{\psi k}}, \mathcal{S}_{\bar{\mathbf{o}}_{\psi k}}$ composite subgradients of $\bar{\mathbf{u}}_{\psi_k}, \bar{\mathbf{o}}_{\psi_k}$, respectively. Then, for any $r \in \mathbb{N}_+$ ($r \geq 1$) the elements of the sequences $\{\mathbf{z}_k^{j,cv}\}_{j=0}^r$ and $\{\mathbf{z}_k^{j,cc}\}_{j=0}^r$ calculated within Algorithm 2 are convex and concave relaxations of $\mathbf{z}_k$ on P, respectively. Further, the elements of the sequences $\{\mathbf{s}_{\mathbf{z}_k}^{j,cv}\}_{j=0}^r$ and $\{\mathbf{s}_{\mathbf{z}_k}^{j,cc}\}_{j=0}^r$ calculated in Algorithm 2 are subgradients of the elements of the sequences $\{\mathbf{z}_k^{j,cv}\}_{j=0}^r$ and $\{\mathbf{z}_k^{j,cc}\}_{j=0}^r$, respectively. Furthermore, Algorithm 3 returns $\mathbf{z}^{cv}$ and $\mathbf{z}^{cc}$, convex and concave relaxations of $\mathbf{z}$ on P, respectively, along with their respective subgradients $\mathbf{s}_{\mathbf{z}}^{cv}$ and $\mathbf{s}_{\mathbf{z}}^{cc}$.

Proof. The proof will proceed as follows. First, we will show that relaxations of $\mathbf{M}_k$ are computable. Then, we will show that these relaxations can be used to calculate relaxations of $\boldsymbol{\psi}_k$, for which the implicit function $\mathbf{z}_k$ is a fixed-point for every $\mathbf{p} \in P$.

Consider an arbitrary timestep $2 \leq k \leq K$ and the corresponding block of equations

$$\mathbf{h}_k(\hat{\mathbf{z}}_k, \mathbf{p}) = \boldsymbol{\theta}_{k-2}^2(\hat{\mathbf{z}}_k, \mathbf{z}_{k-1}(\mathbf{p}), \mathbf{z}_{k-2}(\mathbf{p}), \mathbf{p}) = \mathbf{0}.$$

Since Algorithm 2 is defined consistently with Stuber et al.[300], we will show how relaxations of $\boldsymbol{\psi}_k$ as defined in Definition 6.3.5 (and their subgradients) are computable. By the parametric mean value theorem[300], we have

$$\mathbf{M}_k(\mathbf{p})(\mathbf{z}_k(\mathbf{p}) - \boldsymbol{\gamma}(\mathbf{p})) = -\mathbf{Y}_k\mathbf{h}_k(\boldsymbol{\gamma}(\mathbf{p}),\mathbf{p}) = -\mathbf{Y}_k\boldsymbol{\theta}^2_{k-2}(\boldsymbol{\gamma}(\mathbf{p}),\mathbf{z}_{k-1}(\mathbf{p}),\mathbf{z}_{k-2}(\mathbf{p}),\mathbf{p}),\ \forall \mathbf{p}\in P.$$

By definition of $\mathbf{y}^i$, $i = (k-1)n_x + 1,\ldots,kn_x$, by the parametric mean value theorem (in the definition of $\mathbf{M}_k(\mathbf{p})$), any convex and concave relaxations of $\mathbf{z}_k$ on P which are also valid for $\boldsymbol{\gamma}$, are valid for $\mathbf{y}^i$ for $i = (k-1)n_x + 1,\ldots,kn_x$ (see Lemma 3.16 Stuber et al.[300] for further reading). By construction (Line 6, Alg. 2) $\boldsymbol{\gamma}^j$ satisfies this condition for $\mathbf{z}_k^{j,a}$ and $\mathbf{z}_k^{j,A}$. Therefore, for each j, the affine functions $\mathbf{z}_k^{j,a}, \mathbf{z}_k^{j,A}$ are respectively valid convex and concave relaxations of $\mathbf{y}^i$ (from Def. 6.3.3) on P for every $i = (k-1)n_x + 1,\ldots,kn_x$. As a result, relaxations of $\mathbf{M}_k$ on P are calculable as:

$$\begin{aligned}\mathbf{M}_k^{j,cv}(\mathbf{p}) :=& \mathbf{u}_{\mathbf{B}k}(\mathbf{z}_k^{j,a}(\mathbf{p}),\mathbf{z}_k^{j,A}(\mathbf{p}),\ldots,\mathbf{z}_k^{j,a}(\mathbf{p}),\mathbf{z}_k^{j,A}(\mathbf{p}),\mathbf{p})\\ \mathbf{M}_k^{j,cc}(\mathbf{p}) :=& \mathbf{o}_{\mathbf{B}k}(\mathbf{z}_k^{j,a}(\mathbf{p}),\mathbf{z}_k^{j,A}(\mathbf{p}),\ldots,\mathbf{z}_k^{j,a}(\mathbf{p}),\mathbf{z}_k^{j,A}(\mathbf{p}),\mathbf{p}).\end{aligned}$$

Thus, by definition, $\mathbf{M}_k^{j,cv}, \mathbf{M}_k^{j,cc}$ are convex and concave relaxations of $\mathbf{M}_k$ on P, respectively, for every j. The corresponding subgradient calculations follow analogously. Now, we will show that $\mathbf{z}_k$ is a fixed-point of $\boldsymbol{\psi}_k$.

By Assumption 6.3.2.2, we have $0 \notin M_{k,ii} \supset m_{k,ii}(P),\ \forall i$. Then, for $i = 1,\ldots,n_x$, we can

write:

$$z'_{k,i}(\mathbf{p}) = \sum_{j<i} m_{k,ij}(\mathbf{p})(z_{k,j}(\mathbf{p}) - \gamma_j(\mathbf{p})) + \sum_{j>i} m_{k,ij}(\mathbf{p})(z_{k,j}(\mathbf{p}) - \gamma_j(\mathbf{p})), \tag{6.3.13}$$

$$z_{k,i}(\mathbf{p}) = \gamma_i(\mathbf{p}) - \frac{b_{k,i}(\boldsymbol{\gamma}(\mathbf{p}), \mathbf{z}_{k-1}(\mathbf{p}), \mathbf{z}_{k-2}(\mathbf{p}), \mathbf{p}) + z'_{k,i}(\mathbf{p})}{m_{k,ii}(\mathbf{p})}. \tag{6.3.14}$$

By Definition 6.3.5, we can write:

$$\begin{aligned} z^*_{k,1}(\mathbf{p}) &= \psi_{k,1}(\boldsymbol{\gamma}(\mathbf{p}), \mathbf{M}_k(\mathbf{p}), \mathbf{z}_k(\mathbf{p}), \mathbf{z}_{k-1}(\mathbf{p}), \mathbf{z}_{k-2}(\mathbf{p}), \mathbf{p}) \\ &= \gamma_1(\mathbf{p}) - \frac{b_{k,1}(\boldsymbol{\gamma}(\mathbf{p}), \mathbf{z}_{k-1}(\mathbf{p}), \mathbf{z}_{k-2}(\mathbf{p}), \mathbf{p}) + \sum_{j>i} m_{k,1j}(\mathbf{p})(z_{k,j}(\mathbf{p}) - \gamma_j(\mathbf{p}))}{m_{k,11}(\mathbf{p})} \\ &= z_{k,1}(\mathbf{p}). \end{aligned}$$

By induction, it follows that $z_{k,i}(\mathbf{p}) = \psi_i(\boldsymbol{\gamma}(\mathbf{p}), \mathbf{M}_k(\mathbf{p}), \mathbf{z}_k(\mathbf{p}), \mathbf{z}_{k-1}(\mathbf{p}), \mathbf{z}_{k-2}(\mathbf{p}), \mathbf{p}) = z^*_{k,i}$ for every i. Thus, $\mathbf{z}_k$ is a fixed-point of $\boldsymbol{\psi}_k$ for every $\mathbf{p} \in P$.

By hypothesis, we have valid convex and concave relaxations $\mathbf{z}^{cv}_{k-1}(\mathbf{p}), \mathbf{z}^{cv}_{k-2}(\mathbf{p})$ and $\mathbf{z}^{cc}_{k-1}(\mathbf{p}), \mathbf{z}^{cc}_{k-2}(\mathbf{p})$ of $\mathbf{z}_{k-1}(\mathbf{p}), \mathbf{z}_{k-2}(\mathbf{p})$ on P, respectively, for $k \geq 2$ and their corresponding subgradients. Further, by design $\boldsymbol{\gamma} : P \to X$ is affine (both convex and concave). As a result, relaxations of the function $\mathbf{b}_k = \mathbf{Y}_k \boldsymbol{\theta}_{k-2}$ (as defined in Definition 6.3.5) on P are calculable as

$$\mathbf{u}_{\mathbf{b}_k}(\boldsymbol{\gamma}(\mathbf{p}), \boldsymbol{\gamma}(\mathbf{p}), \mathbf{z}^{cv}_{k-1}(\mathbf{p}), \mathbf{z}^{cc}_{k-1}(\mathbf{p}), \mathbf{z}^{cv}_{k-2}(\mathbf{p}), \mathbf{z}^{cc}_{k-2}(\mathbf{p}), \mathbf{p}) \tag{6.3.15}$$

$$\mathbf{o}_{\mathbf{b}_k}(\boldsymbol{\gamma}(\mathbf{p}), \boldsymbol{\gamma}(\mathbf{p}), \mathbf{z}^{cv}_{k-1}(\mathbf{p}), \mathbf{z}^{cc}_{k-1}(\mathbf{p}), \mathbf{z}^{cv}_{k-2}(\mathbf{p}), \mathbf{z}^{cc}_{k-2}(\mathbf{p}), \mathbf{p}) \tag{6.3.16}$$

and similarly, their subgradients are calculable.

We must also consider the $k = 1$ case. Then, we have the corresponding block of equations:

$$\mathbf{h}_1(\hat{\mathbf{z}}_1, \mathbf{p}) = \boldsymbol{\xi}_0^1(\hat{\mathbf{z}}_1, \mathbf{z}_0(\mathbf{p}), \mathbf{p}) = \mathbf{0}.$$

By hypothesis, we have $\mathbf{z}_0^{cv}, \mathbf{z}_0^{cc}$ and $\mathbf{s}_{\mathbf{z}_0}^{cv}, \mathbf{s}_{\mathbf{z}_0}^{cc}$. Then, the previous results still hold with $\mathbf{z}_{k-1}^{cv}, \mathbf{z}_{k-2}^{cv} := \mathbf{z}_0^{cv}$, $\mathbf{z}_{k-1}^{cc}, \mathbf{z}_{k-2}^{cc} := \mathbf{z}_0^{cc}$, $\mathbf{s}_{\mathbf{z}_{k-1}}^{cv}, \mathbf{s}_{\mathbf{z}_{k-2}}^{cv} := \mathbf{s}_{\mathbf{z}_0}^{cv}$, and $\mathbf{s}_{\mathbf{z}_{k-1}}^{cc}, \mathbf{s}_{\mathbf{z}_{k-2}}^{cc} := \mathbf{s}_{\mathbf{z}_0}^{cc}$ in (6.3.15) and (6.3.16). It follows immediately that if we know $\mathbf{z}_k^{j,cv}$ and $\mathbf{z}_k^{j,cc}$, then

$$\begin{aligned}
\mathbf{z}_k^{j+1,cv}(\cdot) :=& \bar{\mathbf{u}}_{\psi_k}(\boldsymbol{\gamma}(\cdot), \boldsymbol{\gamma}(\cdot), \mathbf{M}_k^{j,cv}(\cdot), \mathbf{M}_k^{j,cc}(\cdot), \mathbf{z}_k^{j,cv}(\cdot), \mathbf{z}_k^{j,cc}(\cdot), \mathbf{z}_{k-1}^{cv}(\cdot), \mathbf{z}_{k-1}^{cc}(\cdot), \mathbf{z}_{k-2}^{cv}(\cdot), \mathbf{z}_{k-2}^{cc}(\cdot), \cdot) \\
\mathbf{z}_k^{j+1,cc}(\cdot) :=& \bar{\mathbf{o}}_{\psi_k}(\boldsymbol{\gamma}(\cdot), \boldsymbol{\gamma}(\cdot), \mathbf{M}_k^{j,cv}(\cdot), \mathbf{M}_k^{j,cc}(\cdot), \mathbf{z}_k^{j,cv}(\cdot), \mathbf{z}_k^{j,cc}(\cdot), \mathbf{z}_{k-1}^{cv}(\cdot), \mathbf{z}_{k-1}^{cc}(\cdot), \mathbf{z}_{k-2}^{cv}(\cdot), \mathbf{z}_{k-2}^{cc}(\cdot), \cdot).
\end{aligned}$$

are convex and concave relaxations of $\mathbf{z}_k$ on P, respectively. The analogous result holds for their subgradients. Since Algorithm 2 provides $\mathbf{z}_k^{j,cv}, \mathbf{z}_k^{j,cc}$ for $j = 0$, by induction, we conclude that this result (and the analogous subgradient result) holds for every $j = 1, \ldots, r$. Thus, the elements of $\{\mathbf{z}_k^{j,cv}\}_{j=0}^r$ and $\{\mathbf{z}_k^{j,cc}\}_{j=0}^r$ as calculated within Algorithm 2 are valid convex and concave relaxations of $\mathbf{z}_k$ on P, respectively, and $\mathbf{s}_{\mathbf{z}_k}^{j,cv}$ and $\mathbf{s}_{\mathbf{z}_k}^{j,cc}$ are their respective subgradients for $j = 0, \ldots, r$. Since we showed that relaxations and their subgradients are calculable by Algorithm 2 for each timestep $k = 1, \ldots, K$, we conclude that Algorithm 3 returns $\mathbf{z}^{cv}$ and $\mathbf{z}^{cc}$, valid convex and concave relaxations of $\mathbf{z}$ on P, respectively, along with their respective subgradients $\mathbf{s}_{\mathbf{z}}^{cv}$ and $\mathbf{s}_{\mathbf{z}}^{cc}$. □

6.3.3 Partition Convergence

In this section, we show that Algorithm 3 generates a relaxation which exhibits partition convergence given that Assumption 6.3.2 is entirely satisfied. This result follows directly from established properties of the implicit relaxation algorithm presented in Stuber et al.[300] and

Algorithm 1 Affine Reference Function for Relaxations of Implicit Functions

procedure AFF($\mathbf{c}, \mathbf{C}, \mathbf{s_c}, \mathbf{s_C}, \lambda, X, P, \bar{\mathbf{p}}$)
  for $i \leftarrow 1$ **to** n_x **do**
    $X_i^a \leftarrow c_i + \sum_{j=1}^{n_p} (\mathbf{s_c}^{\mathrm{T}})_{ij}(P_j - \bar{p}_j)$ ▷ Interval arithmetic calculation
    $X_i^A \leftarrow C_i + \sum_{j=1}^{n_p} (\mathbf{s_C}^{\mathrm{T}})_{ij}(P_j - \bar{p}_j)$ ▷ Interval arithmetic calculation
    $\Omega_i \leftarrow \lambda X_i^a + (1-\lambda) X_i^A$ ▷ Interval arithmetic calculation
    if $\omega_i^L < x_i^L$ **then**
      $(\mathbf{s_c})_{ij} \leftarrow 0, \ \forall j = 1, \ldots, n_p$
      $c_i \leftarrow x_i^L$
    end if
    if $\omega_i^U > x_i^U$ **then**
      $(\mathbf{s_C})_{ij} \leftarrow 0, \ \forall j = 1, \ldots, n_p$
      $C_i \leftarrow x_i^U$
    end if
  end for
  return $\mathbf{c}, \mathbf{C}, \mathbf{s_c}, \mathbf{s_C}$
end procedure

Algorithm 2 Relaxation of a Single Block of a 2-Step PILMS Method

procedure BLOCK($\mathbf{h}_k$, X,$\mathbf{p}$, $\bar{\mathbf{p}}$, P, $\mathbf{z}_{k-1}^{cv}$, $\mathbf{z}_{k-1}^{cc}$,$\mathbf{s}_{\mathbf{z}_{k-1}}^{cv}$,$\mathbf{s}_{\mathbf{z}_{k-1}}^{cc}$,$\mathbf{z}_{k-2}^{cv}$, $\mathbf{z}_{k-2}^{cc}$,$\mathbf{s}_{\mathbf{z}_{k-2}}^{cv}$,$\mathbf{s}_{\mathbf{z}_{k-2}}^{cc}$,$r$,$\lambda$)
  $\mathbf{z}_k^{0,cv}, \mathbf{z}_k^{0,cc}, \mathbf{s}_{\mathbf{z}_k}^{0,cv}, \mathbf{s}_{\mathbf{z}_k}^{0,cc} \leftarrow \mathbf{x}^L, \mathbf{x}^U, \mathbf{0}, \mathbf{0}$
  for $j \leftarrow 0$ **to** $r-1$ **do**
    $\mathbf{c}, \mathbf{C}, \mathbf{s_c}, \mathbf{s_C} \leftarrow$ AFF($\mathbf{z}_k^{j,cv}(\bar{\mathbf{p}}), \mathbf{z}_k^{j,cc}(\bar{\mathbf{p}}), \mathbf{s}_{\mathbf{z}_k}^{j,cv}(\bar{\mathbf{p}}), \mathbf{s}_{\mathbf{z}_k}^{j,cc}(\bar{\mathbf{p}}), \lambda, X, P, \bar{\mathbf{p}}$)
    $\mathbf{z}_k^{j,a} \leftarrow \mathbf{c} + \mathbf{s_c}^{\mathrm{T}}(\mathbf{p} - \bar{\mathbf{p}})$ ▷ Affine relaxation lower bound
    $\mathbf{z}_k^{j,A} \leftarrow \mathbf{C} + \mathbf{s_C}^{\mathrm{T}}(\mathbf{p} - \bar{\mathbf{p}})$ ▷ Affine relaxation upper bound
    $\boldsymbol{\gamma}^j \leftarrow \lambda \mathbf{z}_k^{j,a} + (1-\lambda)\mathbf{z}_k^{j,A}$
    $\mathbf{s}_{\boldsymbol{\gamma}}^j \leftarrow \lambda \mathbf{s_c} + (1-\lambda)\mathbf{s_C}$
    $\mathbf{M}^{j,cv} \leftarrow \mathbf{u}_{\mathbf{B}k}(\mathbf{z}_k^{j,a}, \mathbf{z}_k^{j,A}, \ldots, \mathbf{z}_k^{j,a}, \mathbf{z}_k^{j,A}, \mathbf{p})$
    $\mathbf{M}^{j,cc} \leftarrow \mathbf{o}_{\mathbf{B}k}(\mathbf{z}_k^{j,a}, \mathbf{z}_k^{j,A}, \ldots, \mathbf{z}_k^{j,a}, \mathbf{z}_k^{j,A}, \mathbf{p})$
    $\mathbf{s}_{\mathbf{M}}^{j,cv} \leftarrow \mathcal{S}_{\mathbf{u}_B}(\mathbf{z}_k^{j,a}, \mathbf{z}_k^{j,A}, \mathbf{s_c}, \mathbf{s_C}, \ldots, \mathbf{z}_k^{j,a}, \mathbf{z}_k^{j,A}, \mathbf{s_c}, \mathbf{s_C}, \mathbf{p})$
    $\mathbf{s}_{\mathbf{M}}^{j,cc} \leftarrow \mathcal{S}_{\mathbf{o}_B}(\mathbf{z}_k^{j,a}, \mathbf{z}_k^{j,A}, \mathbf{s_c}, \mathbf{s_C}, \ldots, \mathbf{z}_k^{j,a}, \mathbf{z}_k^{j,A}, \mathbf{s_c}, \mathbf{s_C}, \mathbf{p})$
    $\mathbf{z}_k^{j+1,cv} \leftarrow \bar{\mathbf{u}}_\psi(\boldsymbol{\gamma}^j, \boldsymbol{\gamma}^j, \mathbf{M}^{j,cv}, \mathbf{M}^{j,cc}, \mathbf{z}_k^{j,cv}, \mathbf{z}_k^{j,cc}, \mathbf{z}_{k-1}^{cv}, \mathbf{z}_{k-1}^{cc}, \mathbf{z}_{k-2}^{cv}, \mathbf{z}_{k-2}^{cc}, \mathbf{p})$
    $\mathbf{z}_k^{j+1,cc} \leftarrow \bar{\mathbf{o}}_\psi(\boldsymbol{\gamma}^j, \boldsymbol{\gamma}^j, \mathbf{M}^{j,cv}, \mathbf{M}^{j,cc}, \mathbf{z}_k^{j,cv}, \mathbf{z}_k^{j,cc}, \mathbf{z}_{k-1}^{cv}, \mathbf{z}_{k-1}^{cc}, \mathbf{z}_{k-2}^{cv}, \mathbf{z}_{k-2}^{cc}, \mathbf{p})$
    $\mathbf{s}_{\mathbf{z}_k}^{j+1,cv} \leftarrow \mathcal{S}_{\bar{\mathbf{u}}_\psi}(\boldsymbol{\gamma}^j, \boldsymbol{\gamma}^j, \mathbf{s}_\gamma^j, \mathbf{s}_\gamma^j, \mathbf{M}^{j,cv}, \mathbf{M}^{j,cc}, \mathbf{s}_{\mathbf{M}}^{j,cv}, \mathbf{s}_{\mathbf{M}}^{j,cc}, \mathbf{z}_k^{j,cv}, \mathbf{z}_k^{j,cc}, \mathbf{s}_{\mathbf{z}_k}^{j,cv}, \mathbf{s}_{\mathbf{z}_k}^{j,cc}, \mathbf{p})$
    $\mathbf{s}_{\mathbf{z}_k}^{j+1,cc} \leftarrow \mathcal{S}_{\bar{\mathbf{o}}_\psi}(\boldsymbol{\gamma}^j, \boldsymbol{\gamma}^j, \mathbf{s}_\gamma^j, \mathbf{s}_\gamma^j, \mathbf{M}^{j,cv}, \mathbf{M}^{j,cc}, \mathbf{s}_{\mathbf{M}}^{j,cv}, \mathbf{s}_{\mathbf{M}}^{j,cc}, \mathbf{z}_k^{j,cv}, \mathbf{z}_k^{j,cc}, \mathbf{s}_{\mathbf{z}_k}^{j,cv}), \mathbf{s}_{\mathbf{z}_k}^{j,cc}, \mathbf{p})$
  end for
  return $\mathbf{z}_k^{r,cv}, \mathbf{z}_k^{r,cc}, \mathbf{s}_{\mathbf{z}_k}^{r,cv}, \mathbf{s}_{\mathbf{z}_k}^{r,cc}$
end procedure

Algorithm 3 Relaxations of Parametric IVPs using 2-Step PILMS Methods

procedure IVPBound(X, $\mathbf{p}$, $\bar{\mathbf{p}}$, P,$\mathbf{h}$, $\mathbf{x}_0$, r, λ)
  $\mathbf{z}_0^{cv}, \mathbf{z}_0^{cc}, \mathbf{s}_{\mathbf{z}_0}^{cv}, \mathbf{s}_{\mathbf{z}_0}^{cv} \leftarrow$ McCormickRelax($X, P, \mathbf{x}_0(\mathbf{p})$) ▷ McCormick relaxation of $\mathbf{x}_0$
  $\mathbf{z}_1^{cv}(\mathbf{p}), \mathbf{z}_1^{cc}(\mathbf{p}), \mathbf{s}_{\mathbf{z}_1}^{cv}(\mathbf{p}), \mathbf{s}_{\mathbf{z}_1}^{cc}(\mathbf{p}) \leftarrow$ Block($\mathbf{h}_1, X, \mathbf{p}, \bar{\mathbf{p}}, P, \mathbf{z}_0^{cv}, \mathbf{z}_0^{cc}, \mathbf{s}_{\mathbf{z}_0}^{cv}, \mathbf{s}_{\mathbf{z}_0}^{cc}, \mathbf{x}^L, \mathbf{x}^U, \mathbf{0}, \mathbf{0}, r, \lambda$)
  for $k \leftarrow 2$ **to** K **do**
    $k_1, k_2 \leftarrow k-1, k-2$
    $\mathbf{z}_k^{cv}(\mathbf{p}), \mathbf{z}_k^{cc}(\mathbf{p}), \mathbf{s}_{\mathbf{z}_k}^{cv}(\mathbf{p}), \mathbf{s}_{\mathbf{z}_k}^{cc}(\mathbf{p}) \leftarrow$Block($\mathbf{h}_k, X, \mathbf{p}, \bar{\mathbf{p}}, P, \mathbf{z}_{k_1}^{cv}, \mathbf{z}_{k_1}^{cc}, \mathbf{s}_{\mathbf{z}_{k_1}}^{cv}, \mathbf{s}_{\mathbf{z}_{k_1}}^{cc}, \mathbf{z}_{k_2}^{cv}, \mathbf{z}_{k_2}^{cc}, \mathbf{s}_{\mathbf{z}_{k_2}}^{cv}, \mathbf{s}_{\mathbf{z}_{k_2}}^{cc}, r, \lambda$)
  end for
  $\mathbf{z}^{cv}(\mathbf{p}), \mathbf{s}_{\mathbf{z}}^{cv}(\mathbf{p}) \leftarrow (\mathbf{z}_0^{cv}(\mathbf{p}), \mathbf{z}_1^{cv}(\mathbf{p}), \ldots, \mathbf{z}_K^{cv}(\mathbf{p})),\ (\mathbf{s}_{\mathbf{z}_0}^{cv}(\mathbf{p}), \mathbf{s}_{\mathbf{z}_1}^{cv}(\mathbf{p}), \ldots, \mathbf{s}_{\mathbf{z}_K}^{cv}(\mathbf{p}))$
  $\mathbf{z}^{cc}(\mathbf{p}), \mathbf{s}_{\mathbf{z}}^{cc}(\mathbf{p}) \leftarrow (\mathbf{z}_0^{cc}(\mathbf{p}), \mathbf{z}_1^{cc}(\mathbf{p}), \ldots, \mathbf{z}_K^{cc}(\mathbf{p})),\ (\mathbf{s}_{\mathbf{z}_0}^{cc}(\mathbf{p}), \mathbf{s}_{\mathbf{z}_1}^{cc}(\mathbf{p}), \ldots, \mathbf{s}_{\mathbf{z}_K}^{cc}(\mathbf{p}))$
  return $\mathbf{z}^{cv}(\mathbf{p}), \mathbf{z}^{cc}(\mathbf{p}), \mathbf{s}_{\mathbf{z}}^{cv}(\mathbf{p}), \mathbf{s}_{\mathbf{z}}^{cc}(\mathbf{p})$
end procedure

ensures that a B&B algorithm when used in conjunction with the relaxations developed herein will terminate in finite time.

Proposition 6.3.4. Consider a nested sequence of intervals $\{P_q\}$, $P_q \subset P$, $q \in \mathbb{N}$, such that $\{P_q\} \to [\bar{\mathbf{p}}, \bar{\mathbf{p}}]$ for some $\bar{\mathbf{p}} \in P$. Let $\mathbf{z}_q^{cv}, \mathbf{z}_q^{cc}$ be relaxations of $\mathbf{z}$ on P_q obtained using Algorithm 3 and denote the state variable convex and concave relaxations at the k^{th} timestep as $\mathbf{z}_{k,q}^{cv}, \mathbf{z}_{k,q}^{cc}$, respectively. Let $\phi_q^{cv}(\cdot) = u_\phi(\mathbf{z}_q^{cv}(\cdot), \mathbf{z}_q^{cc}(\cdot), \cdot)$ be a convex relaxation of the objective function ϕ on P^q. Let $\hat{\phi}_q^{cv} = \min_{\mathbf{p} \in P_q} \phi_q^{cv}(\mathbf{p})$. Then $\lim_{q\to\infty} \phi_q^{cv} = \phi(\mathbf{z}(\bar{\mathbf{p}}), \bar{\mathbf{p}})$.

Proof. Consider $K = 1$, then $\mathbf{h} = \mathbf{h}_1(\hat{\mathbf{z}}_1, \mathbf{p}) = \boldsymbol{\xi}_0^1(\hat{\mathbf{z}}_1, \hat{\mathbf{z}}_0, \mathbf{p})$ trivially reduces to the form considered explicitly in the lower-bounding problem formulation (16) in Stuber et al.[300] and Lemma 4.3 in Stuber et al.[300] with respect to the state variables $\hat{\mathbf{z}}_1$ as $\hat{\mathbf{z}}_0 = \mathbf{x}_0(\mathbf{p})$. Now we proceed for general $K > 1$ with a proof by contradiction. Suppose that for $K > 1$ we have $\lim_{q\to\infty} \phi_q^{cv} \neq \phi(\mathbf{z}(\bar{\mathbf{p}}), \bar{\mathbf{p}})$. As u_ϕ^{cv} is constructed by generalized McCormick relaxations, it is continuous and exhibits partition convergence [271]. This implies that there must exist a k such that $\lim_{q\to\infty} \mathbf{z}_{k,q}^{cv} \neq \mathbf{z}_{k,q}(\bar{\mathbf{p}})$ or $\lim_{q\to\infty} \mathbf{z}_{k,q}^{cc} \neq \mathbf{z}_{k,q}(\bar{\mathbf{p}})$. However, this implies that either $\lim_{q\to\infty} \mathbf{z}_{k-1,q}^{cv} \neq \mathbf{z}_{k-1,q}(\bar{\mathbf{p}})$ or $\lim_{q\to\infty} \mathbf{z}_{k-1,q}^{cc} \neq \mathbf{z}_{k-1,q}(\bar{\mathbf{p}})$ as continuity of $z_{q,k}$ and componentwise partition convergence of valid relaxations $\mathbf{z}_{k-1,q}^{cv}$ and $\mathbf{z}_{k-1,q}^{cc}$ result in componentwise partition convergence of $\mathbf{z}_{k,q}^{cv}$ and $\mathbf{z}_{k,q}^{cc}$.

Continuing this reasoning by reverse induction, we see that this would imply that for $k = 1$ the partition convergence is not observed, which is a direct contradiction of the $K = 1$ case. □

Theorem 6.3.5. Let X be defined as in Def. 4.1 of Stuber et al.[300] and suppose Assumption 4.2 of Stuber et al.[300] holds. Further, suppose the hypotheses of Proposition 6.3.4 hold. Then, after finitely many iterations, κ, the spatial B&B algorithm of Stuber et al.[300] with relaxations calculated by Algorithm 3 terminates either with an ϵ-optimal global solution such that $\alpha_\kappa - \beta_\kappa \leq \epsilon$, or a certificate of infeasibility.

Proof. Lemma 4.4 through 4.8 in Stuber et al.[300] hold from the pointwise convergence property of Proposition 6.3.4 and continuity assumptions without further modification. As a result, the finite convergence theorem (Thm. 4.9) in Stuber et al.[300] holds. □

6.3.4 Implementation

All numerical experiments in this work were run on a single thread of an Intel Xeon E3-1270 v5 3.60/4.00GHz (base/turbo) processor with 16GB ECC RAM allocated to an Ubuntu 18.04LTS operating system virtual machine and Julia v1.1[36]. Absolute and relative convergence tolerances for the B& B algorithm of 10^{-2} and 10^{-5}, respectively were specified for all example problems. A solver extension to the EAGO.jl package [327] was created to implement the algorithm detailed above and is located at `https://github.com/PSORLab/EAGODifferential.jl`. The Intel MKL (2019 Update 2)[95] was used to perform all LAPACK[11, 320] and BLAS[40] routines.

An affine lower-bounding problem was constructed using the relaxations evaluated at their midpoint and their respective subgradients[191]. Two iterations of the PILMS method developed here were used to compute the lower bound, (i.e., $r = 2$). The lower-bounding problems were

solved using CPLEX 12.8.0 [66]. In addition to the relaxations of the objective function, linear objective cuts were used to restrict the feasibility region based on the global upper bound. Additionally, duality-based bounds tightening (DBBT) was performed[251], using the multipliers obtained when solving the lower-bounding problem.

Any feasible point provides a valid upper bound for the optimization problem. As no equality constraints are left in any of the formulations addressed below, we can simply solve the system for the state variables at a specified point in the subdomain of interest. Any inequality constraints in the upper-bounding problem are then evaluated at this solution point and the upper-bounding problem is feasible if this point is feasible. We use an adaptation of this approach to furnish the upper bound. At each node in the B&B tree, the parametric ODE-IVP is numerically integrated at the midpoint of the respective decision space, $\mathbf{p}^* = \text{mid}(P_q)$, using the fixed-stepsize integration scheme corresponding to the discretized system of equations in (6.2.3). The DifferentialEquations.jl[239] package is used to perform each numerical integration step.

Prior to calculating the lower bound, it is often advantageous to contract the initial state space bounds for the q^{th} node, $X_{q,0}$ by application of a parametric interval method. We perform up to five iterations of parametric interval-Newton or terminate if the bounds fail to further contract within 5 iterations. These state variable bounds $X_{q,\kappa}$ are then stored as the state variable bounds of the resulting child nodes provided the problem is feasible. It merits noting that the PILMS methods developed here are at least as tight as the parametric interval method for bounds tightening. However, it requires additional calculations to determine the values of the relaxations and their respective subgradients. As such, a decrease in overall computation time can be realized by applying this contractor then computing the relaxation of the implicit function.

6.4 Illustrative Examples

Example 6.4.1 (A Scalar Parametric ODE-IVP)**.** Consider the simple scalar ODE-IVP presented in Section 4.1 of Sahlodin and Chachuat[256] with a single parameter:

$$\begin{aligned} \dot{x}(p,t) &= -x^2 + p, \quad t \in I = [0,1], \ p \in P = [-1,1] \\ x_0(p) &= 9, \quad p \in P. \end{aligned} \tag{6.4.1}$$

We make use of the developed theory of relaxations of implicit functions to construct bounds of the solution on P with initial state bounds as $X = [0.1, 9]$. A single iteration of the parametric interval-Newton method yields new state bounds entirely within the interior of the prior bounds. By Theorem 6.3.1 above and Theorem 5.1.8 in Neumaier[219], this verifies the existence of a unique implicit function in X. Both the two-step PILMS methods (6.3.5) and (6.3.7) were used to construct bounds with and without the use of a parametric interval contraction method.

In Figure 6.4.1, the relaxations are obtained using only the state bounds specified as X. As illustrated in Figure 6.4.1, the bounds obtained expand as time progresses and increasing the number of discretization points K can lead to weaker bounds due to the dependency issue common among set-valued arithmetic [256]. However, these bounds still exhibit the pointwise convergence property and the use of these methods in conjunction with domain reduction techniques can alleviate these issues. This is observed by the tight bounds obtained using $K = 200$ discretization points and contracting the state variables prior to each block solve of the relaxation algorithm. Finally, we note that while the standard algorithm for n_x-dimensional systems requires an $n_x \times n_x$ matrix inversion to calculate $\mathbf{Y}_k$ and precondition each block prior to applying the iterative relaxation method (Alg. 3), the algorithm can be run in a modest amount of time as detailed in Table 6.4.1. Furthermore, other preconditioners which may be less expensive

to compute can be used for larger systems, if necessary. In this one-dimensional case, the midpoint inverse of the interval derivative was used.

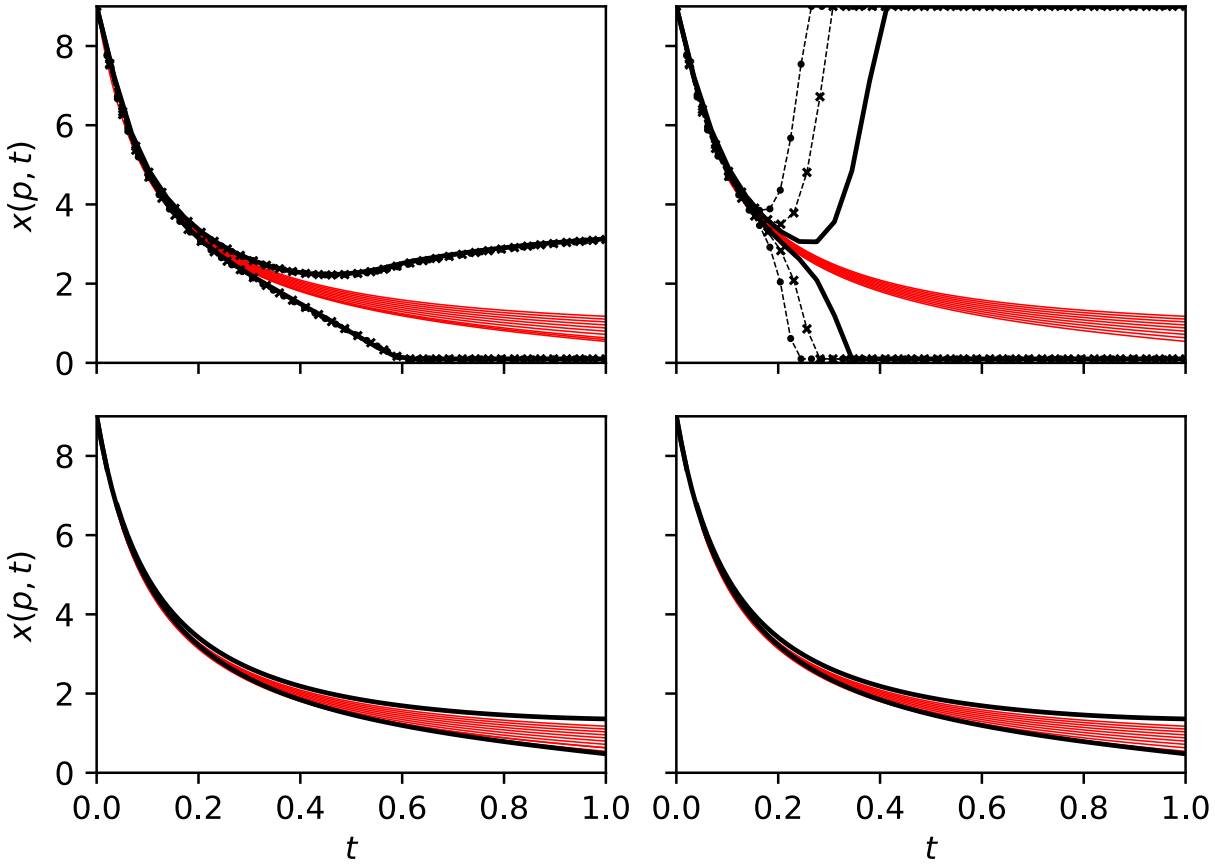

FIGURE 6.4.1: The results of Example 6.4.1 are illustrated here. Lower and upper bounds shown here are respectively the minimum and maximum values of $z_k^{cv}(\cdot)$ and $z_k^{cc}(\cdot)$ attained on P for each k. **Upper Left:** Bounds on $x(p,t)$ of (6.4.1) obtained by using the two-step AM method for $r = 3$ with $K = 30$ (black), $K = 40$ (light gray), and $K = 50$ (gray). **Upper Right:** Bounds on $x(p,t)$ of (6.4.1) obtained by using a two-step BDF method for $r = 3$ with $K = 30$ (black), $K = 40$ (light gray), and $K = 50$ (gray). **Lower Left:** Bounds on $x(p,t)$ of (6.4.1) obtained by using the two-step AM method for $r = 3$ after applying 5 iterations of the parametric interval-Newton method using $K = 200$. **Lower Right:** Bounds on $x(p,t)$ of (6.4.1) obtained by using a second-order BDF method for $r = 3$ after applying 5 iterations of the parametric interval-Newton method using $K = 200$.

In order to compare our bounding results with those of Sahlodin and Chachuat [256], we evaluate the enclosure width, $\Delta\omega_k$, at discretization point k corresponding to time t_k, as the

Timesteps	$K = 10$	$K = 20$	$K = 30$	$K = 40$	$K = 50$
AM, PI + relax (CPU sec)	15×10^{-5}	29×10^{-5}	44×10^{-5}	61×10^{-5}	73×10^{-5}
BDF, PI + relax (CPU sec)	14×10^{-5}	28×10^{-5}	43×10^{-5}	58×10^{-5}	73×10^{-5}
AM, PI + relax, width at $t = 1$	0.703	0.806	0.841	0.8550	0.8634
AM, PI only, width at $t = 1$	1.812	1.906	1.938	1.955	1.955
BDF, PI + relax, width at $t = 1$	0.703	0.810	0.901	0.937	0.951
BDF, PI only, width at $t = 1$	0.713	0.822	0.916	0.954	0.969

TABLE 6.4.1: The CPU times required to construct parametric interval (PI) bounds and convex/concave relaxation-based bounds (and associated subgradients) for Example 6.4.1 as well as the enclosure width at $t = 1$ associated with each variant used to construct relaxations.

distance between the maximum of the concave relaxation and the minimum of the convex relaxation. For the iterative relaxation method, Alg. 3, $\Delta\omega_k = \max_{p\in P} z_k^{r,cc}(p) - \min_{p\in P} z_k^{r,cv}(p)$. For interval methods, this simplifies to the diameter of the bounding interval. For the AM-type method, the parametric interval-Newton method is less effective and the relaxation method presented here achieves a bound approximately one-third the enclosure width at $t = 1$. In the case of the BDF method, the parametric interval-Newton method is far more effective resulting in significantly tighter refinements of X and only a 1-2% improvement was achieved using the corresponding relaxation method. The results are summarized in Table 6.4.1.

The computational performance of the presented methods were compared with the timings presented by Sahlodin and Chachuat [256]. The results are contained in Table 6.4.1. For a fair comparison across hardware specifications, we normalize for each CPU's single-core IPC using the Cinebench R15 (Maxon, Newbury Park, CA) single-core benchmark. We estimate that the single core performance of our computer (Cinebench R15 score of 172) is approximately 1.51 times faster than the computer used by Sahlodin and Chachuat [256] (Cinebench R15 score of 114). Therefore, we compute hardware-normalized timings by dividing the results of Sahlodin

and Chachuat [256] by the factor 1.51. For $K = 100$ timesteps, both the two-step AM method and the two-step BDF method terminate in only $14{\times}10^{-4}$ s and achieve 1.143 and 1.026 width bounds at $t = 1$, respectively, for a single p point evaluation. In contrast, the method of Sahlodin and Chachuat [256] takes a normalized time of between $93{\times}10^{-4}$ s and $29{\times}10^{-3}$ s to compute bounds of width 0.914 at $t = 1$ using orders 5 to 20 Taylor-series expansions. While the results obtained via the iterative relaxation (Alg. 3) are only bounds of the numerical (approximate) solution, decreased computation time relative to the discretize-then-relax approach may be advantageous for some problems.

Example 6.4.2 (Reversible Isomerization)**.** Consider the simple kinetic equations that result from a reversible isomerization reaction with $k_1 = 10$ and $k_2 = 10^{-2}$ given by (6.4.2). The system initially consists of only the isomer $\mathbf{x}_0(p) = (p, 0)$, with $p \in P = [0.8, 1]$ and the reaction is allowed to progress for $t \in I = [0, 1]$ seconds. The X bounds were chosen to be nonnegative and below the maximum value of 1. As the uncertainty is present only in the initial condition, the $J_k^s(X, P)$ is real valued, and the implicit function exists as $J_k^s(X, P)$ is nonsingular. This first-order ODE-IVP system is a typical example of a stiff system as the reverse reaction occurs on a much longer time scale than the forward reaction. Bounds were computed using both the two-step AM method and two-step BDF method. As illustrated by the plots provided in Figure 6.4.2, tight bounds on the characteristically-stiff system (6.4.2) can be readily obtained using either two-step PILMS method. For each method used, less than 1ms was needed to generate each relaxation and their respective subgradients at a given p reference value.

$$\begin{aligned} \dot{x}_1(\mathbf{p}, t) &= k_2 x_2 - k_1 x_1 \\ \dot{x}_2(\mathbf{p}, t) &= k_1 x_1 - k_2 x_2 \end{aligned} \tag{6.4.2}$$

The sharpness of the bounds provided here can be attributed to the fact that the right-hand

side of (6.4.2) is a parametric linear equation and uncertainty is only introduced via the initial condition. As a result, the two-step PILMS schemes themselves result in parametric linear algebraic equations as is each Newton-type update. As a consequence, the relaxations calculated by Algorithm 1 become tight as do the relaxations generated by Algorithm 3.

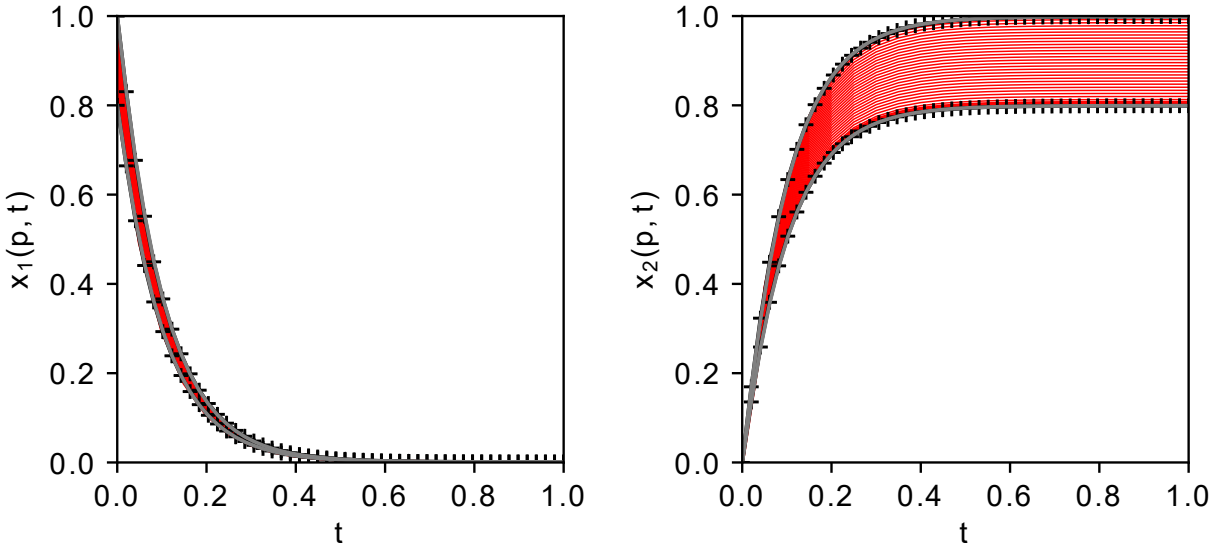

FIGURE 6.4.2: The results of Example 6.4.2 are illustrated. **(Left):** Bounds on $x_1(p,t)$ of (6.4.2) determined using the two-step BDF (black) and Adams-Moulton methods (gray-plus) using $K = 50$ discretization points. **(Right):** Bounds on $x_2(p,t)$ of (6.4.2) determined using the two-step BDF (black) and Adams-Moulton methods (gray-plus) using $K = 50$ discretization points.

Example 6.4.3 (Kinetic Parameter Estimation)**.** Consider the kinetic mechanism problem first presented in Mitsos et al.[191] as an adaptation of the parameter estimation problem encountered for the oxygen addition to cyclohexadienyl radicals[287, 306]. The reaction mechanism can be

modeled as the ODE-IVP:

$$\dot{x}_A(\mathbf{p},t) = k_1 x_Z x_Y - c_{O_2}(k_{2f}+k_{3f})x_A + \frac{k_{2f}}{K_2}x_D + \frac{k_{3f}}{K_3}x_B - k_5 x_A^2$$
$$\dot{x}_B(\mathbf{p},t) = k_{3f}c_{O_2}x_A - \left(\frac{k_{3f}}{K_3}+k_4\right)x_B, \quad \dot{x}_Z(\mathbf{p},t) = -k_1 x_Z x_Y$$
$$\dot{x}_D(\mathbf{p},t) = k_{2f}c_{O_2}x_A - \frac{k_{2f}}{K_2}x_D, \quad \dot{x_Y}(\mathbf{p},t) = -k_{1s} x_Z x_Y$$
$$x_{A,0} = 0, x_{B,0} = 0, x_{D,0} = 0, x_{Y,0} = 0.4, x_{Z,0} = 140,$$

where x_j is the concentration of species $j \in \{A, B, D, Y, Z\}$. The constants are then given by $T = 273$, $K_2 = 46\exp(6500/T - 18)$, $K_3 = 2K_2$, $k_1 = 53$, $k_{1s} = k_1 \times 10^{-6}$, $k_5 = 1.2 \times 10^{-3}$, and $c_{O_2} = 2 \times 10^{-3}$. Intensity versus time data is available in Stuber[296] and exhibits a known dependency on the species concentrations as $I^c = x_A + \frac{2}{21}x_B + \frac{2}{21}x_D$ originating from the Beer-Lambert Law with a multi-species correction[283]. The unknown reaction rate constants are $k_{2f} \in [10, 1200]$, $k_{3f} \in [10, 1200]$, and $k_4 \in [0.001, 40]$, and together form the parameter vector $\mathbf{p} = (k_{2f}, k_{3f}, k_4)$ for the parameter estimation problem.

In Mitsos et al.[191], the explicit Euler discretization of the problem was solved by directly calculating bounds and relaxations on the state variables for the discretization from explicitly-defined equations then propagating calculated bounds and relaxations to the objective function. An implicit Euler discretization was constructed in Stuber et al.[300] and solved via the global optimization of implicit functions approach. State variable bounds on X provided in Stuber et al.[300] were used to bound the PILMS methods used. In that work, at least nine suboptimal local minima were discovered and reported, motivating the need for deterministic global optimization. The objective function for this problem is given by

$$\phi(\hat{\mathbf{z}}, \mathbf{p}) = \sum_{i=1}^{n} \left(I_i^c - I_i^d\right)^2 \tag{6.4.3}$$

where I_i^c are the calculated intensity values at timestep i from the model and I_i^d are the values corresponding to the experimental data. The performance of the algorithm is detailed in Table 6.4.2 and illustrated in Figure 6.4.3.

For $K = 100$, the two-step methods both failed to reach convergence within the 7200 CPU seconds allowed. Further degradation in the convergence rates was observed on initial trials that used $K = 50$ steps to discretize the system. In principle, an upper bound could be furnished by using a local solver to locate a feasible point of the full-space formulation. However, such a routine will readily become the most expensive step of the solution process and the overall solution time will depend heavily on heuristics used to limit the number of local solves. We omit this here for the sake of simplicity. Both the two-step Adams-Moulton and the two-step BDF method yield a superior fit (a smaller minimum SSE) at termination compared with the implicit Euler method for each number of time steps owing to the higher numerical accuracy of the second-order method. This is true even for the cases that failed to converge after 7200 CPU seconds. As such, the implicit methods presented here may be chosen to achieve an optimal trade-off between computational time spent on each block relaxation and the number of total block relaxations. For high values of K, the number of nonlinear computations in intermediate steps is proportional to K and complexity of the block sequential preconditioning step of the relaxation scale linearly with K. As detailed in Table 6.4.2, no clear relationship exists between the solution time and K. This is not entirely unexpected as each discretized model represents a fundamentally different optimization problem that must be solved. As illustrated by the quick convergence of the two-step AM method for $K = 200$, some cases exist where the decreased truncation error can be obtained at no further cost. Note that the implicit Euler integration scheme with $K = 200$ was performed in Stuber et al.[300]. An implementation of the explicit Euler approach presented in Mitsos et al.[191] was also tested but the solver failed to converge to the

desired tolerance within the time limit (a relative gap of 3.558×10^{-2} was achieved at termination).

TABLE 6.4.2: The CPU times required to solve the kinetic parameter estimation problem (Ex. 6.4.3) using each bounding method for various K values after applying five iterations of the parametric interval-Newton method.

Solution Method	Steps K	Iterations	Time per Iteration	Solve Time	SSE Computed
Implicit Euler	100	33987	45×10^{-3}s	29.7min	26947.246
	200	23,525	59×10^{-3}s	23.4min	16796.038
2-Step AM	100	62024	12×10^{-2}s	>2h	N/A*
	200	6068	22×10^{-2}s	22.6min	13077.998
2-Step BDF	100	88408	81×10^{-3}s	>2h	N/A*
	200	27600	26×10^{-2}s	>2h	N/A*
Explicit Euler	100	>300,000	23×10^{-4}s	>2h	N/A
	200	>300,000	24×10^{-4}s	>2h	N/A

*Note that while these examples failed to converge to a global minimum within the 7200-second limit, in some cases, progressive convergence is observed and additional run time may allow for full convergence.In the case of the 2-Step BDF with $K = 200$, a lower bound of 12876.763 and an upper bound of 13336.471 was furnished on termination. In contrast, minimal convergence is observed for either 2-Step method with $K = 100$. This suggests that the 2-Step PILMS method may perform better when higher K values are used.

Example 6.4.4 (Transient Plug Flow Reactor (PFR))**.** In this example we consider a single-species degradation reaction in an air-sparged PFR. Assuming that dispersion in the PFR is negligible, and first-order degradation proceeds under isothermal conditions, the system can be modeled by the following dimensionless partial differential equation (PDE):

$$\frac{\partial x}{\partial t} = -\frac{\partial x}{\partial y} - Dax \tag{6.4.4}$$

where x is the nondimensional spatiotemporal-varying species concentration, $Da = k\tau$ is the

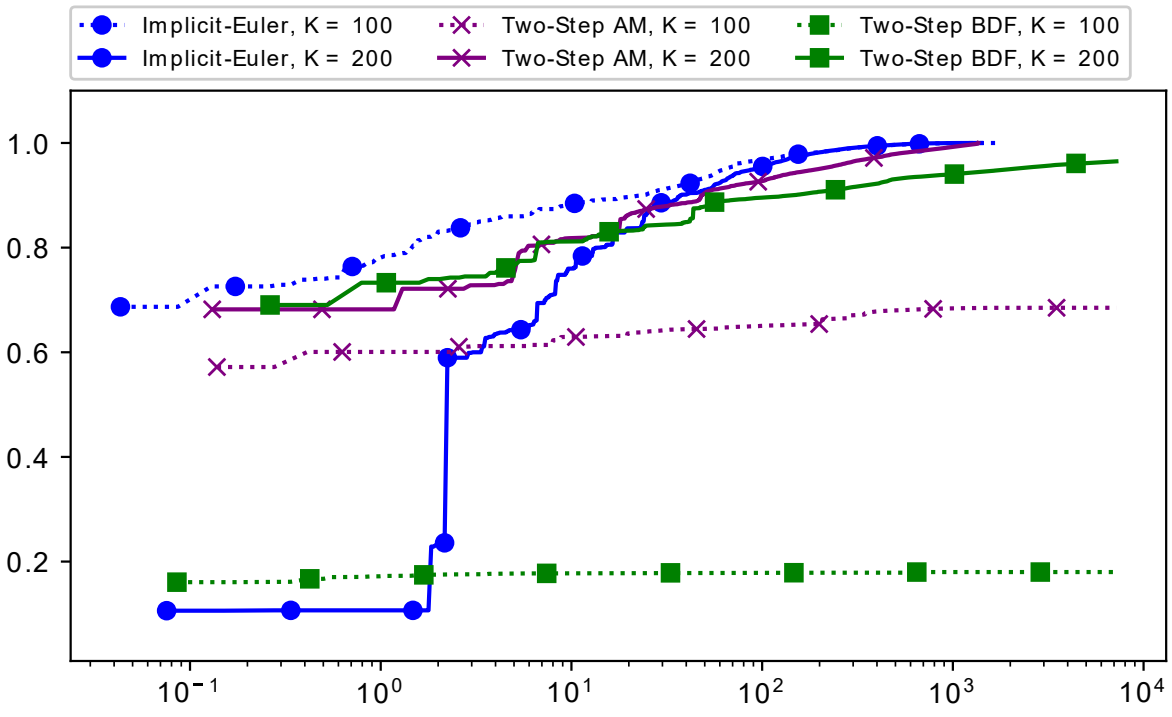

FIGURE 6.4.3: A convergence plot of each variation of the global optimization algorithm for timesteps $K = 100$ and $K = 200$ as demonstrated on the kinetic parameter estimation example (Ex. 6.4.3). For each K, the algorithm using the two-step AM method produces the tightest upper bound earlier in time. However, in each case, the algorithm using the implicit Euler scheme exhibits faster overall convergence. The user must consider the trade-off between accuracy of the integration method (integration error) and solution time.

Damköhler number, τ [s] is the mean residence time, and k [s^{-1}] is the first-order reaction rate constant. Parameter values for the example are as follows: the reactor volume is $1.5{\times}10^4$ cm^3, volumetric flow rate is $1.5{\times}10^3$ cm^3/h, and $k = 0.35h^{-1}$. The dimensionless axial spatial coordinate is taken to be $y \in [0, 1]$. This PDE is solved via the method of lines assuming an inlet concentration fixed to $\tilde{x}_0 = 1$ and an otherwise zero initial concentration. This yields the following spatially-discretized system of IVPs to be solved:

$$\frac{d\tilde{\mathbf{x}}}{dt}(\mathbf{p}, t) = -\frac{\Delta\tilde{\mathbf{x}}}{\Delta y} - Da\tilde{\mathbf{x}} \tag{6.4.5}$$

where $\Delta\tilde{x}_i = \tilde{x}_i - \tilde{x}_{i-1}$ is the backwards finite difference of the states, and $\tilde{\mathbf{x}} \in \mathbb{R}^N$ is the vector of state variables at each discrete spatial grid point (of which there are N). The backwards difference scheme was chosen for spatial discretization as convection dominates the axial transport under operating conditions and central differencing schemes lead to instability unless stepsizes are extremely restricted when using explicit methods. For comparison and visualization, the system of ODE-IVPs given in (6.4.5) was numerically integrated using both implicit-Euler and an explicit fourth-order Runge-Kutta (RK4). As seen in Figure 6.4.4, the solution obtained using the RK4 method exhibits oscillations throughout the concentration profiles, while the solution obtained using the implicit Euler method does not yield oscillatory behavior. This stiff behavior results from the step change in the concentration profile specified at the initial condition. This motivates the use of implicit methods that exhibit superior numerical stability at higher temporal stepsizes and the use of the methods developed in the Relaxation Algorithm section to construct relaxations used in the optimization formulation. In fact, a higher accuracy model could be obtained by using central differencing spatial discretization in conjunction with the implicit relaxation method detailed in that section. We abstained from doing this in the implicit formulation to provide a more direct comparison with the formulation used with explicit integration schemes.

The PFR model describes a step change in the inlet concentration, which may result from feedstock variability or as part of start-up operations following a shut-down. We mandate that effluent concentration must stay below $\lambda = 0.08$. This limitation is somewhat contrived as actual conversion specifications are both process- and location-dependent. In the case of wastewater treatment, inlet ammonia concentrations may vary by multiple orders of magnitude between some residential and industrial sources. Since, the PFR model is known to possess monotonic concentration profiles with respect to both space and time, only a constraint on the last spatial discretization variable is required (i.e., the outlet). However, the sequential-block solution

structure of the algorithm and additional constraints may benefit the B& B algorithm in quickly fathoming nonviable solutions. The implicit optimization problem formulation is given as:

$$\begin{aligned} \phi^* &= \min_{p \in P} p \\ \text{s.t.} \quad & z_{K,\text{exit}}(p) - \lambda \leq 0 \end{aligned} \tag{6.4.6}$$

where $z_{K,\text{exit}}$ is simply the concentration at the exit of the PFR at the final time. Lastly, we assume that the reaction is mass-transport limited and the Damköhler number by changing the flow rate of sparging air, which modulates the local mixing rate. We assume that $Da = 0.1 + 0.3p$ where p is normalized to a value between one and zero.

We initially endeavored to solve this problem using the state bounds $X = [0, 1]^N$, using $K = 200$ fixed timesteps and $N = 20$ spatial discretization points. However, no convergence was observed. Additionally, the desired accuracy of the bounded model in concentration dictates an extremely small stepsize must be used even in conjunction with an implicit method. In order to achieve an absolute tolerance of the integrator of 10^{-3}, an initial stepsize of 8×10^{-8} is needed using implicit Euler while a stepsize of 2×10^{-4} is acceptable for either two-step AM or BDF methods. In this case, the prescribed absolute tolerance represents a significantly more restrictive limit than stability (as these are A-stable) or the stepsize limitation introduced to ensure that Assumption 6.3.2.3 is met. While the use of two-step implicit methods achieves four orders-of-magnitude improvement over the implicit Euler method with respect to accuracy of the bounded ODE-IVP system, the resulting 4000 state variable formulation is still intractable. We resolve these issues by resorting to a variable timestep scheme. The timesteps to be used are determined as follows: first, we note that the concentration profile is monotonic in time, spatial dimension, and that the concentration at a given point in time and space exhibits a monotonic dependence on p.

A variable stepsize temporal discretization scheme was determined by integrating the ODE-IVP over a span of values of p. The timesteps corresponding to the finest resolution discretization scheme was then used to formulate the optimization problem. This allows for a mere $K = 30$ temporal discretization points to be used in order to achieve the desired accuracy. As such, the model can then be solved in the single control variable as opposed to the 601 variables (600 state variables occurring the discretization of the ODE-IVP and 1 control variable). Interestingly, the two-step BDF method dramatically outperforms the two-step AM method on this problem. Only 347 iterations are required to achieve convergence of the BDF method which was achieved in 382 seconds of CPU time when the original bounds on X were used. The convergence profile is illustrated in Figure 6.4.5. For the first 260 seconds, minimal improvement on the lower and upper bounds occurs while the B&B routine naively partitions the decision space P, followed by a speedy convergence to an ϵ-optimal solution. In contrast, the two-step AM method never achieved convergence in the full 24 h run time nor, in fact, any improvement on the initial bounds in this time. As such the results were omitted from Figure 6.4.5 for the sake of clarity. One explanation for this stark contrast between methods may be due to the fact that BDF methods are known to integrate extremely stiff ODE-IVPs more efficiently than competing PILMS methods and this problem exhibits an extreme degree of stiffness about the inlet at the initial time value whereas the system addressed in Example 6.4.3 (for which the two-step BDF method had worse performance than the two-step AM method) is significantly less stiff.

Example 6.4.5 (Bounding State Trajectories of Denitrification in Biological Nutrient Removal)**.** Currently, most wastewater treatment plants operate at one-third efficiency, and aeration accounts for 45-75% of plant-wide energy consumption[249]. By utilizing highly-predictive modeling of a wastewater treatment system, control methods can be developed to optimize aeration operations and reduce these inefficiencies. Henze and Gradys[131] first activated sludge model (ASM1)

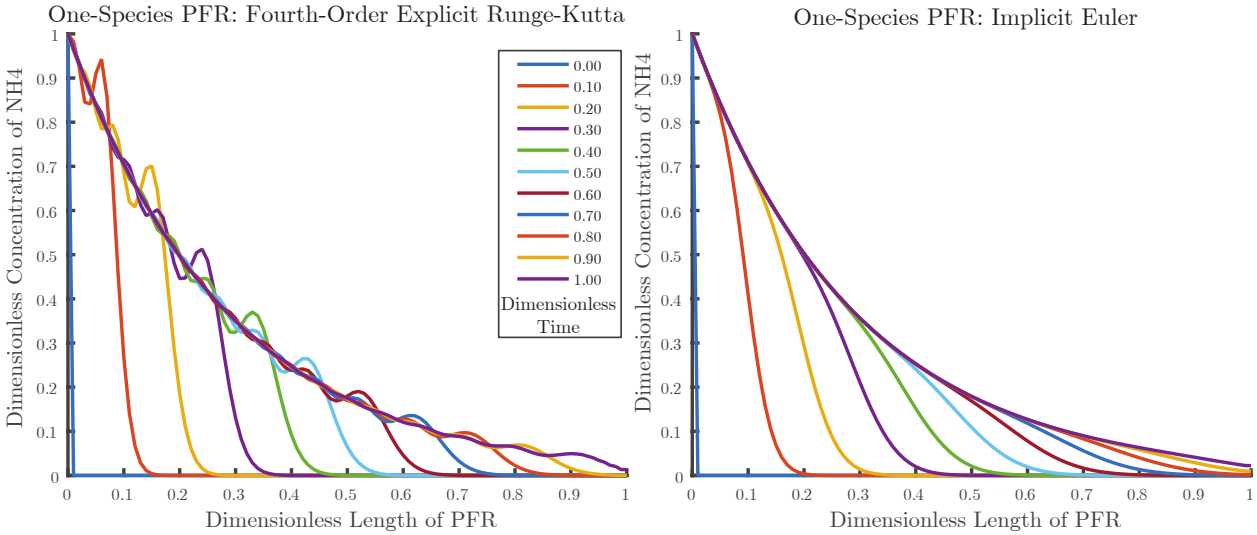

FIGURE 6.4.4: The one-species PFR model of Example 6.4.4 is simulated using (**Left**) the explicit fourth-order Runge-Kutta method and (**Right**) implicit (backward) Euler. The explicit method results in spurious oscillatory behavior of the concentration profiles which is not present using the implicit method.

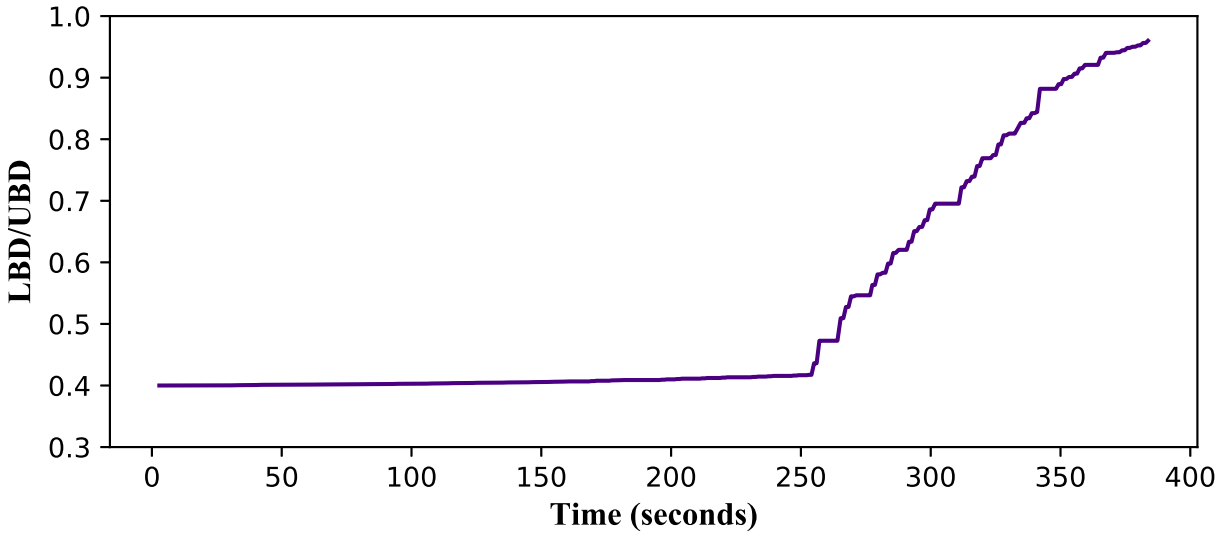

FIGURE 6.4.5: In this convergence plot, we see that the improvement of the bounds of (6.4.6) generated using the two-step BDF method in Example 6.4.4 remains stagnant until around 250 seconds when the these bounds begin to converge and the absolute convergence tolerance of 10^{-2} is satisfied for this example. At convergence, the relative gap is approximately $LBD/UBD = 0.99$.

provides a suitable foundation for dynamic model development. The ASM1[131] model has proved to be an excellent tool for modeling nitrification-denitrification processes. This model includes a system of 9 ODE-IVPs ($n_x = 9$) and the respective rate equations for state variables ranging from autotrophic and heterotrophic bacteria to substrate, ammonium, nitrogen, and dissolved oxygen. Here, the focus is on dissolved oxygen for optimizing aeration operations due to its interactions in the biological system model [162]. A sparged CSTR model of biological nutrient removal is detailed below which makes use of this ASM1 model. For typical processing parameters chosen for this case study, the system of equations in (6.4.7) describes the kinetics of the nine species of interest:

$$
\begin{aligned}
\dot{x}_1(\mathbf{p}, t) &= \tau(x_{in,1} - x_1) + 3.93(10 - x_1) + r_{11} \\
\dot{x}_2(\mathbf{p}, t) &= \tau(x_{in,2} - x_2) + 0.484x_2 + r_6 \\
\dot{x}_3(\mathbf{p}, t) &= \tau(x_{in,3} - x_3) + 0.484x_3 + r_5 \\
\dot{x}_4(\mathbf{p}, t) &= \tau(x_{in,4} - x_4) + r_2 \\
\dot{x}_5(\mathbf{p}, t) &= \tau(x_{in,5} - x_5) + r_7 \\
\dot{x}_6(\mathbf{p}, t) &= \tau(x_{in,6} - x_6) + r_8 \\
\dot{x}_7(\mathbf{p}, t) &= \tau(x_{in,7} - x_7) + r_9 \\
\dot{x}_8(\mathbf{p}, t) &= \tau(x_{in,8} - x_8) + 0.484x_8 + r_4 \\
\dot{x}_9(\mathbf{p}, t) &= \tau(x_{in,9} - x_9) + 0.484x_9 + r_{10}
\end{aligned}
\tag{6.4.7}
$$

where the initial condition is taken to be $\mathbf{x}_0 = (0.0, 91, 1075, 2.62, 33.31, 0.41, 0.93, 29.46, 2.54)$ [g/m^3] and the reaction rates $r_i,\ i \in \{2, 4, 5, 6, 7, 8, 9, 10, 11\}$ are given in (6.4.8). The half-saturation coefficients are given by, $K_s = 10$ [g/m^3], $K_X = 1$ [g/m^3], $K_{NO} = 0.5$ [g/m^3], $K_O = 0.3$ [g/m^3], $K_{NHA} = 1$ [g/m^3], the decay rates are $b_H = 0.039$ [day^{-1}] and $b_A = 0.002$

[day^{-1}]. The residence time in the reactor is $\tau = 1.85$ [day^{-1}]. The maximum specific hydrolysis rate is taken to be $k_H = 0.125$ [day^{-1}] and the ammonification rate is $k_A = 0.05$ [m^3/(g·day)] [131]. The biomass yield is taken to be $y_A = 0.24$. An inlet composition of $\mathbf{x}_{in} = (0, 0.001, 96, 64, 1, 12.5, 10.1, 160, 18.28)$ [g/m^3] is assumed.

$$
\begin{aligned}
r_2 &= -\frac{\mu_H}{y_H}\frac{x_3 x_4}{K_s + x_4}\left(\frac{x_1}{K_O + x_1} + \frac{0.8K_O}{K_O + x_1}\frac{x_5}{K_{NO} + x_5}\right) + k_H x_3 \frac{x_8/x_3}{K_X + x_8/x_3}\left(\frac{x_1}{K_O + x_1}\right) \\
&\quad \ldots + \frac{0.8K_O}{K_O + x_1}\left(\frac{x_5}{K_{NO} + x_1}\right) \\
r_4 &= 0.92(b_H x_3 + b_A x_2) - k_H x_3 \frac{x_8/x_3}{K_X + x_8/x_3}\left(\frac{x_1}{K_O + x_1}\right) + \frac{0.8K_O}{K_O + x_1}\left(\frac{x_5}{K_{NO} + x_5}\right) \\
r_5 &= \mu_H \frac{x_3 x_4}{K_s + x_4}\left(\frac{x_1}{K_O + x_1} + 0.8\frac{K_O}{K_O + x_1}\frac{x_5}{K_{NO} + x_5}\right) - b_H x_3 \\
r_6 &= \mu_A \frac{x_2 x_6}{K_{NHA} + x_6}\left(\frac{x_1}{K_O + x_1}\right) - b_A x_2 \\
r_7 &= -0.088\mu_H \frac{x_3 x_4}{K_s + x_4}\frac{K_O}{K_O + x_1}\left(\frac{x_5}{K_{NO} + x_5}\right) + \frac{\mu_A}{y_A}\frac{x_2 x_6}{K_{NHA} + x_6}\left(\frac{x_1}{K_O + x_1}\right) \\
r_8 &= -0.068\mu_H \frac{x_3 x_4}{K_s + x_4}\left(\frac{x_1}{K_O + x_1} + \frac{K_O}{K_O + x_1}\frac{0.8x_5}{K_{NO} + x_5}\right) - \mu_A \frac{4.23x_6}{K_{NHA} + x_6}\left(\frac{x_1 x_2}{K_O + x_1}\right) + k_A x_7 x_3 \\
r_9 &= -k_A x_7 x_3 + k_H x_3 \frac{x_9/x_3}{K_X + x_8/x_3}\left(\frac{x_1}{K_O + x_1}\right) + \frac{0.8K_O}{K_O + x_1}\left(\frac{x_5}{K_{NO} + x_5}\right) \\
r_{10} &= 0.063(b_H x_3 + b_A x_2) - k_H \frac{(x_9/x_3)}{K_X + x_8/x_3}\left(\frac{x_1 x_3}{K_O + x_1}\right) + \frac{0.8K_O}{(K_O + x_1)}\left(\frac{x_5}{(K_{NO} + x_5)}\right) \\
r_{11} &= -\left(0.32\mu_H \frac{x_4 x_3}{K_s + x_4} + 18.04\mu_A \frac{x_6 x_2}{K_{NHA} + x_6}\right)\frac{x_1}{K_O + x_1}
\end{aligned}
\tag{6.4.8}
$$

This system consists of several reactions that operate on significantly different time scales, and as a result is characterized by a large degree of stiffness [9]. In practice, exact kinetics may depend on the exact bacterial ecology of the process unit. Both the maximum specific growth rate for heterotrophs, μ_H, and for autotrophs, μ_A potentially depend on such ecological considerations. We

make use of the two-step AM method to generate bounds on the parametric ODE-IVP defined by, (6.4.7), and (6.4.8) with $\mu_H \in [0.14, 0.16]$ and $\mu_A \in [0.019, 0.021]$. Plots of the bounds generated using the two-step AM method are included in Figure 6.4.6 along with select trajectories from numerically integrating these equations at typical values of μ_A and μ_H in this range. No parametric interval method was used to contract the bounds prior to generating these plots. The system was then numerically integrated over an 8-hour time interval to simulate the resulting transients. The bounds on the states are given by $\mathbf{x}^L = (0, 90, 900, 0.25, 25, 0.2, 0.01, 25, 2.5)$ and $\mathbf{x}^U = (8, 94, 1100, 2.75, 40, 2.0, 0.93, 400, 50)$. Applying parametric interval-Newton yields state variable bounds that have been contracted into the interior of the initial bounds confirming that a unique implicit function exists on this domain.

Valid bounds can also be generated using the two-step BDF method. However, for the initial μ_H, μ_A intervals considered these bounds are not significantly tighter than the original bounds provided by $\mathbf{x}^L$ and $\mathbf{x}^U$ while the two-step AM method yield significant refinement to the initial bounds. Again, this is likely due to competing sources of wrapping effects present in either method: whether the additional function evaluation results in a more expansive relaxation than the inclusion of additional subtraction operators, or vice versa.

6.5 Concluding Remarks

In this work, the guaranteed global solution of dynamic optimization problems was addressed, with specific interest in the application to stiff parametric ODE-IVP systems. This interest was motivated by process systems engineering applications of model-based design, rigorous model validation, optimal control, and robust control.

In the developed approach, the dynamic optimization problem was reformulated as an

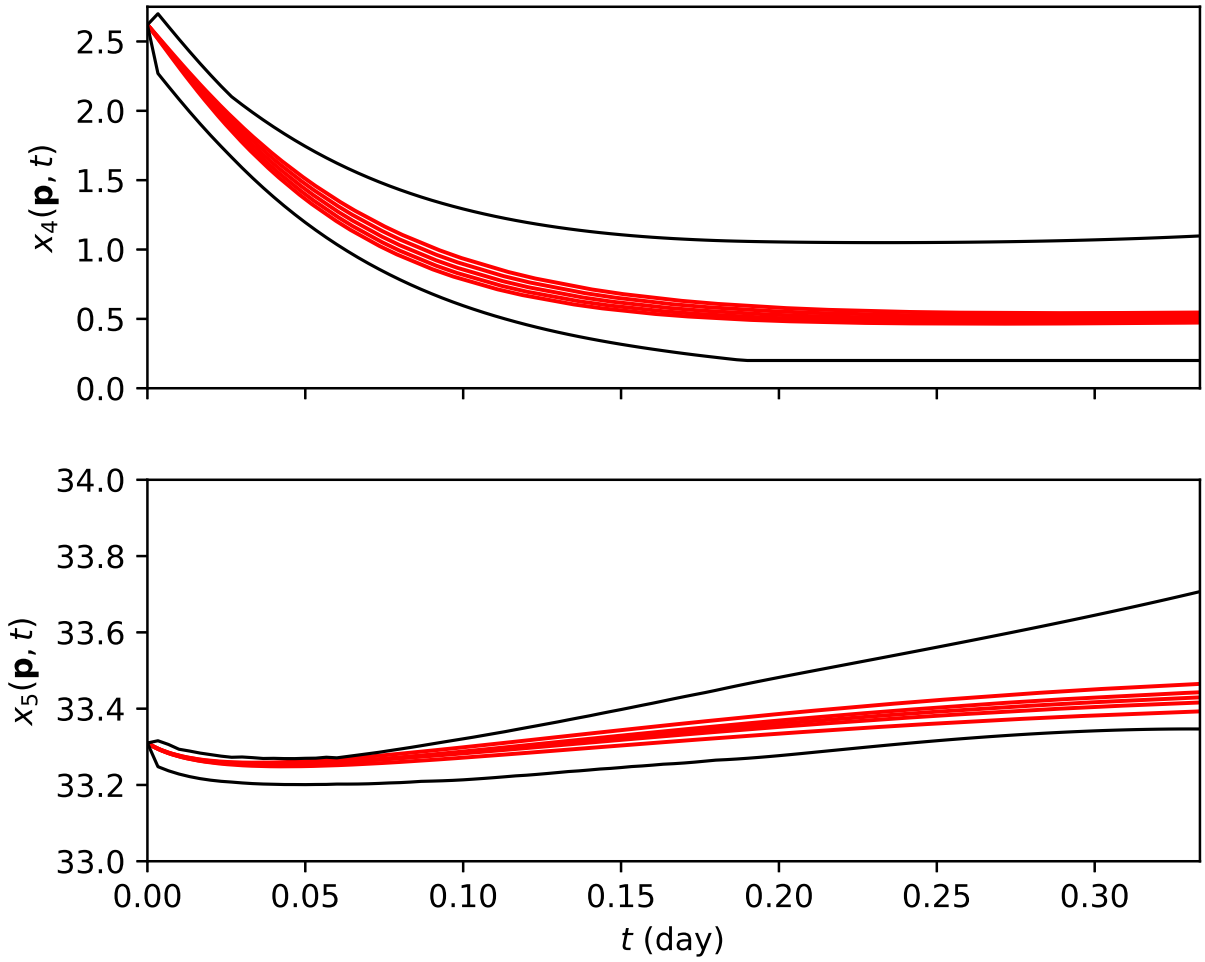

FIGURE 6.4.6: The results from Example 6.4.5 are illustrated. **(Top):** Bounds on $x_4(\mathbf{p}, t)$ determined using the two-step Adams-Moulton method using $K = 200$ discretization points. **(Bottom):** Bounds on $x_5(\mathbf{p}, t)$ determined using the two-step Adams-Moulton method using $K = 200$ discretization points.

equality-constrained NLP by discretizing the time horizon and applying an unconditionally-stable implicit integration scheme to form the equality constraint equations. Specifically, two second-order implicit linear multistep integration methods were considered: the two-step Adams-Moulton and the two-step backward-difference formula methods. The equality constraints were subsequently eliminated from the nonlinear programming formulation with the introduction of an implicit function as the (parametric) solution of the equality constraints taken as a large

system of parametric nonlinear algebraic equations. The algebraic systems formed from this approach exhibit a sparse block-diagonal occurrence matrix which can be exploited for numerical efficiency in numerical equation solving as well as in the construction of rigorous bounds and convex/concave relaxations of implicit function solutions. The theory of convex and concave relaxations of implicit functions [300] was extended to cover this class of problems with this special structure to provide the necessary bounds required for deterministic global optimization using the spatial B&B framework. The methods were demonstrated on five examples relevant in process systems engineering to illustrate the calculation of rigorous bounds and the solution of the nonconvex dynamic optimization problem to guaranteed global optimality. Overall, this approach yields tight, accurate, and fast bounds on numerical approximations of the state trajectories of stiff systems enabling the efficient solution of a class of deterministic global optimization problems with stiff dynamical systems embedded.

This work serves as the foundation for future work on robust design (including robust control) problems for rigorous worst-case performance and safety verification under transients as well as rigorous dynamic flexibility analysis for general nonconvex dynamical systems models.

Chapter 7

Validated Convex and Concave Relaxations of Parametric Solutions of Ordinary Differential Equations Via Implicit Integration Methods

Two novel methods for computing convex and concave relaxations of the solutions of parametric nonlinear ordinary differential equations (pODEs) are described herein. Convex/concave relaxations are necessary for deterministic global optimization algorithms of problems with embedded pODEs, a common class of problems that arise from dynamic simulations. Namely, these relaxations allow for construction of relaxed subproblems repeatedly solved within spatial branch and bound algorithm. The methods developed are based on recent work relating to discretize-then-relax methods, parametric implicit linear methods, and Interval Hermite-Obreschkoff methods for bounding solution sets of pODEs. In each discretize-then-relax algorithm, a two-stage procedure is performed in which valid convex/concave relaxations of the state variables are formed over the entire time-step and then pointwise-in-time convex/concave

relaxations are subsequently refined. The novel methods described in this chapter are applicable to the later step. Two case studies are examined which illustrate the potential advantages of using these novel approaches; namely, computational efficiency and tightness of the underlying relaxations.

7.1 Introduction

This chapter presents novel methods for computing state relaxations and interval bounds of the solutions of nonlinear parametric ordinary differential equations (pODEs) of the form given by:

$$\begin{aligned}\dot{\mathbf{x}}(\mathbf{p},t) = \frac{d\mathbf{x}}{dt}(\mathbf{p},t) = \mathbf{f}(\mathbf{x}(\mathbf{p},t),\mathbf{p}), \quad t\in I=[t_0,t_f],\ \mathbf{p}\in P \\ \mathbf{x}(\mathbf{p},t_0) = \mathbf{x}_0(\mathbf{p}), \quad \mathbf{p}\in P,\end{aligned} \tag{7.1.1}$$

with mappings $\mathbf{f} : D\times\Pi\to\mathbb{R}^{n_x}$ and $\mathbf{x}_0 : P\to D$, $D\subset\mathbb{R}^{n_x}$ and $\Pi\subset\mathbb{R}^{n_p}$ open sets, $P\in\mathbb{\Pi}$, and $I\in\mathbb{R}$. This information is essential for global stability and controllability analysis [41, 137]. Furthermore, the bounds and relaxations of solutions of pODEs are fundamental to deterministic global dynamic optimization methods [161, 229, 284], which are themselves relevant to a breadth of applications such as parameter estimation with dynamic models [287], drug scheduling [56, 175, 319], safety verification [140], and fault detection [232]. Unfortunately, the use of deterministic global optimization methods remains limited, as existing methods for computing sufficiently tight bounds on the parametric solution sets of pODEs (i.e., the *reachable set*) are computationally expensive.

Overestimation of the reachable set is an intrinsic feature of set-valued approaches for bounding solution trajectories of pODEs on a given parameter space [164]. This overestimation

occurs when set-valued approaches use convex approximations such as interval bounds, polyhedrals, or relaxations to enclose a nonconvex surface, since these representations are necessarily inexact. Additionally, these representations do not fully capture the dependence of intermediate terms on any shared parameters, leading to the well-known "dependency problem" [193, 255]. Accordingly, the preponderance of research in this area has focused on developing methods by which tight bounds may be efficiently computed, in the hopes of reducing overestimation at minimal computational cost. These methods can generally be divided into two broad families: continuous-time methods, in which relaxations are themselves represented by auxiliary ODEs; and discrete-time approaches, in which pODEs are discretized into a series of time points, and then relaxations are computed at each time point.

The first attempts at discrete-time methods were made as part of a simple existence and uniqueness test based on interval arithmetic [192]. Since then, this approach has been generalized to a family of two-stage methods that find their basis in Taylor series representations of functions [33, 214]. The first stage determines a stepsize ($h : t_{i+1} = t_i + h$) and *a priori* enclosure of the solution set of the pODEs system along the entire step ($t \in [t_i, t_{i+1}]$). In the second stage, a tightened enclosure is determined at the end of the step, t_{i+1}. The two stages are then repeated over the entire time domain or until the determined stepsize diminishes below user-specified tolerances. Subsequently, several modifications have been introduced to improve the calculated bounds, including the use of mean-value representations of the integration set [246], the introduction of a shrink-wrapping algorithm [35, 52], and adaptations regarding the preconditioning algorithms used in each step [168]. Interval methods have been extended to develop analogous convex and concave relaxations approaches which rely on characterizing the truncation error associated with each step by reduced-space convex and convex relaxations (McCormick relaxations) of the remainder terms [256] and McCormick-Taylor models [255]. Alternative approaches have also been described as Taylor models were constructed with

ellipsoidal enclosures of the remainders [314]. Houska et al. [138] then presented a set-stable integrator based on Taylor Model arithmetic along with an extensive analysis of its local asymptotic stability. Moreover, it has been noted that these discrete-time relaxations can be related to the analogous continuous-time approaches [315].

One such continuous-time relaxation method, established by Scott and Barton [270], is that of differential inequalities. As with all continuous-time approaches, this method represents state relaxations using an auxiliary system of ODEs that can then be integrated using state-of-the-art software, such as CVODES [274], which makes use of implicit integration schemes, adaptive time-stepping, and event detection protocols. However, the number of integration variables in the auxiliary ODEs with n_x state variables and n_p parameters are $4n_x$, or $4n_x + 2n_x n_p$ provided that sensitivity information is required. Additionally, the use of an event detection scheme is required to fully define the algorithm which may lead to numerical issues. Shen and Scott [276] have proposed methods to further tighten state relaxations by exploiting model redundancy using invariants. Subsequently, an affine relaxation-based method was detailed in which interval and affine relaxations are used concurrently to refine each other [123].

While continuous-time methods benefit from the use of implicit integration schemes, minimal attention has been paid to the use of implicit methods for constructing discrete-time relaxations. This has been limited to a preliminary evaluation by Rihm [247], in which the authors disregard the applicability of implicit Taylor series methods since the interval extension of the Jacobian of the underlying equation may not exist; a necessary condition for employing an interval-Newton method. However, they also noted that, in a simple nonstiff example, the implicit method may produce enclosures with widths that are thirteen orders of magnitude smaller than the corresponding explicit method, suggesting that a substantial benefit may be achieved provided that nonsingularity conditions for the Jacobian are met. Implicit linear multistep interval methods

were addressed by Marciniak et al. [174], but explicit consideration of pODEs was omitted. In addition, the authors used a fixed-point iteration method to solve these problems. The convergence of such methods and, in turn, the refinement of state bounds are only guaranteed in the limit [296]. Furthermore, Marciniak et al. [174] sought to directly adapt implicit linear multistep methods without consideration for potential improvements that may result from the consideration of analogous mean value representations. This, coupled with a detailed evaluation of a potential preconditioning method, may be expected to yield tighter relaxations without greatly increasing the computational burden.

In our previous work, Wilhelm et al. [328] developed methods to compute relaxations of discretized systems of equations derived by applying an implicit linear multistep integration scheme to pODEs. These relaxations provided rigorous bounds in the original decision of the nonlinear equations present at each discrete-time point. However, this approach omitted any explicit treatment of the discretization error. The principal contributions of this chapter build on these results in the following manner:

1. We extend the theory of interval implicit linear multistep methods used in a series of previous works to allow for rigorously bounding the solution sets of pODEs. We then build on this to generate convex/concave state relaxations using generalized McCormick relaxation theory.

2. We investigate implicit approaches by generalizing the interval Hermite-Obreschkoff method to apply to pODEs and then further develop this into methods for generating convex/concave state relaxations.

3. We present two case studies taken from the existing literature and compare the computational efficiency and tightness of the relaxations of the proposed methods.

This chapter is arranged as follows. In Section 7.2, we establish conventions for mathematical notation and provide a review of the relevant mathematical background. Section 7.3 details the construction of state relaxations and bounds of pODEs using implicit linear multistep methods, beginning with fixed stepsize interval methods and concluding with variable stepsize methods for computing state relaxations. Next, in Section 7.4, we examine the alternative interval Hermite-Obreschkoff methods. This is followed by two numerical case studies in Section 7.5. Lastly, we provide our concluding remarks and future research directions in Section 7.6.

7.2 Background

7.2.1 Parametric Ordinary Differential Equations

We adopt the following notation with respect to discrete time points for the pODEs problem (7.1.1). Let a series of discretization points be denoted $t_0 \leq t_1 \leq \ldots \leq t_m = T$. The stepsize between t_j and t_{j+1} is given by $h_j = t_{j+1} - t_j$. The solution of (7.1.1) for a parameter value $\mathbf{p} \in P$ at time t is denoted $\mathbf{x}(\mathbf{p}, t)$. State bounds of (7.1.1) evaluated at $t = t_j$ are denoted as X_j and the solution of (7.1.1) for a particular $\mathbf{p} \in P$ evaluated at $t = t_j$ is denoted $\mathbf{x}_j : P \to \mathbb{R}^{n_x}$. Throughout this chapter, it will be necessary to distinguish between initially known *a priori* state bounds from state bounds computed over the course of an algorithm. Therefore, we denote *a priori* state bounds as $\tilde{X}_{j,0}$. Let the set of solutions of (7.1.1) from time $t_0 = t_j$ to $t_f = t_v$ with $\mathbf{x}_0(\mathbf{p}) = \mathbf{x}_j(\mathbf{p})$ be denoted by $\check{\mathbf{x}}(\mathbf{x}_j, \mathbf{p}, t_j, t_v)$.

For the remainder of the chapter, we make the following Assumption 7.2.1 which ensures that the pODEs problem (7.1.1), and relaxations of the right-hand side function $\mathbf{f}$ and initial value function $\mathbf{x}_0$, are all well-posed.

Assumption 7.2.1. The system of pODEs (7.1.1) satisfies the following conditions:

1. $\mathbf{x}_0 : P \to D$ is locally Lipschitz continuous on P, and

2. $\mathbf{f}$ is continuously differentiable on $D \times \Pi$,

A solution of (7.1.1) is any continuous $\mathbf{x} : P \times I \to D$ such that, for every $\mathbf{p} \in P$, $\mathbf{x}(\mathbf{p}, \cdot) : I \to D$ is continuously differentiable and satisfies (7.1.1) on I. Furthermore, it is assumed that a unique solution exists over the time domain I for every $\mathbf{p} \in P$. Note that there is no loss of generality when restricting the analysis to the autonomous case, as the independent variable t may be treated as an additional dependent variable $x_{n_x+1}(\mathbf{p}, t) = t$ appended to the solution vector $\mathbf{x}(\mathbf{p}, t)$ of (7.1.1) with $\dot{x}_{n_x+1}(\mathbf{p}, t) = 1$ and $x_{n_x+1}(\mathbf{p}, 0) = t_0$. The i-th Taylor coefficient of the solution of (7.1.1) with respect to t, denoted $\mathbf{f}^{[i]}$, is defined by the sequence of functions

$$\begin{aligned}
\mathbf{f}^{[0]}(\mathbf{x}(\mathbf{p}, t), \mathbf{p}) &= \mathbf{x}(\mathbf{p}, t) \\
\mathbf{f}^{[i]}(\mathbf{x}(\mathbf{p}, t), \mathbf{p}) &= \frac{1}{i}\mathbf{f}(\mathbf{x}(\mathbf{p}, t), \mathbf{p})\frac{\partial \mathbf{f}^{[i-1]}}{\partial \mathbf{x}}(\mathbf{x}(\mathbf{p}, t), \mathbf{p}), \qquad i \geq 1
\end{aligned}$$

for which

$$\mathbf{x}(\mathbf{p}, t + h) = \sum_{i=0}^{k-1} h^i \mathbf{f}^{[i]}(\mathbf{x}(\mathbf{p}, t), \mathbf{p}) + h^k \mathbf{f}^{[k]}(\mathbf{x}(\mathbf{p}, t + \tau), \mathbf{p}), \ \tau \in [0, h]$$

holds. Moreover, we note that the i-th Taylor coefficient of $\mathbf{f}$ and $(i + 1)$-th Taylor coefficient of $\mathbf{x}(\mathbf{p}, t)$ are related by the simple formula

$$\mathbf{x}^{[i]}(\mathbf{p}, t) = \mathbf{f}^{[i-1]}(\mathbf{x}(\mathbf{p}, t), \mathbf{p}).$$

The focus of this chapter is on computing rigorous under/overestimators of solutions of the

pODEs system, referred to as *state bounds* and *state relaxations* formalized in Definitions 7.2.1 and 7.2.2.

Definition 7.2.1 (State Bounds [270])**.** Continuous functions $\mathbf{v}, \mathbf{w} : I \to \mathbb{R}^{n_x}$ are called *state bounds* for (7.1.1) on $P \times I$ if $\mathbf{v}(t) \leq \mathbf{x}(\mathbf{p}, t) \leq \mathbf{w}(t)$, $\forall (\mathbf{p}, t) \in P \times I$.

Definition 7.2.2 (State Relaxations)**.** Continuous functions $\mathbf{x}^{cv}, \mathbf{x}^{cc} : I \to \mathbb{R}^{n_x}$ are called *state relaxations* for (7.1.1) on $P \times I$ if $\mathbf{x}^{cv}(\mathbf{p}, t) \leq \mathbf{x}(\mathbf{p}, t) \leq \mathbf{x}^{cc}(\mathbf{p}, t)$, $\forall (\mathbf{p}, t) \in P \times I$ provided that $\mathbf{x}^{cv}$, $\mathbf{x}^{cc}$ are convex and concave on $\mathbf{p} \in P$, $\forall t \in I$, respectively. For compactness, *state relaxations* may be represented as the tuple $\{\mathbf{x}^{cv}, \mathbf{x}^{cc}\}(\mathbf{p}, t)$.

7.2.2 Existence and Uniqueness

Discretize-then-relax methods for computing interval bounds and relaxations of the solutions of (7.1.1) are predicated on first computing a stepsize h_j and an *a priori* enclosure $\tilde{X}_j$ that contains all solutions of (7.1.1) for all $\mathbf{p} \in P \in \mathbb{IR}^{n_p}$ over the time interval $[t_j, t_j + h_j]$. We denote the range of the solution $\{\mathbf{x}(\mathbf{p}, t) \mid t_j \leq t \leq t_{j+1}\}$ for a particular $\mathbf{p} \in P$ as $\mathbf{x}(\mathbf{p}, t_j)$ These enclosures are typically computed by adaptively applying an existence and uniqueness test to establish an adequate stepsize over which an *a priori* enclosure of the parametric solution of (7.1.1) can be guaranteed for all $\mathbf{p} \in P$. In this subsection, we trace the existing developments of methods for generating these *a priori* enclosures. These begin with the computation of an *a priori* enclosure through the application of (7.2.2), as a direct result of the following Theorem 7.2.3.

Theorem 7.2.3. (Corollary 2 [256]) Let $\tilde{X}_j^0 \in \mathbb{I}D$, and let $\hat{\mathbf{x}}_j \in \text{int}(\tilde{X}_j^0)$ with $P \in \mathbb{IR}^{n_p}$. If

$$\left\{ \sum_{i=0}^{k-1} (t - t_j)^i \mathbf{f}^{[i]}(\hat{\mathbf{x}}_j, \mathbf{p}) + (t - t_j)^k F^{[k]}(\tilde{X}_j^0, \mathbf{p}) : t_j \leq t \leq t_{j+1}, \mathbf{p} \in P \right\} \subset \tilde{X}_j^0, \tag{7.2.1}$$

then

$$\check{\mathbf{x}}(\mathbf{x}_j(\mathbf{p}), \mathbf{p}, t_j, t_{j+1}) \in \sum_{i=0}^{k-1} (t - t_j)^i \mathbf{f}^{[i]}(\hat{\mathbf{x}}_j, \mathbf{p}) + (t - t_j)^k F^{[k]}(\tilde{X}_j^0, \mathbf{p}), \tag{7.2.2}$$

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

Accordingly, an *a priori* enclosure may then be computed as follows:

$$\tilde{X}_j = \sum_{i=0}^{k-1} [0, h_j]^i F^{[i]}(X_j, P) + [0, h_j]^k F^{[k]}(\tilde{X}_j^0, P),$$

provided that $\tilde{X}_j^0$ and $h_j = t_{j+1} - t_j$ satisfy (7.2.1) for each $\hat{\mathbf{x}}_j \in X_j$. In the context of enclosing parametric solutions of (7.1.1), satisfaction of this condition is determined iteratively by progressively evaluating potential step sizes.

Theorem 7.2.4. (Adapted from Theorem 4 and Corollary 5 of [256]) Let $\tilde{X}_j^0 = [\tilde{\mathbf{x}}_j^{L,0}, \tilde{\mathbf{x}}_j^{U,0}] \subset D$, let $X_j \subset \text{int}(\tilde{X}_j^0)$, and let the functions $\mathbf{x}_j^{cv}, \mathbf{x}_j^{cc} : P \to X_j$ be, respectively, convex and concave relaxations of $\mathbf{x}_j$ on P. Suppose that functions $\mathbf{f}^{[i],cv}, \mathbf{f}^{[i],cc} : D \times D \times P \to \mathbb{R}^{n_x}$, $i = 0, \ldots, k-1$, 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 P and

$$\mathbf{f}^{[i],cv}(\mathbf{x}_j^{cv}(\mathbf{p}), \mathbf{x}_j^{cc}(\mathbf{p}), \mathbf{p}) \leq \mathbf{f}^{[i]}(\mathbf{x}_j(\mathbf{p}), \mathbf{p}) \leq \mathbf{f}^{[i],cc}(\mathbf{x}_j^{cv}(\mathbf{p}), \mathbf{x}_j^{cc}(\mathbf{p}), \mathbf{p}),$$
$$\forall \mathbf{x}_j \in [\mathbf{x}_j^{cv}, \mathbf{x}_j^{cc}](\mathbf{p}), \forall \mathbf{p} \in P.$$

Furthermore, suppose that interval bounds of $\mathbf{f}^{[i]}$ on $X_j \times P$ are denoted by $F^{[i]} = [\mathbf{f}^{[i],L}, \mathbf{f}^{[i],U}]$ for $i = 1, \ldots, k$ and are available. Lastly, suppose that $\tilde{F}^{[k]} = [\tilde{\mathbf{f}}^{[k],L}, \tilde{\mathbf{f}}^{[k],U}] \supset F(\tilde{X}_j^0, P)$ are also

available. If

$$\left\{ \sum_{i=0}^{k-1} (t-t_j)^i F^{[i]} + (t-t_j)^k \tilde{F}^{[k]} : t_j \le t \le t_{j+1} \right\} \subset \tilde{X}_j^0, \tag{7.2.3}$$

then

1. $\tilde{\mathbf{x}}_j^{cv,1}(\mathbf{p},t) \le \mathbf{x}_j(\mathbf{p},t) \le \tilde{\mathbf{x}}_j^{cc,1}(\mathbf{p},t)$ for all $\mathbf{x}_j \in [\mathbf{x}_j^{cv}, \mathbf{x}_j^{cc}](\mathbf{p})$, all $\mathbf{p} \in P$ and all $t \in [t_j, t_{j+1}]$
2. the functions $\tilde{\mathbf{x}}_j^{cv,1}(\cdot,t)$ and $\tilde{\mathbf{x}}_j^{cc,1}(\cdot,t)$, are, respectively, convex and concave relaxations of $\mathbf{x}_j(\cdot,t)$ on P for each $t \in [t_j, t_{j+1}]$,

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

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

State relaxations $\{\tilde{\mathbf{x}}_j^{cv}, \tilde{\mathbf{x}}_j^{cc}\}(\mathbf{p})$ at time t_j can then be computed as

$$\begin{aligned}
\{\tilde{\mathbf{x}}_j^{cv}, \tilde{\mathbf{x}}_j^{cc}\}(\mathbf{p}) = &\sum_{i=0}^{k-1} [0,h]^i \times \{\mathbf{f}^{[i],cv}, \mathbf{f}^{[i],cc}\}(\mathbf{x}_j^{cv}(\mathbf{p}), \mathbf{x}_j^{cc}(\mathbf{p}), \mathbf{p}) \\
&+ [0,h]^k F^{[k]}(\tilde{X}_j^0, P).
\end{aligned} \tag{7.2.4}$$

In this chapter, a local excess-per-unit-step (LEPUS) approach is used that determines a stepsize based on user-specified error tolerances [215]. After this step, the interval bounds or relaxations may be tightened using a second stage. We leave a thorough discussion of this second stage for later sections, as it is the primary focus of this chapter.

7.2.3 Affine Refinement of Interval Bounds

It should be noted that when relaxations are computed, their associated subgradients can be used to construct interval bounds that are tighter than those computed by means of interval arithmetic [257]. These tighter interval bounds, in turn, may lead to less conservative relaxations computed through a generalized McCormick relaxation framework. This is achieved by taking a natural interval extension of the implied affine relaxations, an approach used in several applications [206, 257, 300]. This approach is summarized in Proposition 7.2.5.

Proposition 7.2.5. Let $Z \in \mathbb{IR}^n$ be nonempty and let $f^{cv}, f^{cc} : Z \to \mathbb{R}$ be convex and concave relaxations of $f : Z \to \mathbb{R}$, respectively. Let $\mathbf{s}_f^{cv}, \mathbf{s}_f^{cc} : Z \to \mathbb{R}^n$ be subgradients of f^{cv} and f^{cc} at $\bar{\mathbf{z}} \in Z$, respectively. The lower/upper affine relaxations of f, denoted $f^{a,l}, f^{a,u} : Z \to \mathbb{R}$, respectively, are given by:

$$f^{a,l}(\mathbf{z}) = f^{a,l}(\bar{\mathbf{z}}) + \mathbf{s}_f^{cv}(\bar{\mathbf{z}})^T(\mathbf{z} - \bar{\mathbf{z}}),$$
$$f^{a,u}(\mathbf{z}) = f^{a,l}(\bar{\mathbf{z}}) + \mathbf{s}_f^{cc}(\bar{\mathbf{z}})^T(\mathbf{z} - \bar{\mathbf{z}}).$$

Furthermore, the extrema of the affine relaxations bounding f over Z are given by

$$\min\{f^{a,l}(\boldsymbol{\theta}) : \boldsymbol{\theta} \in Z\} \leq f(\mathbf{z}) \leq \max\{f^{a,u}(\boldsymbol{\theta}) : \boldsymbol{\theta} \in Z\}.$$

The values of the extrema can be computed by direct application of interval arithmetic. Furthermore, the functions $f^{a,l}$ and $f^{a,u}$ attain their minima and maxima on Z, at $\mathbf{z}^{a,l}$ and $\mathbf{z}^{a,u}$, given by:

$$z_i^{a,l} = \begin{cases} z_i^L & \text{if } \; s_{f,i}^{cv}(\bar{\mathbf{z}}) \geq 0, \\ z_i^U & \text{otherwise,} \end{cases} \qquad z_i^{a,u} = \begin{cases} z_i^L & \text{if } \; s_{f,i}^{cc}(\bar{\mathbf{z}}) \leq 0, \\ z_i^U & \text{otherwise,} \end{cases}$$

for $i = 1, \ldots, n$.

7.3 Parametric Implicit Linear Multistep Methods

In this section, we build on two distinct groups of recent research contributions. We generalize the recent work of Marciniak et al [173, 174], which details the direct application to parametric interval methods to address parametric forms of the Adams-Moulton linear multistep method. Moreover, we build on the work of [328] which details a methodology by which convex and convex relaxation of the numerical solutions of pODEs obtained using implicit linear multistep methods may be constructed. Parametric interval methods are developed to bound these forms for all $\mathbf{p} \in P$. In order to construct these methods, two key pieces of information are required: the bounds on the approximate solution by the underlying numerical method and the remainder on the solution. The approximate solution is specified by evaluating the system of algebraic equations arising from the base numerical method. In general, this remainder bound can be formed by computing bounds of the higher-order term in the series approximation used to derive the corresponding numerical method.

We proceed by introducing a parametric interval version of the Adams-Moulton method and derive a corresponding mean-valued representation of this method. Next, we will use this mean-valued form to outline both fixed-point and semi-explicit approaches. Lastly, we will generalize this to compute convex/concave relaxations of $\mathbf{x}(\cdot\,, t)$ on P for each $t \in I$.

7.3.1 Mean-Value Form of Parametric Adams-Moulton Methods

Parametric implicit linear multistep methods of order n—also referred to as n-step methods—require that the values $\mathbf{x}(\mathbf{p}, t_{j-1}), \ldots, \mathbf{x}(\mathbf{p}, t_{j-n})$ are known (numerically approximated) at the j^{th} time step, and these values are used to compute $\mathbf{x}(\mathbf{p}, t_j)$. Interval methods require that enclosures of $\mathbf{x}(\mathbf{p}, t_j), \mathbf{x}(\mathbf{p}, t_{j-1}), \ldots, \mathbf{x}(\mathbf{p}, t_{j-n})$ are known during each step j, denoted as X_j, $X_{j-1}, \ldots X_{j-n}$, respectively. The most natural way to attain bounds X_j lies in the use of *a priori* bounds computed using a Picard existence and uniqueness test [87] or by making use of the higher-order existence (HOE) test [65, 214, 256] originating from Theorem 7.2.3, while $X_{j-1}, \ldots, X_{j-n}$ are the values computed during previous time steps. Alternatively, a different Phase 2 contractor, such as Lohner's method [163], or the Interval Hermite-Obreschkoff method [212], can be applied first to refine *a priori* bounds of X_j.

The parametric interval Adams-Moulton method only requires derivative information, namely, the Jacobian of the right-hand side function $\mathbf{f}$, whereas Lohner's method [163] at $t = t_j$ and the Interval Hermite-Obreschkoff method [212] require derivatives of Taylor coefficients. This can substantially reduce the CPU time spent computing the Jacobian of $\mathbf{f}$ with respect to $\mathbf{x}$. For an expression that involves N floating point operations per component of $\mathbf{f}$, the computational complexity of evaluating the Jacobian of $\mathbf{f}$ with respect to $\mathbf{x}$ is $O(Nn_x^2)$. This contrasts the time complexity associated with computing the Jacobian of the q^{th} Taylor series coefficient, which has a complexity between $O(qNn_x^2)$ and $O(q(q+1)Nn_x^2)$ due to the recursive nature of the calculation [214]. We begin by detailing the fixed stepsize Adams-Moulton method.

Definition 7.3.1 (Fixed Stepsize Parametric Adams-Moulton Method [173])**.** Let $n \in \mathbb{Z}^+$. The n-step fixed stepsize parametric Adams-Moulton method is

$$\mathbf{x}(\mathbf{p}, t_j) = \mathbf{x}(\mathbf{p}, t_{j-1}) + h \sum_{k=0}^{n} \bar{\beta}_k \mathbf{f}(\mathbf{x}(\mathbf{p}, t_{j-k}), \mathbf{p}),$$

where $t_{j-n}, \ldots, t_j \in [t_{j-n}, t_j]$ such that $t_i - t_{i-1} = h$ for $i = (j - n + 1), \ldots, j$ and

$$\bar{\beta}_k = (-1)^k \sum_{m=k}^{n} \binom{m}{k} \bar{\gamma}_m \tag{7.3.1}$$

such that

$$\bar{\gamma}_0 = 1, \qquad \bar{\gamma}_i = \frac{1}{i!} \int_{-1}^{0} \prod_{i=1}^{n+1} (w + i - 1) dw$$

The truncation error τ_n associated with this n-step method is then given by

$$\begin{aligned} \tau_n :=& h^{n+2} \bar{\gamma}_{n+1} \mathbf{f}^{[n+1]}(\mathbf{x}(\mathbf{p}, \bar{\nu}), \mathbf{p}) \\ =& h^{n+2} \bar{\gamma}_{n+1} \mathbf{x}^{[n+2]}(\mathbf{p}, \bar{\nu}), \end{aligned}$$

where $\bar{\nu} \in [t_{j-n}, t_j]$.

A parametric interval version of the Adams-Moulton method can then be expressed as follows. Let X_j be interval bounds of $\mathbf{x}(\mathbf{p}, t_j)$ on P. Then

$$X_j := X_{j-1} + h \sum_{k=0}^{n} \bar{\beta}_k F(X_j, P) + h^{n+2} \bar{\gamma}_{n+1} F^{[n+1]}(\tilde{X}_{j-k}, P) \tag{7.3.2}$$

defines an update. However, this version is necessarily expansive with $\mathrm{w}(X_j) \geq \mathrm{w}(X_{j-1})$. To mitigate this, we subsequently develop potentially contractive methods based on a mean-value

expansion of form (7.3.2). Let $(\hat{x}_j, \ldots, \hat{x}_{j-n}, \mathbf{p}) \in X_j \times \ldots \times X_{j-n} \times P$, and $\boldsymbol{\mu}_j : P \to X_j$ and $\boldsymbol{\rho} : P \to P$ are given by (7.3.3) and (7.3.4), respectively with $\eta \in [0, 1]$.

$$\boldsymbol{\mu}_j(\mathbf{p}) = \eta(\mathbf{x}_j(\mathbf{p}) - \hat{\mathbf{x}}_j) + \hat{\mathbf{x}}_j, \tag{7.3.3}$$

$$\boldsymbol{\rho}(\mathbf{p}) = \eta(\mathbf{p} - \hat{\mathbf{p}}) + \hat{\mathbf{p}}. \tag{7.3.4}$$

Furthermore, let $\mathbf{x}(\mathbf{p}, \bar{\nu})$ where $\bar{\nu} \in [t_{j-n}, t_j]$ be a point in X_j such that

$$\mathbf{x}(\mathbf{p}, t_j) = \mathbf{x}(\mathbf{p}, t_{j-1}) + h \sum_{k=0}^{n} \bar{\beta}_k \mathbf{f}(\mathbf{x}(\mathbf{p}, t_{j-k}), \mathbf{p}) + h^{n+2} \bar{\gamma}_{n+1} \bar{\psi}(\mathbf{x}(\mathbf{p}, \bar{\nu}), \mathbf{p}), \tag{7.3.5}$$

holds exactly, satisfying the intermediate value theorem. Further, we define the terms

$$\begin{aligned}
\mathbf{J}_j^x(\mathbf{x}_j(\mathbf{p}), \mathbf{p}) &= h\bar{\beta}_j \frac{\partial \mathbf{f}}{\partial \mathbf{x}}(\boldsymbol{\mu}_j(\mathbf{p}), \boldsymbol{\rho}(\mathbf{p})) \\
\mathbf{J}_j^p(\mathbf{x}_j(\mathbf{p}), \mathbf{p}) &= h \sum_{k=j}^{j-n} \bar{\beta}_k \frac{\partial \mathbf{f}}{\partial \mathbf{p}}(\boldsymbol{\mu}_k(\mathbf{p}), \boldsymbol{\rho}(\mathbf{p})) \\
\mathbf{r}_j(\mathbf{x}_j(\mathbf{p}), \mathbf{p}) &= h^{n+2} \bar{\gamma}_{n+1} \mathbf{f}^{[n+1]}(\mathbf{x}_j(\mathbf{p}), \mathbf{p}) \\
\mathbf{d}_j(\mathbf{p}) &= \hat{\mathbf{x}}_{j-1} + h \sum_{k=0}^{n} \bar{\beta}_k \mathbf{f}(\hat{\mathbf{x}}_{j-k}, \hat{\mathbf{p}}) + \mathbf{r}_j(\mathbf{x}(\mathbf{p}, \bar{\nu}), \mathbf{p}).
\end{aligned}$$

The mean-value form of (7.3.5) is then given by:

$$\begin{aligned}
\mathbf{x}_j(\mathbf{p}) = \mathbf{d}_j(\mathbf{p}) &+ \mathbf{J}_j^p(\mathbf{x}_j(\mathbf{p}), \mathbf{p})(\mathbf{p} - \hat{\mathbf{p}}) \\
&+ (\mathbf{I} + \mathbf{J}_{j-1}^x(\mathbf{x}_{j-1}(\mathbf{p}), \mathbf{p}))(\mathbf{x}_{j-1}(\mathbf{p}) - \hat{\mathbf{x}}_{j-1}) \\
&+ \sum_{\substack{k=0 \\ k \neq 1}}^{n} \mathbf{J}_{j-k}^x(\mathbf{x}_{j-k}(\mathbf{p}), \mathbf{p})(\mathbf{x}_{j-k}(\mathbf{p}) - \hat{\mathbf{x}}_{j-k}).
\end{aligned} \tag{7.3.6}$$

Before we proceed, we illustrate how the development of a variable stepsize method closely

parallels the fixed stepsize method.

Definition 7.3.2 (Variable-Step Parametric Adams-Moulton Method [173])**.** The n-step variable stepsize implicit Adams-Moulton method is given by

$$\mathbf{x}(\mathbf{p}, t_j) = \mathbf{x}(\mathbf{p}, t_{j-1}) + h_j \sum_{k=0}^{n} \hat{\beta}_{kj} \mathbf{f}(\mathbf{x}(\mathbf{p}, t_{j-k}), \mathbf{p}),$$

where $t_{j-n}, \ldots, t_j \in [t_{j-n}, t_j]$ such that $h_i = t_i - t_{i-1}$ for $i = (j - n + 1), \ldots, j$, and

$$\sum_{k=0}^{n} \hat{\beta}_{kj} \mathbf{f}(\mathbf{x}(\mathbf{p}, t_{j-k}), \mathbf{p}) = \sum_{k=0}^{n} g_k(j) \boldsymbol{\phi}_k(j) \tag{7.3.7}$$

where $g_k(j)$ and $\boldsymbol{\phi}_k(j)$ are terms defined by a sequence of calculations discussed below, and the truncation error associated with this n-step method is then given by

$$\begin{aligned} h_j^{n+2} \hat{\beta}_{nj} \bar{\psi}(\mathbf{x}(\mathbf{p}, \bar{\nu}), \mathbf{p}) &= h_j^{n+2} \hat{\beta}_{nj} \mathbf{f}^{[n+1]}(\mathbf{x}(\mathbf{p}, \bar{\nu}), \mathbf{p}) \\ &= h_j^{n+2} \hat{\beta}_{nj} \mathbf{x}^{[n+2]}(\mathbf{p}, \bar{\nu}), \end{aligned}$$

where $\bar{\nu} \in [t_{j-n}, t_j]$.

The coefficients $\hat{\beta}_{kj}$ could then be computed by the method of divided differences given by Krogh [157]. However, we wish to avoid including prior evaluations of $\mathbf{f}$ and the introduction of subtraction in the computation, which may yield potentially expansive forms. We implement a symbolic computation of (7.3.7) by associating each $\mathbf{f}(\mathbf{x}(\mathbf{p}, t_{j-1}), \mathbf{p})$ for $j = 1, \ldots, n$ with a distinct unit vector. The addition is performed in a vector fashion, and multiplication is interpreted as scalar multiplication of a vector. The resulting vectors are summed according to (7.3.7) resulting in the vector of coefficients $\hat{\beta}_{kj}$ for $k = 0, 1, \ldots, n$. The following relationships define the recursive formulas needed to compute the $\hat{\beta}_{kj}$ terms using this approach. The values $g_k(j)$ and $\boldsymbol{\phi}_k(j)$ for

$k = 0, 1, \ldots, n-1$ are computed from

$$\begin{aligned}
\boldsymbol{\phi}_0(j) &= \bar{\boldsymbol{\phi}}_0(j) = \mathbf{f}(\mathbf{x}(\mathbf{p}, t_{j-1}), \mathbf{p}) \\
\bar{\boldsymbol{\phi}}_k(j) &= \bar{\boldsymbol{\phi}}_{k-1}(j) - \boldsymbol{\phi}_{k-1}(j-1) \\
\boldsymbol{\phi}_k(j) &= \beta_k(j)\bar{\boldsymbol{\phi}}_k(j), \qquad k = 1, \ldots, n-1.
\end{aligned}$$

The coefficients $\beta_k(j)$ are determined by

$$\beta_k(j) = \prod_{i=0}^{k-1} \frac{t_j - t_{j-i-1}}{t_{j-1} - t_{j-i-2}}.$$

Furthermore, the calculations of $g_k(j)$ are performed in the following fashion

$$g_k(j) = c_{k,1}(t_j), \qquad j = 1, 2, \ldots, n-1$$

where $c_{k,q}(t_j)$ is defined recursively as follows:

$$\begin{aligned}
c_{0,q}(t_j) &= \frac{1}{q}, \qquad c_{1,q}(t_j) = \frac{1}{q(q+1)}, \\
c_{k,q}(t_j) &= c_{k-1,q}(t_j) - c_{k-1,q+1}(t_j)\frac{h_j}{t_j - t_{j-k}}.
\end{aligned}$$

Note that this derivation of the variable stepsize parametric implicit linear multistep method finds its theoretical foundation in Newton-type interpolation of a polynomial and, as a consequence, is quite similar to the full-space collocation approach employed in dynamic optimization [16].

Remark 7.3.3. Note that when applying the variable stepsize Adams-Moulton method, the coefficients $\hat{\beta}_{kj}$ for $k = 1, \ldots, n$ depend only on the time values $t_j, \ldots, t_{j-n}$. Thus, the fixed stepsize

and variable stepsize versions vary only with respect to the value of these constants. As such, we subsequently use $\bar{\beta}_k$ for all $k = 1, \ldots, n$ to denote the coefficients throughout this chapter for simplicity, noting that they can easily be replaced with the coefficients $\hat{\beta}_{kj}$ to implement the variable stepsize method.

7.3.2 Interval Parametric Implicit Linear Multistep Methods

First, we note that $\mathbf{x}_{j-k}$ for $k = 1, \ldots, n$ are simply functions of $\mathbf{p} \in P$, and as a result, we may apply a parametric interval method to these functions on $X_j \times P$ in a block sequential manner similar to the approach used in [328]. Noting that $\mathbf{x}_{j-1}$ and $\mathbf{r}_j$ can be considered strictly functions of $\mathbf{p}$ (and not $\mathbf{x}_j$), (7.3.8) can define the implicit function $\mathbf{x}_j : P \to X_j$. Assuming that $\mathbf{x}(\mathbf{p}, \bar{\nu})$ is determined *a priori*, then $\mathbf{r}_j(\mathbf{p})$ is only a function of $\mathbf{p}$. The residual, $\mathfrak{r}_j$, is defined as:

$$\begin{aligned}\mathfrak{r}_j(\mathbf{x}_j(\mathbf{p}), \mathbf{p}) =& \mathbf{J}_j^p(\mathbf{x}_j(\mathbf{p}), \mathbf{p})(\mathbf{p} - \hat{\mathbf{p}}) + \mathbf{d}_j(\mathbf{p}) \\ &+ (\mathbf{I} + \mathbf{J}_{j-1}^x(\mathbf{x}_{j-1}(\mathbf{p}), \mathbf{p}))(\mathbf{x}_{j-1}(\mathbf{p}) - \hat{\mathbf{x}}_{j-1}) \\ &+ \sum_{\substack{k=0 \\ k\neq 1}}^{n} \mathbf{J}_{j-k}^x(\mathbf{x}_{j-k}(\mathbf{p}), \mathbf{p})(\mathbf{x}_{j-k}(\mathbf{p}) - \hat{\mathbf{x}}_{j-k}) - \mathbf{x}_j(\mathbf{p}),\end{aligned} \tag{7.3.8}$$

Equation (7.3.8) may be used directly with a parametric interval method [296] such that $\tilde{\mathbf{x}}(\mathbf{p}, \bar{\nu}) \in \tilde{X}(t_{j-n}; t_j) := \bigcup_{i=0}^{n} \tilde{X}_{j-n+i}$. However, this may lead to expansiveness. An interval extension of $\mathbf{r}_j$ is computed according to

$$R_j(P) = h_j^{n+2} \bar{\gamma}_{n+1} \bigcup_{i=0}^{n} F^{[n+1]}(\tilde{X}_{j-n+i}, P), \tag{7.3.9}$$

which is at least as tight as $h_j^{n+2} \bar{\gamma}_{n+1} F^{[n+1]}(\tilde{X}(t_{j-n}; t_j), P)$. Note that the term $F^{[n+1]}(\tilde{X}_{k-n+i}, P)$ is computed during the first phase of the algorithm. Provided that this value is saved after each time

step, only a single evaluation of F is required and is therefore efficient. Moreover, interval bounds may be computed via a parametric interval method. However, $\mathbf{J}_j^x(\mathbf{x}_j(\mathbf{p}), \mathbf{p})(\mathbf{x}_j(\mathbf{p}) - \hat{\mathbf{x}}_j)$ can contribute significantly to overestimation and result in weaker bounds. As an alternative, we can propagate the uncertainty set as a parallelepiped in the manner of Lohner [164]. That is, we assume that there exists a real-valued matrix $\mathbf{A}_j \in \mathbb{R}^{n_x \times n_x}$ and $\boldsymbol{\delta}_j(\mathbf{p}) \in \boldsymbol{\Delta}_j$ and $\mathbf{x}_j(\mathbf{p}) - \hat{\mathbf{x}}_j = \mathbf{A}_j \boldsymbol{\delta}_j(\mathbf{p})$ form an alternative representation of (7.3.8):

$$\begin{aligned}\mathfrak{r}_j(\mathbf{x}_j(\mathbf{p}), \mathbf{p}) =& \mathbf{J}_j^p(\mathbf{x}_j(\mathbf{p}), \mathbf{p})(\mathbf{p} - \hat{\mathbf{p}}) + (\mathbf{I} + \mathbf{J}_{j-1}^x)(\mathbf{x}_{j-1}(\mathbf{p}), \mathbf{p})\mathbf{A}_{j-1}\boldsymbol{\delta}_{j-1}(\mathbf{p}) \\ &+ D_j(\mathbf{p}) + \sum_{k=2}^{n} \mathbf{J}_{j-k}^x(\mathbf{x}_{j-k}(\mathbf{p}), \mathbf{p})\mathbf{A}_{j-k}\boldsymbol{\delta}_{j-k}(\mathbf{p}) \\ &+ \mathbf{J}_j^x(\mathbf{x}_j(\mathbf{p}), \mathbf{p})(\mathbf{x}_j(\mathbf{p}) - \hat{\mathbf{x}}_j) - \mathbf{x}_j(\mathbf{p}). \end{aligned} \tag{7.3.10}$$

Alternatively, we can make use of a semi-explicit update:

$$\begin{aligned}X_j :=& \Big(J_j^p(X_j, P)(P - \hat{\mathbf{p}}) + (\mathbf{I} + J_{j-1}^x(X_{j-1}, P))\mathbf{A}_{j-1}\boldsymbol{\Delta}_{j-1} \\ &+ D_j(P) + \sum_{k=2}^{n} J_{j-k}^x(X_{j-k}, P)\mathbf{A}_{j-k}\boldsymbol{\Delta}_{j-k} \\ &+ J_j^x(X_j^0, P)(X_j^0 - \hat{\mathbf{x}}_j^0) \Big) \cap X_j^0, \end{aligned} \tag{7.3.11}$$

where X_j^0 is an *a priori* interval bound of $\mathbf{x}(\mathbf{p}, t_j)$ on P that may be available as the result of the Phase 1 method and $\mathbf{x}_j^0 \in X_j^0$. The approach is initialized with $\boldsymbol{\Delta}_0 = X_0 - \hat{\mathbf{x}}_0$ and $\mathbf{A}_0 = \mathbf{I}$. The matrix $\mathbf{A}_j$ is subsequently determined by taking the orthogonal matrix $\mathbf{Q}$ obtained from the QR

decomposition of $m(\mathbf{J}_j^x \mathbf{A}_{j-1})$ and $\mathbf{\Delta}_j$ is determined by the following update

$$\begin{aligned}\mathbf{\Delta}_j =& \mathbf{A}_j^{-1}\Big(J_j^p(X_j, P)(P - \hat{\mathbf{p}}) + (\mathbf{I} + J_{j-1}^x(X_{j-1}, P))\mathbf{A}_{j-1}\mathbf{\Delta}_{j-1} \\ &+ \mathbf{d}_j(P) + \sum_{k=2}^{n} J_{j-k}^x(X_{j-k}, P)\mathbf{A}_{j-k}\mathbf{\Delta}_{j-k} \\ &+ J_j^x(X_j^0, P)(X_j - \hat{\mathbf{x}}_j) - \hat{\mathbf{x}}_j\Big),\end{aligned}$$

which simplifies to

$$\begin{aligned}\mathbf{\Delta}_j =& \mathbf{A}_j^{-1} J_j^p(P)(P - \hat{\mathbf{p}}) + \mathbf{A}_j^{-1}(\mathbf{I} + J_{j-1}^x(X_{j-1}, P))\mathbf{A}_{j-1}\mathbf{\Delta}_{j-1} \\ &+ \mathbf{A}_j^{-1}(\mathbf{d}_j(P) - \hat{\mathbf{x}}_j) + \sum_{k=2}^{n} \mathbf{A}_j^{-1} J_{j-k}^x(X_{j-k}, P)\mathbf{A}_{j-k}\mathbf{\Delta}_{j-k} \\ &+ \mathbf{A}_j^{-1} J_j^x(X_j^0, P)(X_j - \hat{\mathbf{x}}_j)\end{aligned}$$

Example 7.3.4 (A Scalar Parametric ODE-IVP)**.** Consider the simple scalar ODE-IVP presented in [256] with a single parameter:

$$\begin{aligned}&\dot{x}(p, t) = -x^2 + p, \quad t \in I = [0, 1], \ p \in P = [-1, 1] \\ &x_0(p) = 9, \quad p \in P.\end{aligned} \tag{7.3.12}$$

with initial state bounds specified as $x^L(t) = 0.1$ and $x^U(t) = 9$.

The primary challenges that arise from the method presented by Wilhelm et al. [328] relate to the detection of an appropriate stepsize over which a unique solution is assured for the nonlinear system corresponding to the implicit linear multistep discretization. While parametric interval methods may be used to confirm that a unique solution exists, these rely on prior knowledge of state bounds, and an initial overestimation of the state bounds supplied to the

	k-Step		
	2	3	4
Interval enclosure width at $t = 1$	2.72	4.54	5.20
CPU time (s) for interval bounds	0.002	0.0029	0.0035

TABLE 7.3.1: A comparison of the width of the intervals at final time and the CPU run time (s) required to calculate the state bounds for Example 7.3.4 using a k-step interval parametric implicit linear multistep (PILMS) method.

algorithm may lead to nonconvergence of the parametric interval method. For instance, using state bounds of $x^L(t) = -100$ and $x^U(t) = 100$ for $t \in I$ results in nonconvergence. While the method of Wilhelm et al. [328] yields less conservative bounds for Example 7.3.4, as illustrated in Figure 7.3.1 (left panel), the PILMS method presented in this chapter automatically and rigorously accounts for the truncation error, as well as existence and uniqueness concerns at each time step. When less conservative *a priori* state bounds are specified, as is the case for the the initial state bounds of $[0.1, 9]$ in (7.3.12), the explicit accounting of truncation error is the primary source of overestimation leading to more expansive, but rigorous interval bounds on the *true* solution set, not just the numerical solution set.

As seen in the right panel of Figure 7.3.1, the use of a higher-order implicit PILMS which correspond to increasingly accurate discretization schemes does not necessarily give rise to tighter interval bounds. As the truncation error term is computed on a larger time interval this may counter-balance the higher-order power that occurs in the highest order Taylor coefficient. Additionally, the presence of additional terms naturally leads to overestimation due to the classical dependency issue associated with interval arithmetic.

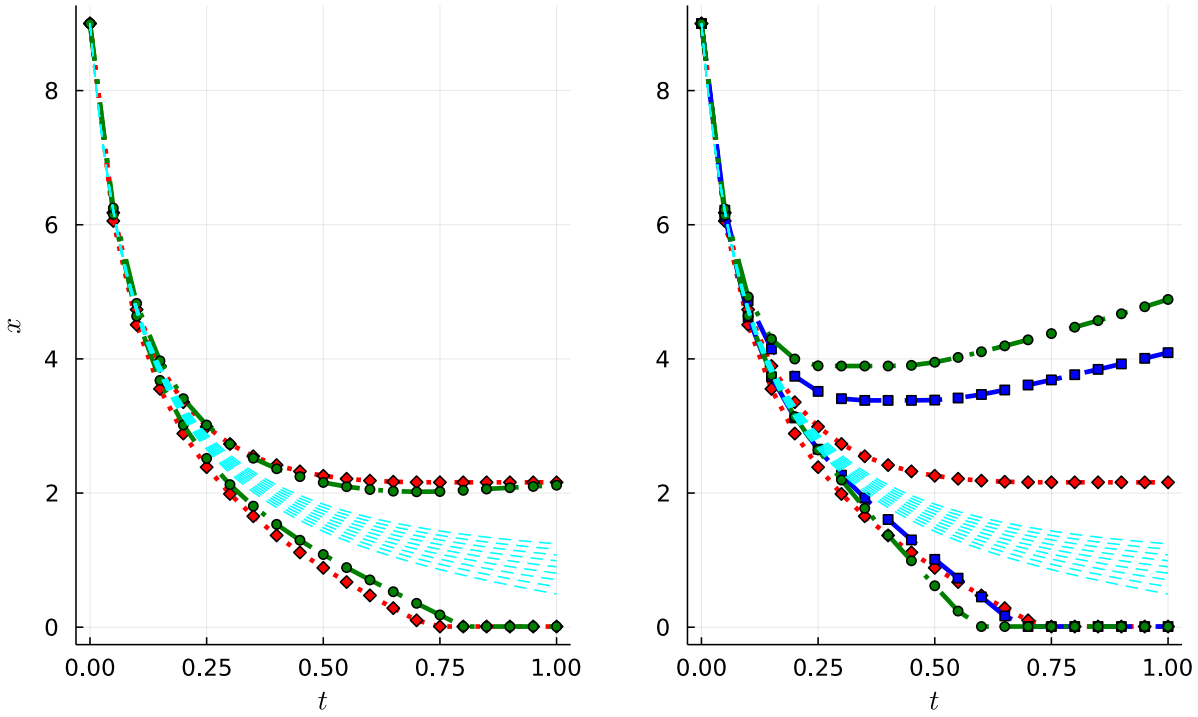

FIGURE 7.3.1: Local solution trajectories of (7.3.12) of Example 7.3.4 for several $p \in P$ are depicted in both plots for reference (dashed-cyan curves). State bounds are illustrated in each plot for k-step interval parametric implicit linear multistep (PILMS) methods along with the previous non-validated solution bounding method presented in [328]. **Left:** A 2-step PILMS method (red-diamond) is compared to the non-validated solution bounding method of [328] (green-circle). **Right:** A 2-step PILMS method (red-diamond) is compared with 3-step PILMS (blue-square) and a 4-step PILMS methods (green-circle).

7.3.3 Convex/Concave Relaxations of Implicit Linear Multistep Methods

In this subsection, we present a similar development for the construction of state relaxations and their corresponding subgradients. We begin by defining the following:

$$\begin{aligned}
\{\mathbf{J}_j^{x,cv}, \mathbf{J}_j^{x,cc}\}(\mathbf{p}) &= h_j\bar{\beta}_j\frac{\partial \mathbf{f}}{\partial \mathbf{x}}(\{\boldsymbol{\mu}_j^{cv}, \boldsymbol{\mu}_j^{cc}\}(\mathbf{p}), \{\boldsymbol{\rho}^{cv}, \boldsymbol{\rho}^{cc}\}(\mathbf{p}))\\
\{\mathbf{J}_j^{p,cv}, \mathbf{J}_j^{p,cc}\}(\mathbf{p}) &= h_j\sum_{k=j}^{j-n}\bar{\beta}_k\frac{\partial \mathbf{f}}{\partial \mathbf{p}}(\{\boldsymbol{\mu}_k^{cv}, \boldsymbol{\mu}_k^{cc}\}(\mathbf{p}), \{\boldsymbol{\rho}^{cv}, \boldsymbol{\rho}^{cc}\}(\mathbf{p}))\\
\{\mathbf{r}_j^{cv}, \mathbf{r}_j^{cc}\}(\mathbf{p}) &= \{\mathbf{f}_\cup^{[n],cv}, \mathbf{f}_\cup^{[n],cc}\}(\mathbf{p})\\
\{\mathbf{d}_j^{cv}, \mathbf{d}_j^{cc}\}(\mathbf{p}) &= h_j^{n+2}\bar{\gamma}_{n+1}\{\mathbf{r}_j^{cv}, \mathbf{r}_j^{cc}\}(\mathbf{p}).
\end{aligned}$$

In an analogous fashion, relaxations may be computed from the following sequence of updates:

$$\begin{aligned}
\{\mathbf{x}_j^{cv}, \mathbf{x}_j^{cc}\}(\mathbf{p}) =&\Big(\{\mathbf{J}_j^{p,cv}, \mathbf{J}_j^{p,cc}\}(\mathbf{p})(\mathbf{p}-\hat{\mathbf{p}})\\
&+ (\mathbf{I} + \mathbf{J}_{j-1}^{x}(\mathbf{p}))\mathbf{A}_{j-1}\{\boldsymbol{\Delta}_{j-1}^{cv}, \boldsymbol{\Delta}_{j-1}^{cc}\}(\mathbf{p})\\
&+ \sum_{k=2}^{n}\{\mathbf{J}_{j-k}^{x,cv}, \mathbf{J}_{j-k}^{x,cc}\}(\mathbf{p})\mathbf{A}_{j-k}\{\boldsymbol{\Delta}_{j-k}^{cv}, \boldsymbol{\Delta}_{j-k}^{cc}\}(\mathbf{p})\\
&+ \{\mathbf{J}_j^{x,cv}, \mathbf{J}_j^{x,cc}\}(\mathbf{p})(\{\mathbf{x}_j^{0,cv}, \mathbf{x}_j^{0,cc}\}(\mathbf{p}) - \hat{\mathbf{x}}_j^0)\\
&+ \{\mathbf{d}_j^{cv}, \mathbf{d}_j^{cc}\}(\mathbf{p})\Big) \cap \{\mathbf{x}_j^{0,cv}, \mathbf{x}_j^{0,cc}\}(\mathbf{p}),
\end{aligned} \tag{7.3.13}$$

using $\hat{\mathbf{x}}_j$ and $\hat{\mathbf{x}}_j^0$ as defined by the interval update given by (7.3.11), and the update used to compute the values of $\boldsymbol{\Delta}_j^{cv}(\cdot), \boldsymbol{\Delta}_j^{cc}(\cdot)$ are performed as follows:

$$\begin{aligned}\{\boldsymbol{\Delta}_j^{cv}, \boldsymbol{\Delta}_j^{cc}\}(\mathbf{p}) = &(\mathbf{A}_j^{-1}\{\mathbf{J}_j^{p,cv}, \mathbf{J}_j^{p,cc}\}(\mathbf{p}))(\mathbf{p} - \hat{\mathbf{p}}) \\ &+ (\mathbf{A}_j^{-1}(\mathbf{I} + \{\mathbf{J}_{j-1}^{x,cv}, \mathbf{J}_{j-1}^{x,cc}\}(\mathbf{p}))\mathbf{A}_{j-1})\{\boldsymbol{\Delta}_{j-1}^{cv}, \boldsymbol{\Delta}_{j-1}^{cc}\}(\mathbf{p}) \\ &+ \sum_{k=2}^{n} \mathbf{A}_j^{-1}((\{\mathbf{J}_{j-k}^{x,cv}, \mathbf{J}_{j-k}^{x,cc}\}(\mathbf{p})\mathbf{A}_{j-k}))\{\boldsymbol{\Delta}_{j-k}^{cv}, \boldsymbol{\Delta}_{j-k}^{cc}\}(\mathbf{p}) \\ &+ \mathbf{A}_j^{-1}(\{\mathbf{d}_j^{cv}, \mathbf{d}_j^{cc}\}(\mathbf{p}) - \hat{\mathbf{x}}_j) \\ &+ (\mathbf{A}_j^{-1}\{\mathbf{J}_j^{x,cv}, \mathbf{J}_j^{x,cc}\}(\mathbf{p}))(\{\mathbf{x}_j^{cv}, \mathbf{x}_j^{cc}\}(\mathbf{p}) - \hat{\mathbf{x}}_j). \end{aligned} \tag{7.3.14}$$

Theorem 7.3.1. Let $\tilde{\mathbf{x}}_{j-i}^{cv,0}, \tilde{\mathbf{x}}_{j-i}^{cc,0} : P \to D$ be convex and concave relaxations of $\mathbf{x}(\mathbf{p}, t)$ on $t \in [t_{j-i-1}, t_{j-i}]$ for $i = 0, \ldots, n-1$. Suppose that functions $\mathbf{f}^{[n],cv}, \mathbf{f}^{[n],cc} : D \times D \times P \to \mathbb{R}^{n_x}$, are available such that $\mathbf{f}^{[n],cv}(\tilde{\mathbf{x}}_j^{cv,0}(\cdot), \tilde{\mathbf{x}}_j^{cc,0}(\cdot), \cdot)$ and $\mathbf{f}^{[n],cc}(\tilde{\mathbf{x}}_j^{cv,0}(\cdot), \tilde{\mathbf{x}}_j^{cc,0}(\cdot), \cdot)$ are convex and concave on P, respectively, and

$$\mathbf{f}^{[n],cv}(\tilde{\mathbf{x}}_{j-i}^{cv,0}(\mathbf{p}), \tilde{\mathbf{x}}_{j-i}^{cc,0}(\mathbf{p}), \mathbf{p}) \leq \mathbf{f}^{[n]}(\mathbf{x}_{j-i}, \mathbf{p}) \leq \mathbf{f}^{[n],cc}(\tilde{\mathbf{x}}_{j-i}^{cv,0}(\mathbf{p}), \tilde{\mathbf{x}}_{j-i}^{cc,0}(\mathbf{p}), \mathbf{p}) \tag{7.3.15}$$

for all $\mathbf{x}_{j-i} \in [\tilde{\mathbf{x}}_{j-i}^{cv,0}, \tilde{\mathbf{x}}_{j-i}^{cc,0}](\mathbf{p})$, for $i = 0, \ldots, n-1$, and for all $\mathbf{p} \in P$. Let

$$\mathbf{f}_{\cup,u}^{[n]}(\mathbf{x}_{j-n}, \ldots, \mathbf{x}_j, \mathbf{p}) = \min\left\{\mathbf{f}^{[n]}(\mathbf{x}_{j-n}, \mathbf{p}), \ldots, \mathbf{f}^{[n]}(\mathbf{x}_j, \mathbf{p})\right\}, \tag{7.3.16}$$

$$\mathbf{f}_{\cup,o}^{[n]}(\mathbf{x}_{j-n}, \ldots, \mathbf{x}_j, \mathbf{p}) = \max\left\{\mathbf{f}^{[n]}(\mathbf{x}_{j-n}, \mathbf{p}), \ldots, \mathbf{f}^{[n]}(\mathbf{x}_j, \mathbf{p})\right\}, \tag{7.3.17}$$

then

$$\mathbf{f}_{\cup}^{[n],cv}(\mathbf{p}) = \mathbf{f}_{\cup,u}^{[n],cv}(\{\tilde{\mathbf{x}}_{j-n}^{cv,0}, \tilde{\mathbf{x}}_{j-n}^{cc,0}\}(\mathbf{p}), \ldots, \{\tilde{\mathbf{x}}_{j}^{cv,0}, \tilde{\mathbf{x}}_{j}^{cc,0}\}(\mathbf{p}), \mathbf{p})$$
$$\mathbf{f}_{\cup}^{[n],cc}(\mathbf{p}) = \mathbf{f}_{\cup,o}^{[n],cc}(\{\tilde{\mathbf{x}}_{j-n}^{cv,0}, \tilde{\mathbf{x}}_{j-n}^{cc,0}\}(\mathbf{p}), \ldots, \{\tilde{\mathbf{x}}_{j}^{cv,0}, \tilde{\mathbf{x}}_{j}^{cc,0}\}(\mathbf{p}), \mathbf{p})$$

where $\mathbf{f}_{\cup}^{[n],cv}, \mathbf{f}_{\cup}^{[n],cc}$ are convex/concave relaxations of $\mathbf{f}^{[n]}(\mathbf{x}(\mathbf{p}, t))$ at $\mathbf{p} \in P$ for all $t \in [t_{j-n}, t_j]$.

Remark 7.3.5. Theorem 7.3.1 provides a generalization of the union of interval bounds used in (7.3.9) to convex and concave relaxations $\mathbf{f}_{\cup}^{[n]}$ at $\mathbf{p} \in P$.

Remark 7.3.6. In practice, the terms (7.3.16) and (7.3.17) can be evaluated using the relation:

$$\mathbf{f}_{\cup,u}^{[n]}(\mathbf{p}) = \min\left\{\mathbf{f}^{[n]}(\mathbf{x}_{j-n}, \mathbf{p}), \min\left\{\ldots, \min\left\{\mathbf{f}^{[n]}(\mathbf{x}_{j-1}, \mathbf{p}), \mathbf{f}^{[n]}(\mathbf{x}_{j}, \mathbf{p})\right\}\ldots\right\}\right\},$$
$$\mathbf{f}_{\cup,o}^{[n]}(\mathbf{p}) = \max\left\{\mathbf{f}^{[n]}(\mathbf{x}_{j-n}, \mathbf{p}), \max\left\{\ldots, \max\left\{\mathbf{f}^{[n]}(\mathbf{x}_{j-1}, \mathbf{p}), \mathbf{f}^{[n]}(\mathbf{x}_{j}, \mathbf{p})\right\}\ldots\right\}\right\},$$

where relaxations of the binary min and max operators can be computed using the rules presented in [191, 310].

7.4 Parametric Hermite-Obreschkoff Method

An alternative implicit relaxation method finds its roots in the Hermite-Obreschkoff method [71, 132]. We now recount the derivation of the Hermite-Obreschkoff method attributed to Darboux [71] and Hermite and Borchardt [132] and extend the interval Hermite-Obreschkoff

method [212] to parametric analogs. First, we define the following quantities:

$$V_{r,q}(w) = \frac{w^q(s-1)^r}{(r+q)!},$$
$$c_i^{r,q} = \frac{q!(q+r-i)!}{(r+q)!(q-i)!},$$
$$\mathbf{g}_i(\mathbf{p}, w) = \frac{1}{i!}\frac{d^i\mathbf{g}}{dw^i}(\mathbf{p}, w),$$

where $r \geq 0$, $q \geq 0$, $0 \leq i \leq q$, and further suppose that $\mathbf{g}(\mathbf{p}, w)$ is $r+q+1$ times differentiable with respect to w for all $\mathbf{p} \in P$. The polynomial

$$\int_0^1 V_{r,q}(w)\frac{d^{r+q+1}\mathbf{g}}{dw^{r+q+1}}(\mathbf{p}, w)dw$$

is integrated by parts repeatedly until we arrive at the following relation:

$$(-1)^{r+q}\int_0^1 V_{r,q}(w)\frac{d^{r+q+1}\mathbf{g}}{dw^{r+q+1}}(\mathbf{p}, w)dw$$
$$= \sum_{i=0}^{q}(-1)^i c_i^{q,r}\mathbf{g}_i(\mathbf{p}, 1) - \sum_{i=0}^{r} c_i^{r,q}\mathbf{g}_i(\mathbf{p}, 0).$$

Suppose that $\mathbf{x}(\mathbf{p}, t)$ is a solution of (7.1.1) and set $\mathbf{g}(\mathbf{p}, w) = \mathbf{x}(\mathbf{p}, t + wh_j)$. Accordingly, for $h_j = t_{j+1} - t_j$ we have

$$\frac{d^{r+q+1}\mathbf{g}}{dw^{r+q+1}}(\mathbf{p}, w) = h_j^{r+q+1}\frac{d^{r+q+1}\mathbf{x}}{dt^{r+q+1}}(\mathbf{p}, t_j + wh_j)$$
$$\mathbf{g}_i(\mathbf{p}, 0) = \frac{1}{i!}\frac{d^i\mathbf{g}}{dw^i}(\mathbf{p}, 0) = h_j^i\frac{1}{i!}\frac{d^i\mathbf{x}}{dt^i}(\mathbf{p}, t_j) = h_j^i\mathbf{f}^{[i]}(\mathbf{x}(\mathbf{p}, t_j), \mathbf{p}, t_j)$$
$$\mathbf{g}_i(\mathbf{p}, 1) = \frac{1}{i!}\frac{d^i\mathbf{g}}{dw^i}(\mathbf{p}, 1) = h_j^i\frac{1}{i!}\frac{d^i\mathbf{x}}{dt^i}(\mathbf{p}, t_j + h_j)$$
$$= h_j^i\mathbf{f}^{[i]}(\mathbf{x}(\mathbf{p}, t_j + h_j), \mathbf{p}, t_j + h_j).$$

Provided that $\bar{\nu} \in [t_{j-1}, t_j]$ is selected such that it satisfies the nonlinear system of equations

$$\begin{aligned}\mathbf{0} = &\sum_{i=0}^{q} (-1)^i c_i^{q,r} h_j^i \mathbf{f}^{[i]}(\mathbf{x}(\mathbf{p}, t_{j+1}), \mathbf{p}, t_{j+1}) \\ &- \sum_{i=0}^{r} c_i^{r,q} h_j^i \mathbf{f}^{[i]}(\mathbf{x}(\mathbf{p}, t_j), \mathbf{p}, t_j) \\ &+ (-1)^q \frac{q!r!}{(r+q)!} h_j^{r+q+1} \frac{d^{r+q+1}\mathbf{x}}{dt^{r+q+1}}(\mathbf{p}, \bar{\nu}),\end{aligned} \tag{7.4.1}$$

then (7.4.1) may be solved to determine $\mathbf{x}(\mathbf{p}, t_{j+1})$ as a single step of a Hermite-Obreschkoff method of order $q - r$ with an approximation of the local error as $O(h_j^{r+q+1})$ [212]. Note that this is a generalization of Taylor series methods, which for $q = 0$, we obtain an explicit Taylor series, and for $r = 0$, an implicit Taylor series is recovered.

7.4.1 Parametric Interval Hermite-Obreschkoff Method

Provided that *a priori* bounds are known and that the regularity conditions are satisfied pertaining to the interval extension of the Jacobian of (7.4.1), state bounds and relaxations can be evaluated using parametric interval contractor methods or via the implicit function theory of Stuber et al. [300] as applied to the nonlinear equation (7.4.1). In either approach, the nonlinear form is iteratively linearized using the mean value theorem, and new relaxations are constructed by relaxing Newton-like iterations. In the following section, an explicit treatment of the Hermite-Obreschkoff form (7.4.1) will be derived. This development parallels that of the relaxation-based method in Sahlodin and Chachuat [255]. This contrasts with the original Lohner's method in that terms originating from an implicit Taylor series expansion are also included in the calculation. Accordingly, this begins by expanding this about the point $(\hat{\mathbf{x}}_j, \hat{\mathbf{x}}_{j+1}, \mathbf{p}) \in X \times X \times P$ using the mean value theorem. For $\bar{\nu} \in [t_{j-1}, t_j]$, let $\mathbf{x}(\mathbf{p}, \bar{\nu})$ be a point such

that

$$\begin{aligned}
&\underbrace{\sum_{i=0}^{q}(-1)^i c_i^{q,r} h_j^i \mathbf{f}^{[i]}(\hat{\mathbf{x}}_{j+1}, \hat{\mathbf{p}})}_{\mathbf{v}_{j+1}} + \underbrace{\left(\sum_{i=0}^{q}(-1)^i c_i^{q,r} h_j^i \mathbf{J}_{\mathbf{x}}^{[i]}(\mathbf{x}_{j+1}(\mathbf{p}), \hat{\mathbf{x}}_{j+1}, \hat{\mathbf{p}})\right)}_{\mathbf{J}_{j+1}^{x}(\mathbf{p})}(\mathbf{x}_{j+1}(\mathbf{p}) - \hat{\mathbf{x}}_{j+1}) \\
&\qquad + \underbrace{\left(\sum_{i=0}^{q}(-1)^i c_i^{q,r} h_j^i \mathbf{J}_{p}^{[i]}(\hat{\mathbf{x}}_{j+1}, \mathbf{p}, \hat{\mathbf{p}})\right)}_{\mathbf{J}_{j+1}^{p}(\mathbf{p})}(\mathbf{p} - \hat{\mathbf{p}}) \\
&= \underbrace{\sum_{i=0}^{r} c_i^{r,q} h_j^i \mathbf{f}^{[i]}(\hat{\mathbf{x}}_j, \hat{\mathbf{p}})}_{\mathbf{v}_j} + \underbrace{\left(\sum_{i=0}^{q} c_i^{r,q} h_j^i \mathbf{J}_{x}^{[i]}(\mathbf{x}_j(\mathbf{p}), \hat{\mathbf{x}}_j, \hat{\mathbf{p}})\right)}_{\mathbf{J}_{j}^{x}(\mathbf{p})}(\mathbf{x}_j(\mathbf{p}) - \hat{\mathbf{x}}_j) \\
&\qquad + \underbrace{\left(\sum_{i=0}^{q} c_i^{r,q} h_j^i \mathbf{J}_{p}^{[i]}(\hat{\mathbf{x}}_j, \mathbf{p}, \hat{\mathbf{p}})\right)}_{\mathbf{J}_{j}^{p}(\mathbf{p})}(\mathbf{p} - \hat{\mathbf{p}}) \\
&\qquad + \underbrace{(-1)^q \frac{q!r!}{(r+q)!} h_j^{r+q+1} \frac{d^{r+q+1}\mathbf{x}}{dt^{r+q+1}}(\mathbf{p}, \bar{\nu})}_{\mathbf{R}_{j+1}(\mathbf{p})}, \qquad (7.4.2)
\end{aligned}$$

where $\mathbf{J}_x^{[i]}(w, \hat{w}, y)$ is the Jacobian of $\mathbf{f}^{[\mathbf{i}]}$ with respect to $\mathbf{x}$ with its l-th row evaluated at $w + \theta_{il}(\hat{w} - w)$ for some $\theta_{il} \in [0, 1]$. Furthermore, $\mathbf{J}_p^{[i]}(y, w, \hat{w})$ is the Jacobian of $\mathbf{f}^{[\mathbf{i}]}$ with respect to $\mathbf{p}$ with its lth row evaluated at $w + \eta_{il}(\hat{w} - w)$ for some $\eta_{il} \in [0, 1]$. We now detail the development of a computational scheme that mirrors that of Lohner's method.

For notational simplicity, let $\mathbf{d}_{j+1}(\mathbf{p}) = \mathbf{v}_j - \mathbf{v}_{j+1} + \mathbf{r}_{j+1}(\mathbf{p})$. Furthermore, assume that $\mathbf{Y} \in \mathbb{R}^{n_x \times n_x}$ is a preconditioning matrix. A particularly common choice for this preconditioning matrix is $\mathrm{m}(J_{j+1}^{x}(X_{j+1}, P))$, the element-wise midpoint of the interval extension of the Jacobian. Once again, we assume that the parallelepiped enclosure of the state variable is available, that is,

there exists $\boldsymbol{\delta}_j(\mathbf{p}) \in \boldsymbol{\Delta}_j(\mathbf{p})$ such that $\mathbf{x}_j(\mathbf{p}) - \hat{\mathbf{x}}_j = \mathbf{A}_j\boldsymbol{\delta}_j(\mathbf{p})$ which is substituted into (7.4.2) to yield

$$\begin{aligned}\mathbf{J}^x_{j+1}(\mathbf{p})(\mathbf{x}_{j+1}(\mathbf{p}) - \hat{\mathbf{x}}_{j+1}) =& \mathbf{d}_{j+1}(\mathbf{p}) + \mathbf{J}^x_j(\mathbf{p})\mathbf{A}_j\boldsymbol{\delta}_j(\mathbf{p}) \\ &+ (\mathbf{J}^p_j(\mathbf{p}) - \mathbf{J}^p_{j+1}(\mathbf{p}))(\mathbf{p} - \hat{\mathbf{p}}).\end{aligned}$$

Next, this equation is multiplied by the preconditioner and rearranged to yield an expression for $\mathbf{x}_{j+1}(\mathbf{p})$ of the following form:

$$\begin{aligned}\mathbf{x}_{j+1}(\mathbf{p}) =& \hat{\mathbf{x}}_{j+1} + \mathbf{Y}^{-1}\mathbf{d}_{j+1}(\mathbf{p}) + (\mathbf{Y}^{-1}(\mathbf{J}^x_j\mathbf{A}_j))\boldsymbol{\delta}_j(\mathbf{p}) \\ &+ (\mathbf{Y}^{-1}(\mathbf{J}^p_j - \mathbf{J}^p_{j+1}))(\mathbf{p} - \hat{\mathbf{p}}) - (\mathbf{Y}^{-1}(\mathbf{J}^x_{j+1} - \mathbf{Y}))(\mathbf{x}_{j+1}(\mathbf{p}) - \hat{\mathbf{x}}_{j+1}).\end{aligned}$$

The parallelepiped enclosure is then updated by computing $\mathbf{A}_{j+1}$ as the $\mathbf{Q}$ matrix from QR-factorization of $\mathrm{m}(B_{j+1}(P))$, where

$$B_{j+1}(P) = \mathrm{m}(J^x_{j+1}(X_{j+1}, P))^{-1}(J^x_j(X_j, P)\mathbf{A}_j),$$

and $\Delta_{j+1}(P)$ is then updated via the identity $\Delta_{j+1}(P) = \mathbf{A}^{-1}_{j+1}(X_{j+1} - \hat{\mathbf{x}}_{j+1})$ after taking an interval extension. This yields the expression:

$$\begin{aligned}\Delta_{j+1}(P) =& (\mathbf{A}^{-1}_{j+1}B_{j+1}(P))\Delta_j(P) + (\mathbf{A}^{-1}_{j+1}C_{j+1}(P))(X^0_{j+1} - \hat{\mathbf{x}}^0_{j+1}) \\ &+ (\mathbf{A}^{-1}_{j+1}(J^p_j(P) - J^p_{j+1}(P))(P - \hat{\mathbf{p}}) + (\mathbf{A}^{-1}_{j+1}\mathbf{Y}^{-1})D_{j+1}(P).\end{aligned}$$

Making use of interval bounds $P \in \mathbb{IR}^{n_p}$ and $X^0_{j+1} \in \mathbb{IR}^{nx}$ such that $\mathbf{p} \in P$ and $\mathbf{x}_{j+1} \in X^0_{j+1}$, we

obtain the parametric interval version of the Hermite-Obreschkoff method:

$$D_{j+1}(P) = \mathbf{v}_j - \mathbf{v}_{j+1} + R_{j+1}(P) \tag{7.4.3}$$

$$B_{j+1}(P) = \mathbf{Y}^{-1}(J_j^x(P)\mathbf{A}_j) \tag{7.4.4}$$

$$C_{j+1}(P) = \mathbf{Y}^{-1}J_{j+1}^x(P) \tag{7.4.5}$$

$$\begin{aligned} X_{j+1} = &(\hat{\mathbf{x}}_{j+1}^0 + B_{j+1}(P)\Delta_j(P) + C_{j+1}(P)(X_{j+1}^0 - \hat{\mathbf{x}}_{j+1}^0) \\ &+ \mathbf{Y}^{-1}D_{j+1}(P)\mathbf{Y}^{-1}(J_j^p(P) - J_{j+1}^p(P))(P - \hat{\mathbf{p}}) \cap X_{j+1}^0 \end{aligned} \tag{7.4.6}$$

$$\mathbf{x}_{j+1} = \mathrm{m}(X_{j+1}) \tag{7.4.7}$$

$$\begin{aligned} \Delta_{j+1}(P) = &(\mathbf{A}_{j+1}^{-1}B_{j+1}(P))\Delta_j(P) + (\mathbf{A}_{j+1}^{-1}C_{j+1}(P))(X_{j+1}^0 - \hat{\mathbf{x}}_{j+1}^0) \\ &+ (\mathbf{A}_{j+1}^{-1}(J_j^p(P) - J_{j+1}^p(P)))(P - \hat{\mathbf{p}}) + (\mathbf{A}_{j+1}^{-1}\mathbf{Y}^{-1})D_{j+1}(P), \end{aligned} \tag{7.4.8}$$

where $\mathbf{A}_{j+1}$ is again the orthogonal matrix $\mathbf{Q}$ from the QR-factorization of $\mathrm{m}(B_{j+1}(P))$. An interval bound X_{j+1}^0 can be obtained via the a priori enclosure obtained using the higher-order enclosure tests [215] or, alternatively, the parametric interval version of the Lohner's method may be applied to first refine these bounds, as discussed by Nedialkov and Jackson [212]. We make use of the latter approach in this chapter, where the order of the Taylor series used in Lohner's method is fixed to r as suggested by Sahlodin and Chachuat [256] to minimize additional computational time. This may serve to minimize redundant calculations, since when a Lohner's method of order r is used, the Jacobian of each Taylor coefficient computed may be stored and reused in the corrector step outlined by (7.4.3)-(7.4.8).

7.4.2 Convex/Concave State Relaxations via Parametric Hermite-Obreschkoff method

We now follow a similar line of development to derive a method to generate convex/concave relaxations of (7.1.1) via a parametric Hermite-Obreschkoff method. We follow the convention introduced in Section 7.3.3, in which $f : P \to Z$, we denote the convex and concave relaxations of $\mathbf{f}$ at a $\mathbf{p} \in P$ as $\{\mathbf{f}^{cv}, \mathbf{f}^{cc}\}(\mathbf{p})$. Accordingly, convex/concave relaxations computed using the parametric Hermite-Obreschkoff method are given by the following:

$$\{\mathbf{d}_{j+1}^{cv}, \mathbf{d}_{j+1}^{cc}\}(\mathbf{p}) = \mathbf{v}_j - \mathbf{v}_{j+1} + \{\mathbf{r}_j^{cv}, \mathbf{r}_j^{cc}\}(\mathbf{p}) \tag{7.4.9}$$

$$\{\mathbf{b}_{j+1}^{cv}, \mathbf{b}_{j+1}^{cc}\}(\mathbf{p}) = \mathbf{Y}^{-1}(\{\mathbf{J}_j^{x,cv}, \mathbf{J}_j^{x,cc}\}(\mathbf{p})\mathbf{A}_j) \tag{7.4.10}$$

$$\{\mathbf{c}_{j+1}^{cv}, \mathbf{c}_{j+1}^{cc}\}(\mathbf{p}) = \mathbf{Y}^{-1}\{\mathbf{J}_{j+1}^{x,cv}, \mathbf{J}_{j+1}^{x,cc}\}(\mathbf{p}) \tag{7.4.11}$$

$$\begin{aligned}\{\mathbf{x}_{j+1}^{cv}, \mathbf{x}_{j+1}^{cc}\}(\mathbf{p}) = (\hat{\mathbf{x}}_{j+1}^0 &+ \{\mathbf{b}_{j+1}^{cv}, \mathbf{b}_{j+1}^{cc}\}(\mathbf{p})\{\boldsymbol{\Delta}_j^{\boldsymbol{cv}}, \boldsymbol{\Delta}_j^{\boldsymbol{cc}}\}(\mathbf{p}) \\ &+ \{\mathbf{c}_{j+1}^{cv}, \mathbf{c}_{j+1}^{cc}\}(\mathbf{p})(\{\mathbf{x}_{j+1}^{0,cv}, \mathbf{x}_{j+1}^{0,cc}\} - \hat{\mathbf{x}}_{j+1}^0) \\ &+ \{\mathbf{d}_{j+1}^{cv}, \mathbf{d}_{j+1}^{cc}\}(\mathbf{p}) \\ &+ \mathbf{Y}^{-1}(\{\mathbf{J}_j^{p,cv}, \mathbf{J}_j^{p,cc}\}(\mathbf{p}) \\ &- \{\mathbf{J}_{j+1}^{p,cv}, \mathbf{J}_{j+1}^{p,cc}\}(\mathbf{p}))(\mathbf{p} - \hat{\mathbf{p}})) \cap \mathbf{x}_{j+1}^0(\mathbf{p})\end{aligned} \tag{7.4.12}$$

$$\mathbf{x}_{j+1} = \mathrm{m}(X_{j+1}) \tag{7.4.13}$$

$$\begin{aligned}\{\boldsymbol{\Delta}_{j+1}^{cv}, \boldsymbol{\Delta}_{j+1}^{cc}\}(\mathbf{p}) = (\mathbf{A}_{j+1}^{-1}&\{\mathbf{b}_{j+1}^{cv}, \mathbf{b}_{j+1}^{cc}\}(\mathbf{p}))\{\boldsymbol{\Delta}_j^{cv}, \boldsymbol{\Delta}_j^{cc}\}(\mathbf{p}) \\ &+ (\mathbf{A}_{j+1}^{-1}\{\mathbf{c}_{j+1}^{cv}, \mathbf{c}_{j+1}^{cc}\}(\mathbf{p}))(\{\mathbf{x}_{j+1}^{cv}, \mathbf{x}_{j+1}^{cc}\}(\mathbf{p}) - \hat{\mathbf{x}}_{j+1}^0) \\ &+ (\mathbf{A}_{j+1}^{-1}(\{\mathbf{J}_j^{p,cv}, \mathbf{J}_j^{p,cc}\}(\mathbf{p}) - \{\mathbf{J}_{j+1}^{p,cv}, \mathbf{J}_{j+1}^{p,cc}\}(\mathbf{p})))(\mathbf{p} - \hat{\mathbf{p}}) \\ &+ (\mathbf{A}_{j+1}^{-1}\mathbf{Y}^{-1})\{\mathbf{d}_{j+1}^{cv}, \mathbf{d}_{j+1}^{cc}\}(\mathbf{p}).\end{aligned} \tag{7.4.14}$$

By construction, relaxations calculated from (7.4.3)-(7.4.8) are necessarily tighter than interval bounds computed according to (7.4.9)-(7.4.14). Moreover, relaxations $\{\mathbf{\Delta}^{cv}_{j+1}, \mathbf{\Delta}^{cc}_{j+1}\}(\mathbf{p})$ and $\{\mathbf{x}^{cv}_{j+1}, \mathbf{x}^{cc}_{j+1}\}(\mathbf{p})$ and their associated subgradients may be used to refine natural interval bounds that are used in the standard McCormick arithmetic according to Proposition (7.2.5). This may serve to further improve relaxations of these values computed at the subsequent time steps $j+2, j+3, \ldots$, and so on.

We now revisit Example 7.3.4 using the Hermite-Obreschkoff parametric interval method described above. A detailed analysis of the relaxation approach will be reserved for Section 7.5. All Hermite-Obreschkoff parametric interval methods yield less conservative relaxations than the previously considered methods. As illustrated in the right panel of Figure 7.4.1, the order 1-1, 2-2, and 3-3 methods all yield similar bounds when a stepsize of $h_j = 0.01$ is specified. The distinction between these methods becomes more evident when a stepsize of $h_j = 0.025$ is specified, and a monotonic trend in interval widths at time $t = 1$ is readily apparent from the left panel of Figure 7.4.1. This results in a higher-order method that effectively reduces the number of necessary steps. It is also worth noting that when 40 steps ($h_j = 0.025$) are taken instead of 100 steps ($h_j = 0.01$), the Hermite-Obreschkoff methods result in less overestimation of the final interval width when compared to the previously considered methods.

7.5 Case Studies

The utility of the two novel approaches introduced herein is assessed on two case studies. All numerical experiments in this work were run on a single thread of an Intel Xeon E3-1270 v5 3.60/4.00GHz (base/turbo) processor with 16GB ECC RAM allocated to an Ubuntu 18.04LTS operating system virtual machine and Julia v1.7.1 [36]. The Intel MKL 2022.1 [96] was used to

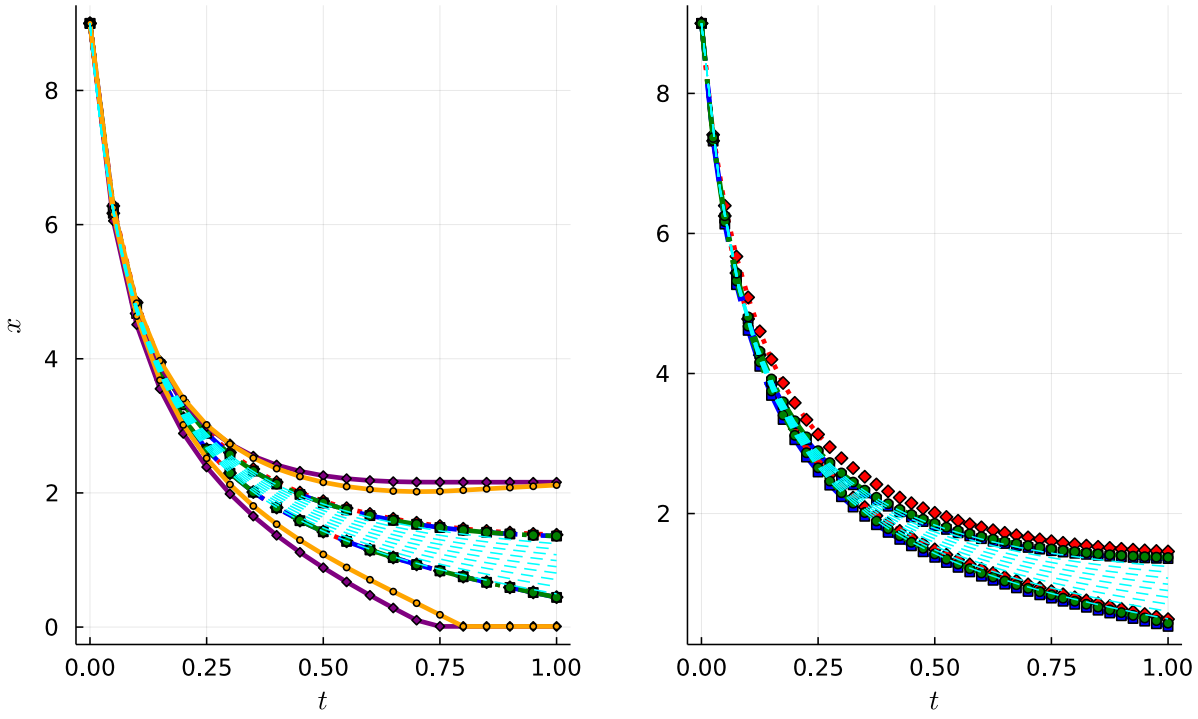

FIGURE 7.4.1: The local solution trajectories of (7.3.12) of Ex. 7.3.4 for several $p \in P$ are depicted in both plots for reference (dashed-cyan). **Left:** An illustration of state bounds for interval Hermite-Obreschkoff methods of $r - q$ order: 1-1 (red-diamond), 2-2 (blue-rectangle), and 3-3 (green-circle). The bounds are compared to those obtained using a 2-step PILMS method (purple-diamond) and the previous non-validated solution bounding method presented [328] (orange-circle). Each method is applied to Example 7.3.4 assuming a stepsize of $h_j = 0.01$. **Right:** The state bounds associated with interval Hermite-Obreschkoff methods of $r - q$ order: obtained when the step-size of $h_j = 0.025$: 1-1 (red-diamond), 2-2 (blue-rectangle), and 3-3 (green-circle).

	$r-q$ Order		
	1-1	2-2	3-3
Interval enclosure width, 100 steps	0.9346	0.9146	0.9138
Interval enclosure width, 25 steps	1.2079	1.0863	0.9600
CPU time (s) interval bounds, 100 steps	0.0049	0.0060	0.0079
CPU time (s) interval bounds, 25 steps	0.0010	0.0015	0.0020

TABLE 7.4.1: Comparison of the width of the interval at final time to the the CPU run time (s) per step required to calculate the state bounds and state relaxations for Example 7.3.4 using a Hermite-Obreschkoff method of order $r-q$.

perform all LAPACK [11, 320] and BLAS [40] routines. All $\partial\mathbf{f}/\partial\mathbf{x}$ and $\partial\mathbf{f}/\partial\mathbf{p}$ functions were symbolically derived, manually simplified, and provided to the integrator as user-defined functions. In principle, an automatic differentiation tool such as ForwardDiff.jl [245] could be used for this step. However, the use of overloading automatic differentiation tools can lead to significant expansiveness in the enclosures due to dependency effects. As such, user-defined functions are often preferred, as these implementation-dependent effects may be avoided. Interval arithmetic was performed using the IntervalArithmetic.jl library [260] and McCormick relaxations were computed using the library present in McCormick.jl [329]. Each example presented in this chapter is openly available on a Github repository, `https://github.com/PSORLab/RSImplicitIntegrators`. The DifferentialEquations.jl package was used to integrate ODEs [239].

Two case studies are considered that are common to the dynamical systems literature: the Van der Pol Oscillator and the Lotka-Volterra Predator-Prey Model. Each of the methods discussed in this work was applied to each case study and analyzed.

7.5.1 Van der Pol Oscillator

Consider the pODE system that represents the Van der Pol Oscillator:

$$\begin{aligned} \dot{x}_1(p,t) &= x_2, \\ \dot{x}_2(p,t) &= p(1-x_1^2)x_2 - x_1, \end{aligned} \tag{7.5.1}$$

over the time domain $I = [0, 3]$, with the initial condition $\mathbf{x}_0(\mathbf{p}) = (1.4, 2.3)$, and parameter $p \in [1.3, 1.4]$. The initial state bounds of $X = [-3, 3] \times [-6, 4]$ are set based on a preliminary exploratory analysis, as these are required by the non-validated 2-step Adams-Moulton method [328]. The following methods for computing state bounds are compared: (i) a fixed stepsize non-validated 2-step Adams-Moulton method [327], (ii) a fixed stepsize validated 2-step Adams-Moulton method presented in Section 7.3.3, and (iii) a Hermite-Obreschkoff method of order 3-3 presented in Section 7.4.2. In each case, 200 steps are used, (i.e., $h = 200/3$). The resulting trajectories are shown in Figure 7.5.1. The validated 2-step Adam-Moulton method detailed in this chapter results in tighter state bounds than the previous non-validated 2-step Adam-Moulton method. Method (iii) results in equivalent state bounds to Method (ii) for x_1 at early times, but begins to diverge at later times, while for x_2 the state bounds computed by Method (ii) are initially weaker until the bounds computed using Method (iii) begin to rapidly diverge.

Methods (i), (ii), and (iii) ran in 0.0032, 0.0065, and 0.436 CPU seconds, respectively. Clearly, a significant reduction in run time is achieved by using the lower-order Method (ii) as opposed to Method (iii). The greater CPU run time of Method (ii) relative to Method (i) is expected due to the inclusion of existence and uniqueness tests in Stage 1 as well as the underlying mean-value expansion. As such, the PILMs method developed herein may present an attractive alternative to existing interval Hermite-Obreschkoff and the non-validated implicit

PILMs method. We now proceed to examine a second example and the potential of adaptive stepsize and relaxation-based approaches in conjunction with these methods.

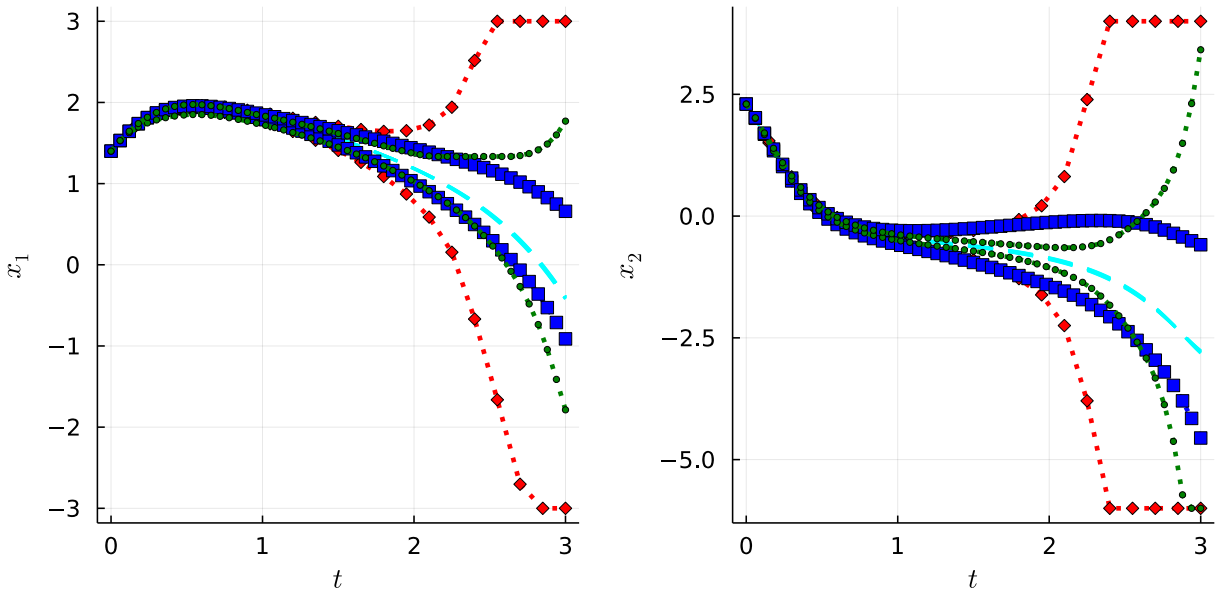

FIGURE 7.5.1: Plots of the state bounds for $x_1(p)$ and $x_2(p)$ of the Van der Pol oscillator example (Sec. 7.5.1) computed using Method (i) a fixed stepsize non-validated 2-step Adams-Moulton method [327] (red-diamond), Method (ii) a fixed stepsize validated 2-step Adams-Moulton method presented in Section 7.3.3 (blue-rectangle), and Method (iii) a Hermite-Obreschkoff method of order 3-3 presented in Section 7.4.2 (green-circle), along with some solution trajectories of (7.5.1) (teal-dash).

7.5.2 Lotka-Volterra System

A Lotka-Volterra system consists of a two-dimensional ODE system used to model predator-prey interactions. A common variation of the Lotka-Volterra system encountered in reachability analysis and state relaxation research [122, 256, 276], consists of the differential equations

$$\begin{aligned} \dot{x}_1(\mathbf{p}, t) &= p_1 x_1(1 - x_2), \\ \dot{x}_2(\mathbf{p}, t) &= p_2 x_2(x_1 - 1). \end{aligned} \tag{7.5.2}$$

We consider the time domain $I = [0, 4]$, which was chosen to help visualize differences among the methods considered. The initial condition is taken to be the constant values $\mathbf{x}_0(\mathbf{p}) = (1.2, 1.1)$, with the parameters $\mathbf{p} \in P = [2.99, 3.01] \times [0.99, 1.01]$. Since the non-validated 2-step Adams-Moulton method [328] requires an initial state bounds be provided and after initial exploratory analysis we assume constant state bounds of $X = [0.1, 2]^2$. However, we note that there are several methods to furnish these state bounds for general systems, including classical parametric interval methods [154, 218].

Five distinct state relaxation methods are compared, allowing for contrasting analyses of both fixed stepsize and adaptive stepsize methods along with both of the implicit methods presented herein. The methods are: (i) a fixed stepsize non-validated 2-step Adams-Moulton method with $h = 0.04$ [327], (ii) a fixed stepsize validated 2-step Adams-Moulton method with $h = 0.04$ presented in Section 7.3.3, (iii) a Hermite-Obreschkoff method of order 3-3 with $h = 0.04$ presented in Section 7.4.2, (iv) the adaptive version of the validated 2-step Adams-Moulton method presented in Section 7.3.3, and (v) the adaptive version of the Hermite-Obreschkoff method of order 3-3 presented in Section 7.4.2. Plots of the state bounds calculated by applying the relaxation methods with subgradient-based interval refinement at the reference point $\mathbf{p} = (3.0, 1.0)$ are shown in Figure 7.5.2. An existence-and-uniqueness test of order 3 was used for Methods (ii) and (iv), while a test of order 7 was used for Methods (iii) and (v). The absolute and relative tolerances for LEPUS control of the adaptive methods were set to 10^{-5} based on the local truncation error. We note that this LEPUS control scheme cannot ensure error below tolerance values due to inherent overestimation, also associated with parametric uncertainty.

We observe that each of the validated methods (ii) - (v), which rigorously account for truncation error and do not require prior knowledge of existing state bounds, provide tighter state

Method	i	ii	iii	iv	v
Steps	100	100	100	56	45
CPU time (ms)	1.5	4.3	33.1	4.4	15.1
CPU time per step (ms)	0.015	0.043	0.331	0.079	0.336

TABLE 7.5.1: A comparison of the CPU run time (ms) and steps taken for each method used for the Lotka-Volterra example presented in Section 7.5.2. The methods compared are as follows: (i) a fixed stepsize non-validated 2-step Adams-Moulton method with $h = 0.04$ [327], (ii) a fixed stepsize validated 2-step Adams-Moulton method with $h = 0.04$ presented in Section 7.3.3, (iii) a Hermite-Obreschkoff method of order 3-3 with $h = 0.04$ presented in Section 7.4.2, (iv) the adaptive stepsize validated 2-step Adams-Moulton method presented in Section 7.3.3, and (v) the adaptive stepsize Hermite-Obreschkoff method of order 3-3 presented in Section 7.4.2.

bounds than Method (i). The adaptive methods (iv) and (v) use fewer time steps than the fixed stepsize approaches. However, additional calculations are required for the LEPUS control approach. As a consequence, the CPU run time used by each algorithm for the adaptive approach may not improve for algorithms which construct second-stage bounds quickly and efficiently. We observe this when comparing methods (ii) and (iv). Although the adaptive method takes 56 steps to furnish wider bounds than the 100-step method, the overall time per step taken increases from 0.043 ms to 0.079 ms fully offsetting any potential improvement to overall CPU time. For the more computationally expensive methods (iii) and (v), the CPU time spent per step was relatively unchanged from 0.331 to 0.336 and as a result the adaptive method resulted in a reduction of overall CPU time from 33.1 ms to 15.1 ms.

7.6 Conclusion

In this work, we introduced two novel implicit discretize-then-relax methods for calculating convex/concave relaxations of the solutions of pODEs. The first method was developed by analyzing mean-value forms of interval PILMs methods and subsequently extending these methods to enable the computation of associated relaxations. A variable stepsize form of the PILMs methods was described that integrates into the LEPUS framework, allowing the number of time steps evaluated to be dictated by user-specified tolerances. A second method by which state relaxations are computed was developed based on an adaptation of the interval Hermite-Obreschkoff approach.

We also addressed two case studies and contrasted our newly developed approaches with the previous fixed stepsize PILMs approach [328]. In each case, the PILMs approaches introduced herein provided tighter bounds and relaxations in slightly more computational time when used with the same number of steps as the fixed stepsize approach while bounding the true solution of the pODEs rather than a numerical approximation thereof. Moreover, when an adaptive stepsize approach was enabled, the novel method provided weaker bounds and relaxations in comparable computational time, illustrating the inherent performance of the simpler method. Additionally, we illustrated that the Hermite-Obreschkoff methods for computing relaxations did not necessarily improve the overestimation of the bounds and relaxations for either fixed or adaptive stepsize routines relative to the PILMs method despite significantly longer run times.

One avenue of future research that merits study lies in the use of our novel methods in conjunction with a discrete-time differential inequality method [336]. The current state-of-the-art method makes use of an explicit Euler discretization; as such, our methods may be integrated as an additional corrector step resulting in further refinement of state bounds. This should reduce overly-conservative state bounds and state relaxation calculations allowing for larger step sizes.

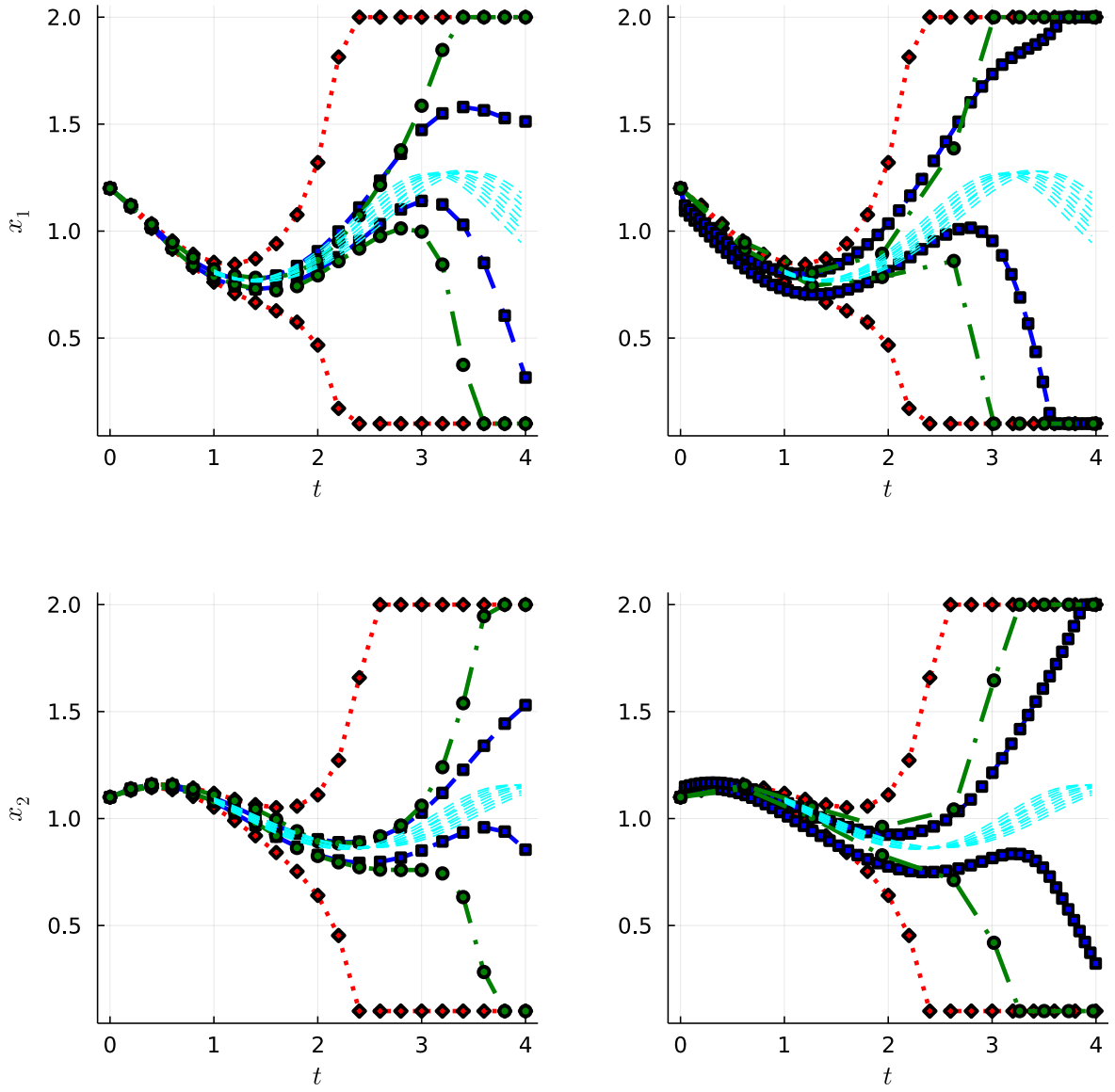

FIGURE 7.5.2: State bounds for the Lotka-Volterra system example (Sec. 7.5.2) are determined using a subgradient refinement procedure with Method (i) a fixed stepsize non-validated 2-step Adams-Moulton method with $h = 0.04$ [327], (red-diamond), Method (ii) a fixed stepsize validated 2-step Adams-Moulton method with $h = 0.04$ presented in Section 7.3.3 (blue-rectangle), and Method (iii) a Hermite-Obreschkoff method of order 3-3 with $h = 0.04$ presented in Section 7.4.2 (green-circle) in the left panels (**left**) and are compared to Method (i) (red-diamond), Method (iv) the adaptive version of the validated 2-step Adams-Moulton method presented in Section 7.3.3 (blue-rectangle), and Method (v) the adaptive version of the Hermite-Obreschkoff method of order 3-3 presented in Section 7.4.2 (green-circle) (**right**) for state variables $x_1(p)$ (**top**) and $x_2(p)$ (**bottom**). Local trajectories for $\mathbf{p} \in P$ are given for reference (teal-dash).

Chapter 8

Robust Optimization of Surrogate Models with Validity Constraints

In Stuber et al. [299], the worst-case design of a subsea oil production facility (illustrated in Figure 8.1.1) was formulated as an operation under uncertainty feasibility problem and solved using several novel methodologies. Namely, the problem was reformulated as an SIP with implicit functions embedded. The subsea separator model uses transcendental functions with definitions on narrow domains that result in numerical difficulties when simulating and optimizing the system. For the purposes of this chapter, the interest is not in the application itself, but in the model as representative of a broader class of industrially-relevant examples plagued by numerical simulation and convergence issues caused by domain violations. Within this context, it is of interest to explore how hybrid modeling approaches might be used to improve the robustness of an first-principles model and solvers (i.e., improve the reliable convergence to accurate solutions).

Domain violations are ubiquitous across process systems engineering applications and pose major challenges to researchers and practitioners of simulation and optimization [17, 102, 321]. Within the broader context of numerical simulation, domain violations are encountered when a

solver attempts to evaluate an expression at a point outside of its defined domain (e.g., divide by zero or square-root a negative number). Hybrid models may pose additional challenges as they may also suffer from violations of their domains of validity. That is, a solver may attempt to evaluate a DDM at a point outside of the domain of inputs for which the DDM is considered to be "valid" (i.e., accurately represents reality). When considering the optimization of hybrid models, domain violations may be frequently encountered when such domains are not explicitly known and accounted for with appropriate constraints.

In Stuber et al. [299], a method of forward-backward interval constraint propagation on the DAG [262, 296], interval contractor methods [219], a novel convex/concave relaxation algorithm [300], and a novel algorithm for solving SIPs [298] were all necessary to solve this problem. While these methods adequately address the problem in question, the broad and robust applicability of this approach to more generalized SIPs is wanting. We should note, however, that this approach reduces the problem in question from a generalized semi-infinite program to that of a standard SIP. This, combined with a desire to generalize the prior results to allow for the incorporation of more complex physical phenomena, further motivates our interest in this example.

8.1 Hybrid Model Formulation

In this study, the focus is on a modification of the gas-liquid/liquid-liquid separation train problem presented in Case 3 of Stuber et al. [299]. To model the performance of gas separation in each separator, simple exponential decay models based on mean gas bubble sizes were assumed [299]. The relationship between inlet and outlet gas quantities may be expected to change in meaningful ways when a population-based model of bubble sizes is incorporated along with information

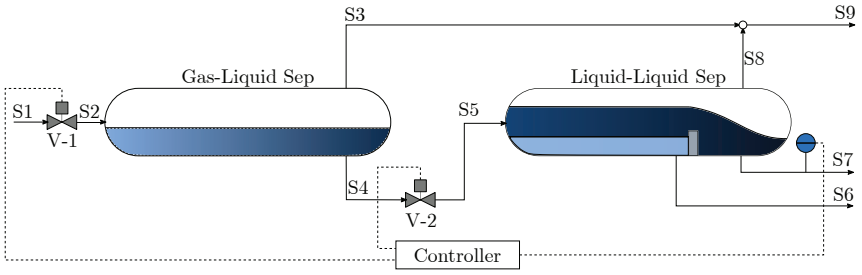

FIGURE 8.1.1: The process flow diagram for the subsea separator (adapted from Stuber et al. [299]), is presented in this figure. This system is considered in the subsea separator case study for the use of hybrid models to overcome numerical domain violation issues. A mixture of gas, oil, and water is fed to the system in S1. Gas is separated from the oil-water mixture in the gas-liquid separator and oil is separated from water in the liquid-liquid separator.

about the equipment's geometry. Moreover, for bubbly mixtures, overflow can occur in volumes less than those considered by solely taking into account liquid levels, provided a large gas concentration is present in the inlet. In practice, this type of problem is typically characterized using a mixture of computational fluid dynamics software and empirical investigation.

We propose simplifying the published model by using an ANN surrogate model to relate the input variables to the gas-liquid separator and the control variable for the second valve (V-2) to the system outputs. This serves to eliminate the domain violation issue inherent in the model, as the activation functions considered lack domain restrictions, and allow the system-level model to be readily generalized to incorporate information from computational experiments generated by CFD models, or elsewhere. The inputs, outputs, and expected ranges of each variable in each ANN are summarized in Table 8.1.1. As the development of CFD models is often time consuming, equipment specific, and not the central focus of this work, we will forgo this and instead illustrate how this approach works at the system level. We use the prior mass balances and process specifications for the gas-liquid separator (GLS) and the liquid-liquid separator (LLS). The governing equations for the first valve (V-1), and the gas mixer will be left unaltered. The

equations governing V-1 lead to the following simplified relationships:

$$\begin{aligned} \xi_{W,1} &= 1 - \xi_{G,1} - \xi_{O,1}, \\ SG_{mix}^{-1} &= \frac{\xi_{G,1}}{SG_G} + \frac{\xi_{W,1}}{SG_W} + \frac{\xi_{O,1}}{SG_O}, \\ \dot{m}_2 &= u_1 C_{v1} \sqrt{SG_{mix}^{-1}(P_{well} - P_{GLS}) + \epsilon_d} \\ \boldsymbol{\xi}_2 &= \boldsymbol{\xi}_1. \end{aligned} \tag{8.1.1}$$

These equations specify that the mass fractions in the input stream ($\xi_{W,1}$, $\xi_{G,1}$, $\xi_{O,1}$) sum to one, provide a formula relating specific gravity of the mixture SG_{mix} to the specific gravity of individual components (SG_G, SG_W, SG_O), and relate the mass flow rate through the valve $\dot{m}_2$ to valve position (u_1), valve coefficient (C_{v1}) and a specified pressure difference ($P_{well} - P_{GLS}$) between the GLS and the wellhead. A small number $\epsilon_d = 10^{-6}$ is added to the argument of the $\sqrt{\cdot}$ function to avoid the introduction of numerically ill-posed gradients that present computational issues for local NLP subproblems encountered during global optimization.

Simple algebraic substitutions of the equations governing V-2 and the LLS behavior lead to the following algebraic expression:

$$\xi_{G,7} = \xi_{G,4} \exp\left(-\dot{m}_4 k_{LLS} \frac{V_{LLS}}{\rho_4 + \epsilon_d}\right). \tag{8.1.2}$$

While additional expressions are required to fully determine all stream characteristics in the flowsheet, the LLS performance specification (8.1.2) is sufficient to construct the SIP constraint. This specification relates the inlet gas mass fraction $\xi_{G,4}$, density ρ_4, and mass flow rate $\dot{m}_4$ to the oil product stream gas mass fraction $\xi_{G,7}$ by means of a performance constant k_{LLS}. Due to downstream equipment specifications, the oil product stream gas mass fraction may not exceed the value $G_{max} = 0.05$. The full model can be found in Stuber et al. [299] with the analysis of the

Variable	Lower	Upper	Unit	Layer
$\dot{m}_2$	8.228	19.517	kg/s	Input
u_2	0.35	0.8	-	Input
$\xi_{G,2}$	0.35	0.5	-	Input
$\xi_{W,2}$	0.1	0.25	-	Input
$\dot{m}_4$	541.364	845.881	kg/s	Output
H_{GLS}	0.462165	0.7992	m	Output
$\xi_{G,4}$	9.463053×10^{-3}	0.36	-	Output
P_4	4.00264×10^6	4.01079×10^6	Pa	Output
ρ_4	584.6	1376.6	kg/m^3	Output

TABLE 8.1.1: The state variables for the subsea separator case study are listed in this table along with their corresponding bounds, units, and identification of whether they are classified as inputs or outputs for the hybrid model. Bounds directly specified by Stuber et al. [299] were used if available. Otherwise, natural interval extensions of known quantities were used to compute necessary values. The parameters C_{v1}, SG_G, SG_W, SG_O, g_a, P_{well} P_{LLS}, P_{GLS}, k_{GLS}, L_{GLS}, and R_{GLS} take the values specified in Stuber et al. [299].

DAG in Stuber [296, Sec. 8.1].

8.2 Data-Driven Model Construction

Training data was generated by repeatedly solving a feasibility problem equivalent to the nonlinear system:

$$
\begin{aligned}
(\xi_{G,2}-1)\dot{m}_2-(\xi_{G,4}-1)\dot{m}_4 &= 0 \qquad (8.2.1)\\
u_2^2 C_{v2}^2 \rho_W^o (P_4-P_{LLS})-\rho_4\dot{m}_4^2 &= 0\\
(P_4-P_{GLS})-\rho_4 g_a H_{GLS} &= 0\\
\xi_{G,2}\exp\left(-k_{GLS}\rho_4\frac{(\xi_{G,4}-1)V_{GLS}(H_{GLS})}{(\xi_{G,2}-1)\dot{m}_2}\right)-\xi_{G,4} &= 0\\
\rho_4\frac{\xi_{G,4}}{SG_G}+\rho_4\frac{\xi_{G,2}(\xi_{G,4}-1)}{SG_W(\xi_{G,2}-1)}+\rho_4\frac{\xi_{G,2}(\xi_{G,4}-1)(1+\xi_{W,2}-\xi_{G,2})}{SG_O(\xi_{G,2}-1)}-\rho_W^o &= 0\\
V_{GLS}-L_{GLS}\left((H_{GLS}-R_{GLS})\sqrt{(2R_{GLS}H_{GLS}-H_{GLS}^2)}+R_{GLS}^2\cos^{-1}\left[1-\frac{H_{GLS}}{R_{GLS}}\right]\right) &= 0
\end{aligned}
$$

that is parameterized by $\mathbf{w}=(\dot{m}_2,u_2,\xi_{G,2},\xi_{W,2})\in W$. Ipopt [317] was used to solve (8.2.1) with a multistart approach using 16 initial guesses chosen via an LHC sampling procedure for each set of parameters considered. An LHC sampling procedure was then performed over a range of valid values given in Table 8.1.1 to generate 10^5 data points used to train the DDM. As noted in Stuber et al. [299], the implicit function characterized by (8.2.1) may not exist for some realization of uncertainty and control variables. Values that yielded a locally-infeasible result were labelled accordingly, while the solutions of the feasible problems were saved. Of the 10^5 points generated, 6,742 infeasible points were evaluated.

The approach to training the ANNs for this problem, parallel the previous examples. The data set was scaled using a min-max normalization and divided randomly into training (70%), validation (15%), and test (15%) sets. Training was performed using the Keras [61] module in the nightly version of Tensorflow [1] with the Adam optimizer. The surrogate ANN consisted of four

inputs, two dense layers, twelve neurons per layer, and utilized the SiLU activation function. A sigmoid output layer was used to ensure that the output results remained within the range of the training data. The surrogate model had min-max-scaled mean-squared-error (MSE) values of 7.74×10^{-5} and 2.2506×10^{-4} on the training and test sets, respectively. The validity constraint consists of an ANN with four inputs, two hidden layers, two neurons per layer, and utilizes the SiLU activation functions with a single hyperbolic tangent output layer that is trained using a binary cross-entropy loss function. This achieved a binary accuracy greater than 99.0% on both the test and training sets. Weights and offsets for both the surrogate model and the validity constraint can be found in the Git repository. Both the surrogate and classifier ANNs used a learning rate schedule that began with a value of 0.1 and was decreased by a factor of 0.5 every 100 epochs. We note here that, due to the nature of the application, no classifier can be expected to be exactly accurate as the valid and invalid regions adjoin one another.

8.3 SIP Formulation and Results

Any ANN can only be expected to provide valid results when interpolating and special consideration must be given to exclude invalid operating parameters. In general, two distinct outcomes must be considered: either a domain violation arises from a purely numerical consideration (e.g., instability) or one that corresponds to a nonphysical operating condition (e.g., negative density). In the former case, the accuracy of the hybrid model should be verified to guarantee the results for the corresponding robust operation problem. In the latter case, restricting the model to a domain of validity is sufficient to ensure a guarantee of robustness.

Ensuring validity regions for surrogate models remains an active area of research within the optimization community. Some approaches include restricting the function evaluations to be

within the convex hull of a finite number of sampled points [18, 145] or categorizing the data using a support vector machine [236, 268]. In either case, this restriction can be framed as a potentially nonconvex constraint $g_c : Z \times \Pi \times U \to \{-1, 1\}$ where $g_c(\hat{\mathbf{z}}, \boldsymbol{\pi}, \mathbf{u}) = -1$ indicates a valid model for $(\hat{\mathbf{z}}, \boldsymbol{\pi}, \mathbf{u}) \in Z \times \Pi \times U$. We note that the forms addressed pertain to standard optimization formulations and the extension of these approaches to multilevel programs has yet to be developed. In keeping with surrogate modeling frameworks adopted in this chapter, we choose to make use of a second ANN, $f_c^{ANN} : Z \times \Pi \times U \to \mathbb{R}$, in addition to the surrogate model, to perform a binary classification task via logistic regression.

The binary classification task is performed as follows. Provided that $f_c^{ANN}(\hat{\mathbf{z}}, \boldsymbol{\pi}, \mathbf{u}) \leq 0$, the input features is classified as $g_c(\hat{\mathbf{z}}, \boldsymbol{\pi}, \mathbf{u}) = -1$ (valid classification). In a corresponding manner, the classification ANN predicts that the the input features will be classified as $g_c(\hat{\mathbf{z}}, \boldsymbol{\pi}, \mathbf{u}) = +1$ (invalid classification) due to a domain violation $f_c^{ANN}(\hat{\mathbf{z}}, \boldsymbol{\pi}, \mathbf{u}) > 0$. With this validity constraint, the robust feasibility constraint takes the logical form:

$$\forall \boldsymbol{\pi} \in \Pi, \exists \mathbf{u} \in U : g(\hat{\mathbf{z}}, \boldsymbol{\pi}, \mathbf{u}) \leq 0 \wedge g_c(\hat{\mathbf{z}}, \boldsymbol{\pi}, \mathbf{u}) \leq 0 \wedge \mathbf{h}(\hat{\mathbf{z}}, \boldsymbol{\pi}, \mathbf{u}) = \mathbf{0}. \tag{8.3.1}$$

For this problem, the state variables $\hat{\mathbf{z}}$ can be calculated as an explicit function $\mathbf{z} : \Pi \times U \to Z$ such that $\mathbf{h}(\mathbf{z}(\boldsymbol{\pi}, \mathbf{u}), \boldsymbol{\pi}, \mathbf{u}) = \mathbf{0}$ for every $(\boldsymbol{\pi}, \mathbf{u}) \in \Pi \times U$. The robust operation problem can then be formulated as an SIP with a nonsmooth semi-infinite constraint:

$$\begin{aligned} \eta^* = \max_{\boldsymbol{\pi} \in \Pi, \eta \in H} \ & \eta \\ \text{s.t.} \quad & \eta \leq \max \{g(\mathbf{z}(\boldsymbol{\pi}, \mathbf{u}), \boldsymbol{\pi}, \mathbf{u}), g_c(\mathbf{z}(\boldsymbol{\pi}, \mathbf{u}), \boldsymbol{\pi}, \mathbf{u})\}, \forall \mathbf{u} \in U. \end{aligned} \tag{8.3.2}$$

Alternatively, (8.3.2) may be reformulated as an SIP with a disjunctive constraint or as a mixed-integer SIP. Note that this form is identical to the structure encountered when relaxing a

generalized semi-infinite program and the reader is directed to Mitsos and Tsoukalas [189] for a discussion of the numerical eccentricities associated with solving that problem class. The robust design problem for the subsea separator may then be formally stated as:

$$\begin{aligned} \eta^* = & \max_{\boldsymbol{\pi}\in\Pi,\eta\in H} \eta \\ & \text{s.t. } \eta \leq \max\left\{\xi_{G,7}(\boldsymbol{\pi},\mathbf{u}) - G^{max}, g_c(\mathbf{z}(\boldsymbol{\pi},\mathbf{u}),\boldsymbol{\pi},\mathbf{u})\right\}, \quad \forall \mathbf{u}\in U \\ & U = [0.35, 0.8]^2 \\ & \Pi = [0.35, 0.5]. \end{aligned} \tag{8.3.3}$$

We note that the valid region of the developed binary classifier is bounded by a 0-sublevel set, which is potentially a disconnected and nonconvex set, and therefore the following equivalence can be established:

$$\{(\boldsymbol{\pi},\mathbf{u})\in\Pi\times U \ : \ g_c(\mathbf{z}(\boldsymbol{\pi},\mathbf{u}),\boldsymbol{\pi},\mathbf{u}) = -1\} \Leftrightarrow \{(\boldsymbol{\pi},\mathbf{u})\in\Pi\times U \ : \ g_t(\mathbf{z}(\boldsymbol{\pi},\mathbf{u}),\boldsymbol{\pi},\mathbf{u}) \leq 0\},$$

with $g_t(\cdot,\cdot,\cdot) \equiv f_c^{ANN}(\cdot,\cdot,\cdot)$. By construction, g_t is continuous on its domain, and so this reformulation ensures that the semi-infinite constraint is continuous, and in turn, ensures that the convex/concave relaxations used in the subproblem of the SIPres algorithm [188] exhibit desirable convergence properties [203]. Under this equivalence, the SIP (8.3.3) is reformulated as:

$$\begin{aligned} \eta^* = & \max_{\boldsymbol{\pi}\in\Pi,\eta\in H} \eta \\ & \text{s.t. } \eta - \max\left\{\xi_{G,7}(\boldsymbol{\pi},\mathbf{u}) - G^{max}, g_t(\mathbf{z}(\boldsymbol{\pi},\mathbf{u}),\boldsymbol{\pi},\mathbf{u})\right\} \leq 0, \quad \forall \mathbf{u}\in U \\ & U = [0.35, 0.8]^2 \\ & \Pi = [0.35, 0.5]. \end{aligned} \tag{8.3.4}$$

We first solved this hybrid model using the SIPres [188] routine provided in EAGO v0.6.1 [326, 327] and using the convex/concave envelope of SiLU described in Chapter 4 [331]. The SIP was solved to an absolute tolerance of 10^{-3}. The algorithm terminated in 3 iterations, taking 2.9 CPU seconds when using the envelope of SiLU when computing relaxations. The SIPres algorithm terminated after an optimal value was found in the lower-bounding problem and the maximal value of the corresponding lower-level problem was found to be nonpositive with a value of $\eta^* = -6.6 \times 10^{-4}$. In contrast, the original method in Stuber et al. [299] provided a solution value of -5.77×10^{-3} for this case study. However, it is worth noting that the method proposed by Stuber et al. [299] has an early-termination criterion whereby the algorithm terminates with a feasible suboptimal solution as soon as robustness is verified. Thus, the solution value obtained by Stuber et al. [299] is an upper bound on the global solution. Despite this, we notice that $\eta^* > -5.77 \times 10^{-3}$, seemingly in violation of the upper bound for the full mechanistic model [299].

Since the hybrid model utilizes an ANN to approximate the original equations exhibiting numerical issues (i.e., domain violations), such discrepancies are anticipated. The level of confidence in the solution lies in the accuracy of the trained model versus the constraint satisfaction and algorithm convergence tolerances. In practice, it may be possible verify SIP feasibility of an optimal solution with respect to the full mechanistic model. However, this depends entirely on the existence and complexity of such a model. For this case, the results verify that both models ensure the robust feasibility of this operation. A performance normalization was used based on CPU single-core IPC using the Cinebench R15 (Maxon, Newbury Park, CA) single-core benchmark to enable a fair comparison of the performance of the approach in this work versus Stuber et al. [299]. The normalized results indicate a 70-fold performance improvement over the original solution time of 549.3 CPU seconds reported by Stuber et al. [299]. In this particular case, we expect this improvement to be genuine as prior comparisons of

Julia/EAGO to C++/MC++ implementations differed only by at most a factor of three [326]. However, the degree of computational performance improvement for the surrogate modeling approach relative to the original work of Stuber et al. [299] will undoubtedly be model-specific. As such, we make no broad claim of superior performance for this method. However, this example does illustrate that the use of surrogate modeling represents a viable approach to eliminate the need to apply specialized parametric interval analysis [219, 296], constraint propagation techniques [299], and implicit relaxation [300] methods when addressing bilevel optimization problems with coupling equality constraints, by replacing these models with a formulation that can be readily addressed with standard global optimization solvers.

Chapter 9

Future Opportunities

9.1 Edge-Convex and Edge-Concave Relaxations of Algorithms

The software methods developed in Chapter 3 and subsequent approaches developed in Chapters 6 and 7 fundamentally rely upon methods by which convex/concave relaxations of general nonlinear functions may be computed. Recent work by Hasan and coworkers [19, 124] detailed a novel method by which a polyhedral convex envelope is derived from edge-concave underestimators. These estimators may be composed with other convex and concave relaxations in order to generate polyhedral edge convex envelopes of general algorithms. Two preliminary theorems are given by 9.1.1 and 9.1.2 that establish rules for composition and addition.

Theorem 9.1.1 (Edge-Concave Underestimators and Edge-Convex Overestimators of Sums)**.** Let $\mathbf{D} = \{d_1, \ldots, d_m\}$ be the set of vectors such that for each edge E of a polyhedron P, D contains a vector parallel to E. Let $Z \subset R^n$ be a nonempty convex set such that $P \cap Z$ is nonempty and

$g, g^u, g^o : Z \to \mathbb{R}$ such that $g(\mathbf{z}) = g_1(\mathbf{z}) + g_2(\mathbf{z})$. Let g_1^u, g_1^o be edge-concave underestimators and overestimators of g_1 on $P \cap Z$. Similarly, suppose that g_2^u, g_2^o be edge-concave underestimators and overestimators of g_2 on $P \cap Z$. Then, $w^u, w^o : Z \to \mathbb{R}$, such that

$$g^u(\mathbf{z}) = g_1^u(\mathbf{z}) + g_2^u(\mathbf{z}), \qquad w^o(\mathbf{z}) = g_1^o(\mathbf{z}) + g_2^o(\mathbf{z}), \tag{9.1.1}$$

are, respectively, an edge-concave underestimator and an edge-concave overestimator of w on $P \cap Z$.

Proof. On the restriction of Z to all segments in P that are parallel to an edge of P, the edge-concave underestimators g_1^u and g_2^u are concave and concavity is closed under addition. Further, g^u is trivially an underestimator of g on Z. An analogous argument holds for edge-concave underestimators. □

Proposition 9.1.2 (Edge-Concave Underestimators and Edge-Convex Overestimators of Univariate Composition)**.** Let P be a polyhedron, let $Z \subset \mathbb{R}^n$ and $X \subset \mathbb{R}$ be convex sets such that $Z \cap P$ and X are nonempty. Consider the composite function $w = \phi \circ q$ where $w : Z \to \mathbb{R}$ is continuous, $\phi : X \to \mathbb{R}$, let $q(Z) \subset X$. Let $q^u : Z \to \mathbb{R}$ and $q^o : Z \to \mathbb{R}$ be edge-concave underestimators and edge-convex overestimators of q on Z, respectively with $q^u(Z) \subset X$ and $q^o(Z) \subset X$. Let $\phi^u : X \to \mathbb{R}$ and $\phi^o : X \to \mathbb{R}$ be convex and concave relaxations of ϕ on X, respectively. Let $\xi^*_{\min}$ be a point at which ϕ^u attains its minimum on X and let $\xi^*_{\max}$ be a point at which ϕ^o attains its maximum on X. Then the convex and concave relaxations are, respectively, given by

$$w^u : Z \to \mathbb{R} : \mathbf{z} \mapsto \phi^u(\mathrm{mid}(q^u(\mathbf{z}), q^o(\mathbf{z}), \xi^*_{\max})) \tag{9.1.2}$$

$$w^o : Z \to \mathbb{R} : \mathbf{z} \mapsto \phi^o(\mathrm{mid}(q^u(\mathbf{z}), q^o(\mathbf{z}), \xi^*_{\min})). \tag{9.1.3}$$

Proof. We clearly have $Q^u(\mathbf{z}) \leq Q^o(\mathbf{z})$ for all $\mathbf{z} \in Z$. We proceed with proof for each of the three cases in (9.1.3). Suppose that $\phi(q^u(\mathbf{z}), q^o(\mathbf{z}), \xi^*_{\min}) = \xi^*_{\min} \in X$, then $\phi^o(\xi^*_{\min})$ is trivially convex. Next suppose that $\phi(q^u(\mathbf{z}), q^o(\mathbf{z}), \xi^*_{\min}) = q^o(\mathbf{z})$, $\xi^*_{\min} \leq q^u(\mathbf{z}) \leq g(\mathbf{z}) \leq q^o(\mathbf{z})$. Since ϕ^o is convex it is non-decreasing where $x \geq \xi^*_{\min}$, and $q^o(\mathbf{z}) \in X$ by assumption then $\phi^o \circ q^o$ is convex. Moreover, suppose that $\phi(q^u(\mathbf{z}), q^o(\mathbf{z}), \xi^*_{\min}) = q^u(\mathbf{z})$ then $q^u(\mathbf{z}) \leq q(\mathbf{z}) \leq q^o(\mathbf{z}) \leq \xi^*_{\min}$ and ϕ^o is convex therefore it is non-increasing on $x \leq \xi^*_{\min}$,and $q^u(\mathbf{z}) \in X$ by assumption then accordingly $\phi^o \circ q^u$ is convex. An analogous proof which holds for (9.1.2) is left to the reader. □

Additional results would be required to establish composition rules for multiplication and further extend edge-convex and edge-concave to factorable programs and general functions defined by algorithms in the manner that exists for McCormick relaxations.

9.2 Composite Relaxations of Functional Forms

In Chapter 4, improved relaxations of common activation functions are detailed, while in Chapter 5 improved relaxations of the composite bilinear term are described. Further analysis of the specific functional forms represents a significant avenue for future research.

Two key functional forms that have yet to be addressed in this manner include the trilinear form and multilinear monomials. These are common to important pooling and chemical kinetic problems, as evidenced by the large body of work devoted to these forms. Crama [67] examined situations in which a standard linearization process yielded the convex envelope of the multilinear function on a unit hypercube. Shereli derived the convex envelope for a series of other classes of 0-1 multilinear functions over the unit hypercube as well as a number of special discrete sets [280]. Additional classes of bounds for multilinear terms have been detailed [68, 74, 75]. Bao et al. [22] detailed an explicit treatment of multilinear intermediates in BARON global optimizer

and provided an exhaustive summary of decomposition schemes used to recognize multilinear terms. Most recently, an adaptive partitioning scheme is detailed along with formulations for the hull of mixed-integer multilinear functions [199, 200].

One particularly interesting class of multilinear terms is that of the monomial. Monomials are known to have convex/concave envelopes that may be represented by a series of facet inducing inequalities [248]. In the case of the trilinear monomial, known expressions exist for the polyhedral convex and concave envelopes over rectangular domains [180, 181]. As such, a tight composite envelope could be developed for the polyhedral envelopes of trilinear terms in a manner that directly mirrors our work on the composite bilinear relaxations. Moreover, the theoretical work of Rikun [248] indicates that *a priori* relaxations of trilinear terms could be developed for both full-space and reduced-space approaches. Moreover, the *a priori* relaxations may be incorporated into the reverse McCormick relaxation framework to tighten relaxations of expressions involving the division operator. Direct extensions to the relaxations of multilinear terms may also be made by expanding products encountered in derivations of the envelope, and then deriving relaxations of the resulting under/overestimators in a manner that parallels the derivation detailed in Theorems 5.2.1 through 5.2.3. Alternatively, a recursive method for generating tighter relaxations of the multilinear term could be implemented using the improved bilinear relaxation defined herein.

9.3 Reduced-Space Relaxations Methods for ANNs

In Chapter 4, methods for improving relaxations of activation functions were considered. Two principal avenues for future work lie in the extension of these methods to continuous-time neural networks and deep ANNs with implicit representations. Continuous-time neural networks,

such as continuous-time recurrent neural networks, may be incorporated into global optimization formulations using modern dynamic relaxation methods [270, 272, 293, 328]. Deep ANNs with an implicit representation [88] may be addressed using fixed-point relaxation methods [300]. Particular attention should be paid to deep residual networks as the overestimation of affine relaxations computed via McCormick rules is minimal with respect to linear combinations. In any case, the further development of improved methods for constructing reduced-space relaxations of activation functions and standard layer structures that participate in ANNs is expected to lead to improved computational performance and remain an intriguing area for future research.

Bibliography

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL `https://www.tensorflow.org/`. Software available from tensorflow.org.

[2] Brian M Adams, WJ Bohnhoff, KR Dalbey, JP Eddy, MS Eldred, DM Gay, K Haskell, Patricia D Hough, and Laura P Swiler. DAKOTA, a multilevel parallel object-oriented framework for design optimization, parameter estimation, uncertainty quantification, and sensitivity analysis: version 5.0 users manual. *Sandia National Laboratories, Tech. Rep. SAND2010-2183*, 2009.

[3] Claire S. Adjiman and Christodoulos A. Floudas. Rigorous convex underestimators for general twice-differentiable problems. *Journal of Global Optimization*, 9(1):23–40, 1996. ISSN 1573-2916. doi: 10.1007/BF00121749.

[4] Claire S Adjiman, Ioannis P Androulakis, Costas D Maranas, and Christodoulos A Floudas. A global optimization method, αBB, for process design. *Computers & Chemical Engineering*, 20:S419–S424, 1996. doi: 10.1016/0098-1354(96)00080-4.

[5] Claire S Adjiman, Ioannis P Androulakis, and Christodoulos A Floudas. A global optimization method, αBB, for general twice-differentiable constrained NLPs–II. Implementation and computational results. *Computers & Chemical Engineering*, 22(9): 1159–1179, 1998. doi: 10.1016/S0098-1354(98)00218-X.

[6] Claire S Adjiman, Stefan Dallwig, Christodoulos A Floudas, and Arnold Neumaier. A global optimization method, αBB, for general twice-differentiable constrained NLPs–I.

Theoretical advances. *Computers & Chemical Engineering*, 22(9):1137–1158, 1998. doi: 10.1016/S0098-1354(98)00027-1.

[7] Ioannis G Akrotirianakis and Christodoulos A Floudas. Computational experience with a new class of convex underestimators: Box-constrained NLP problems. *Journal of Global Optimization*, 29(3):249–264, 2004. doi: 10.1023/B:JOGO.0000044768.75992.10.

[8] Faiz A. Al-Khayyal and James E. Falk. Jointly constrained biconvex programming. *Mathematics of Operations Research*, 8(2):273–286, 1983. doi: 10.1287/moor.8.2.273.

[9] Jamal Alikhania, Arash Massoudiehb, and Ujjal K. Bhowmika. GPU-Accelerated Solution of Activated Sludge Model's system of ODEs with a High Degree of Stiffness. *2016 International Conference on Computational Science and Computational Intelligence (CSCI)*, pages 555–560, 2016. doi: 10.1109/CSCI.2016.0111.

[10] G. Anandalingam and T. L. Friesz. Hierarchical optimization: An introduction. *Annals of Operations Research*, 34(1):1–11, 1992. doi: 10.1007/bf02098169.

[11] Edward Anderson and Zhaojun et al. Bai. *LAPACK Users' guide*. SIAM, 1999.

[12] Ross Anderson, Joey Huchette, Will Ma, Christian Tjandraatmadja, and Juan Pablo Vielma. Strong mixed-integer programming formulations for trained neural networks. *Mathematical Programming*, pages 1–37, 2020. doi: 10.1007/s10107-020-01474-5.

[13] Ioannis P Androulakis, Costas D Maranas, and Christodoulos A Floudas. αBB: A global optimization method for general constrained nonconvex problems. *Journal of Global Optimization*, 7(4):337–363, 1995. doi: 10.1007/BF01099647.

[14] Osyczka Andrzej and Krenich Stanislaw. *Evolutionary Algorithms for Global Optimization*, pages 267–300. Springer US, Boston, MA, 2006. ISBN 978-0-387-30927-9. doi: 10.1007/0-387-30927-6_12.

[15] Hermes R. Sant Anna, Amaro G. Barreto, Frederico W. Tavares, and Maurício B. de Souza. Machine learning model and optimization of a PSA unit for methane-nitrogen separation. *Computers & Chemical Engineering*, 104:377–391, 2017. doi: 10.1016/j.compchemeng.2017.05.006.

[16] Carmen Arévalo, Claus Führer, and Mónica Selva. A collocation formulation of multistep methods for variable step-size extensions. *Applied numerical mathematics*, 42(1-3):5–16, 2002. doi: 10.1016/S0168-9274(01)00138-6.

[17] Mordecai Avriel and Boaz Golany. *Mathematical programming for industrial engineers*, volume 20. CRC Press, New York, NY, 1996.

[18] Jaehan Bae, Hye ji Lee, Dong Hwi Jeong, and Jong Min Lee. Construction of a valid domain for a hybrid model and its application to dynamic optimization with controlled exploration. *Industrial & Engineering Chemistry Research*, 59(37):16380–16395, 2020. doi: 10.1021/acs.iecr.0c02720.

[19] Ishan Bajaj and MM Hasan. Global dynamic optimization using edge-concave underestimator. *Journal of Global Optimization*, 77(3):487–512, 2020. doi: 10.1007/s10898-020-00883-2.

[20] Xiaowei Bao, Nikolaos V Sahinidis, and Mohit Tawarmalani. Multiterm polyhedral relaxations for nonconvex, quadratically constrained quadratic programs. *Optimization Methods & Software*, 24(4-5):485–504, 2009. doi: 10.1080/10556780902883184.

[21] Xiaowei Bao, Nikolaos V Sahinidis, and Mohit Tawarmalani. Semidefinite relaxations for quadratically constrained quadratic programming: A review and comparisons. *Mathematical programming*, 129(1):129–157, 2011. doi: 10.1007/s10107-011-0462-2.

[22] Xiaowei Bao, Aida Khajavirad, Nikolaos V Sahinidis, and Mohit Tawarmalani. Global optimization of nonconvex problems with multilinear intermediates. *Mathematical Programming Computation*, 7(1):1–37, 2015. doi: 10.1007/s12532-014-0073-z.

[23] Michael Bartholomew-Biggs, Steven Brown, Bruce Christianson, and Laurence Dixon. Automatic differentiation of algorithms. *Journal of Computational and Applied Mathematics*, 124(1):171 – 190, 2000. ISSN 0377-0427. doi: https://doi.org/10.1016/S0377-0427(00)00422-2. Numerical Analysis 2000. Vol. IV: Optimization and Nonlinear Equations.

[24] Paul I Barton and Constantinos C Pantelides. Modeling of combined discrete/continuous processes. *AIChE journal*, 40(6):966–979, 1994. URL `https://doi.org/10.1002/aic.690400608`.

[25] Paul Inigo Barton and CC Pantelides. gPROMS-a combined discrete/continuous modelling environment for chemical processing systems. *Simulation Series*, 25:25–25, 1993.

[26] Pietro Belotti. Couenne: a users manual. Technical report, Technical report, Lehigh University, 2009.

[27] Pietro Belotti, Jon Lee, Leo Liberti, Francois Margot, and Andreas Wächter. Branching and bounds tightening techniques for non-convex MINLP. *Optimization Methods & Software*, 24(4-5):597–634, 2009. doi: 10.1080/10556780903087124.

[28] Omar Ben-Ayed. Bilevel linear programming. *Computers & Operations Research*, 20(5): 485–501, 1993. doi: 10.1016/0305-0548(93)90013-9.

[29] Aharon Ben-Tal, Laurent El Ghaoui, and Arkadi Nemirovski. *Robust optimization*, volume 28. Princeton university press, 2009.

[30] Harold P Benson. Separable concave minimization via partial outer approximation and branch and bound. *Operations Research Letters*, 9(6):389–394, 1990. doi: 10.1016/0167-6377(90)90059-E.

[31] B Wayne Bequette. *Process control: modeling, design, and simulation*. Prentice Hall Professional, 2003.

[32] Dimitris Bertsimas, Xavier Boix, Kimberly Villalobos Carballo, and Dick den Hertog. A robust optimization approach to deep learning. *arXiv preprint arXiv:2112.09279*, 2021.

[33] Martin Berz and Georg Hoffstätter. Computation and application of Taylor polynomials with interval remainder bounds. *Reliable Computing*, 4(1):83–97, 1998. doi: 10.1023/A:1009958918582.

[34] Martin Berz and Kyoko Makino. Performance of Taylor model methods for validated integration of ODEs. In *International Workshop on Applied Parallel Computing*, pages 65–73. Springer, 2004. doi: 10.1007/11558958_8.

[35] Martin Berz and Kyoko Makino. Suppression of the wrapping effect by Taylor model-based verified integrators: Long-term stabilization by shrink wrapping. *Int. J. Diff. Eq. Appl*, 10:385–403, 2005.

[36] Jeff Bezanson, Alan Edelman, Stefan Karpinski, and Viral B Shah. Julia: A fresh approach to numerical computing. *SIAM review*, 59(1):65–98, 2017. doi: 10.1137/141000671.

[37] B. Bhattacharjee, P. Lemonidis, W.H. Green Jr., and P.I. Barton. Global solution of semi-infinite programs. *Mathematical Programming*, 103(2):283–307, 2005. ISSN 1436-4646. doi: 10.1007/s10107-005-0583-6.

[38] Lorenz T Biegler. Solution of dynamic optimization problems by successive quadratic programming and orthogonal collocation. *Computers and chemical engineering*, 8(3-4): 243–247, 1984. doi: 10.1016/0098-1354(84)87012-X.

[39] Lorenz T Biegler. *Nonlinear programming: concepts, algorithms, and applications to chemical processes*. SIAM, 2010.

[40] L Susan Blackford, Antoine Petitet, Roldan Pozo, Karin Remington, R Clint Whaley, James Demmel, Jack Dongarra, Iain Duff, Sven Hammarling, Greg Henry, et al. An updated set of basic linear algebra subprograms (blas). *ACM Transactions on Mathematical Software*, 28(2):135–151, 2002. doi: 10.1145/567806.567807.

[41] Franco Blanchini and Stefano Miani. *Set-theoretic methods in control*. Springer, 2008.

[42] Jerry W Blankenship and James E Falk. Infinitely constrained optimization problems. *Journal of Optimization Theory and Applications*, 19(2):261–281, 1976. doi: 10.1007/BF00934096.

[43] Hans Georg Bock and Karl-Josef Plitt. A multiple shooting algorithm for direct solution of optimal control problems. *IFAC Proceedings Volumes*, 17(2):1603–1608, 1984. doi: 10.1016/S1474-6670(17)61205-9.

[44] Agustín Bompadre and Alexander Mitsos. Convergence rate of McCormick relaxations. *Journal of Global Optimization*, 52(1):1–28, 2012. ISSN 1573-2916. doi: 10.1007/s10898-011-9685-2.

[45] Agustín Bompadre, Alexander Mitsos, and Benoît Chachuat. Convergence analysis of Taylor models and McCormick-Taylor models. *Journal of Global Optimization*, 57(1): 75–114, 2013. doi: 10.1007/s10898-012-9998-9.

[46] D Bongartz, J Najman, S Sass, and A Mitsos. MAiNGO: McCormick based algorithm for mixed integer nonlinear global optimization. Technical report, RWTH Aachen, 2018.

[47] Dominik Bongartz. *Deterministic global flowsheet optimization for the design of energy conversion processes*. PhD thesis, Dissertation, Rheinisch-Westfälische Technische Hochschule Aachen, 2020, 2020.

[48] Dominik Bongartz and Alexander Mitsos. Deterministic global optimization of process flowsheets in a reduced space using McCormick relaxations. *Journal of Global Optimization*, 69(4):761–796, Dec 2017. ISSN 1573-2916. doi: 10.1007/s10898-017-0547-4.

[49] Dominik Bongartz and Alexander Mitsos. Deterministic global flowsheet optimization: Between equation-oriented and sequential-modular methods. *AIChE Journal*, 65(3): 1022–1034, 2019. doi: 10.1002/aic.16507.

[50] Dominik Bongartz, Jaromił Najman, and Alexander Mitsos. Deterministic global optimization of steam cycles using the IAPWS-IF97 model. *Optimization and Engineering*, 10, 2020. doi: 10.1007/s11081-020-09502-1.

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

[52] Florian Bünger. Shrink wrapping for Taylor models revisited. *Numerical Algorithms*, 78 (4):1001–1017, 2018. doi: 10.1007/s11075-017-0410-1.

[53] José A. Caballero and Ignacio E. Grossmann. An algorithm for the use of surrogate models in modular flowsheet optimization. *AIChE Journal*, 54(10):2633–2650, 2008. doi: 10.1002/aic.11579.

[54] Huiyi Cao, Yingkai Song, and Kamil A Khan. Convergence of subtangent-based relaxations of nonlinear programs. *Processes*, 7(4):221, 2019. doi: 10.3390/pr7040221.

[55] Adrian Caspari, Hatim Djelassi, Adel Mhamdi, Lorenz T Biegler, and Alexander Mitsos. Semi-infinite programming yields optimal disturbance model for offset-free nonlinear model predictive control. *Journal of Process Control*, 101:35–51, 2021. doi: 10.1016/j.jprocont.2021.03.005.

[56] Filippo Castiglione and Benedetto Piccoli. Cancer immunotherapy, mathematical modeling and optimal control. *Journal of theoretical Biology*, 247(4):723–732, 2007. doi: 10.1016/j.jtbi.2007.04.003.

[57] Ali N. Celik and Mohan Kolhe. Generalized feed-forward based method for wind energy prediction. *Applied Energy*, 101:582–588, 2013. doi: 10.1016/j.apenergy.2012.06.040.

[58] Benoıt C. Chachuat. MC++: Toolkit for Construction, Manipulation and Bounding of Factorable Functions, 2022. URL `https://omega-icl.github.io/mcpp/`.

[59] Jiahao Chen and Jarrett Revels. Robust benchmarking in noisy environments. *arXiv e-prints*, art. arXiv:1608.04295, Aug 2016.

[60] Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David Duvenaud. Neural ordinary differential equations. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pages 6572–6583, 2018. doi: 10.5555/3327757.3327764.

[61] François Chollet et al. Keras, 2015. URL `https://keras.io`. Software available from keras.io, accessed on 2021-06-28.

[62] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289*, 2015.

[63] Thomas Coleman, Mary Ann Branch, and Andrew Grace. Optimization toolbox. *For Use with MATLAB. Users Guide for MATLAB 5, Version 2, Release II*, 1999.

[64] Joao Luiz Dihl Comba and Jorge Stolfi. Affine arithmetic and its applications to computer graphics. In *Proceedings of VI SIBGRAPI (Brazilian Symposium on Computer Graphics and Image Processing)*, pages 9–18, 1993.

[65] George F Corliss and Robert Rihm. Validating an a priori enclosure using high-order Taylor series. *MATHEMATICAL RESEARCH*, 90:228–238, 1996.

[66] CPLEX, IBM ILOG. CPLEX Optimizer: User's Manual. urlhttp://www-01.ibm.com/software/integration/optimization/cplex-optimizer/, Last 2017.

[67] Yves Crama. Concave extensions for nonlinear 0–1 maximization problems. *Mathematical Programming*, 61(1-3):53–60, 1993. doi: 10.1007/BF01582138.

[68] Yves Crama and Elisabeth Rodríguez-Heck. A class of valid inequalities for multilinear 0–1 optimization problems. *Discrete Optimization*, 25:28–47, 2017. doi: 10.1016/j.disopt.2017.02.001.

[69] Guevara Neri Maria Cristina, Vianey Guadalupe Cruz Sanchez, Osslan Osiris Vergara Villegas, Manuel Nandayapa, Humberto de Jesus Ochoa Dominguez, and Juan Humberto Sossa Azuela. Study of the effect of combining activation functions in a convolutional neural network. *IEEE Latin America Transactions*, 19(5):844–852, 2021. doi: 10.1109/TLA.2021.9448319.

[70] CF Curtiss and Joseph O Hirschfelder. Integration of stiff equations. *Proceedings of the National Academy of Sciences of the United States of America*, 38(3):235, 1952. doi: 10.1073/pnas.38.3.235.

[71] G Darboux. Sur les développements en série des fonctions d'une seule variable. *Journal de Mathématiques pures et appliquées*, 2:291–312, 1876.

[72] Luiz Henrique de Figueiredo and Jorge Stolfi. Affine arithmetic: Concepts and applications. *Numerical Algorithms*, 37:147–158, 2004. ISSN 1017-1398. doi: 10.1023/b:numa.0000049462.70970.b6.

[73] Lucas Guedes De Oliveira, Anderson Paulo de Paiva, Pedro Paulo Balestrassi, João Roberto Ferreira, Sebastião Carlos da Costa, and Paulo Henrique da Silva Campos. Response surface methodology for advanced manufacturing technology optimization: theoretical fundamentals, practical guidelines, and survey literature review. *The International Journal of Advanced Manufacturing Technology*, 104(5):1785–1837, 2019. doi: 10.1007/s00170-019-03809-9.

[74] Alberto Del Pia and Aida Khajavirad. On decomposability of Multilinear sets. *Mathematical Programming*, 170(2):387–415, 2018. doi: 10.1007/s10107-017-1158-z.

[75] Alberto Del Pia and Aida Khajavirad. The multilinear polytope for acyclic hypergraphs. *SIAM Journal on Optimization*, 28(2):1049–1076, 2018. doi: 10.1137/16M1095998.

[76] Stephan Dempe. Annotated bibliography on bilevel programming and mathematical programs with equilibrium constraints. *Optimization*, 52(3):333–359, 2003. doi: 10.1080/0233193031000149894.

[77] Hatim Djelassi and Alexander Mitsos. A hybrid discretization algorithm with guaranteed feasibility for the global solution of semi-infinite programs. *Journal of Global Optimization*, 68(2):227–253, 2017. doi: 10.1007/s10898-016-0476-7.

[78] Hatim Djelassi, Moll Glass, and Alexander Mitsos. Discretization-based algorithms for generalized semi-infinite and bilevel programs with coupling equality constraints. *Journal of Global Optimization*, 75(2):341–392, 2019. doi: 10.1007/s10898-019-00764-3.

[79] Hatim Djelassi, Alexander Mitsos, and Oliver Stein. Recent advances in nonconvex semi-infinite programming: Applications and algorithms. *EURO Journal on Computational Optimization*, 9:100006, 2021. doi: 10.1016/j.ejco.2021.100006.

[80] Elizabeth D Dolan and Jorge J Moré. Benchmarking optimization software with performance profiles. *Mathematical programming*, 91(2):201–213, 2002. doi: 10.1007/s101070100263.

[81] Ferenc Domes and Arnold Neumaier. Constraint propagation on quadratic constraints. *Constraints*, 15(3):404–429, 2010. doi: 10.1007/s10601-009-9076-1.

[82] Manuel Dornier, Martine Decloux, Gilles Trystram, and Andr M. Lebert. Interest of neural networks for the optimization of the crossflow filtration process. *LWT - Food Science and Technology*, 28(3):300–309, 1995. doi: 10.1016/s0023-6438(95)94364-1.

[83] K. Du and R. B. Kearfott. The cluster problem in multivariate global optimization. *Journal of Global Optimization*, 5(3):253–265, oct 1994. doi: 10.1007/bf01096455.

[84] Iain Dunning, Joey Huchette, and Miles Lubin. JuMP: A modeling language for mathematical optimization. *SIAM Review*, 59(2):295–320, 2017. doi: 10.1137/15M1020575.

[85] Steffen Eger, Paul Youssef, and Iryna Gurevych. Is it time to swish? Comparing deep learning activation functions across NLP tasks. *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 2018. doi: 10.18653/v1/d18-1472.

[86] Ruediger Ehlers. Formal verification of piece-wise linear feed-forward neural networks. In *International Symposium on Automated Technology for Verification and Analysis*, pages 269–286. Springer, 2017. doi: 10.1007/978-3-319-68167-2_19.

[87] Pieter Eijgenraam. *The solution of initial value problems using interval arithmetic: formulation and analysis of an algorithm*. Centrum Voor Wiskunde en Informatica, 1981.

[88] Laurent El Ghaoui, Fangda Gu, Bertrand Travacca, Armin Askari, and Alicia Tsai. Implicit deep learning. *SIAM Journal on Mathematics of Data Science*, 3(3):930–958, 2021. doi: 10.1137/20M1358517.

[89] Stefan Elfwing, Eiji Uchibe, and Kenji Doya. Expected energy-based restricted boltzmann machine for classification. *Neural Networks*, 64:29–38, 2015. doi: 10.1016/j.neunet.2014.09.006.

[90] Stefan Elfwing, Eiji Uchibe, and Kenji Doya. Sigmoid-weighted linear units for neural network function approximation in reinforcement learning. *Neural Networks*, 107:3–11, 2018. doi: 10.1016/j.neunet.2017.12.012.

[91] David L Elliott. A better activation function for artificial neural networks. Technical report, Institute for Systems Research, 1993. URL http://hdl.handle.net/1903/5355.

[92] Process Systems Enterprise. gPROMS, 2022. URL http://www.psenterprise.com/gproms.

[93] Thomas G. W. Epperly and Efstratios N. Pistikopoulos. A reduced space branch and bound algorithm for global optimization. *Journal of Global Optimization*, 11(3):287–311, 1997. doi: 10.1023/A:1008212418949.

[94] William R Esposito and Christodoulos A Floudas. Deterministic global optimization in isothermal reactor network synthesis. *Journal of Global Optimization*, 22(1):59–95, 2002. doi: 10.1023/A:1013842726210.

[95] Gennady F. and Khang T Nguyen. *Intel Math Kernel Library 2019 Update 2 Release Notes*, 2019. URL https://software.intel.com/en-us/mkl.

[96] Gennady F. and Khang T Nguyen. *Intel Math Kernel Library 2022 Update 1 Release Notes*, 2019. URL https://software.intel.com/en-us/mkl.

[97] Ismail Fahmi and Selen Cremaschi. Process synthesis of biodiesel production plant using artificial neural networks as the surrogate models. *Computers & Chemical Engineering*, 46:105–123, 2012. doi: 10.1016/j.compchemeng.2012.06.006.

[98] Steffen Fahr, Alexander Mitsos, and Dominik Bongartz. Simultaneous deterministic global flowsheet optimization and heat integration: Comparison of formulations. *Computers & Chemical Engineering*, 162:107790, 2022. ISSN 0098-1354. doi: https://doi.org/10.1016/j.compchemeng.2022.107790. URL https://www.sciencedirect.com/science/article/pii/S0098135422001314.

[99] J. E. Falk and R. M. Soland. An Algorithm for Separable Nonconvex Programming Problems. *Management Science*, 15(9):550–569, 1969. doi: 10.1287/mnsc.15.9.550.

[100] James E Falk and Karla Hoffman. A nonconvex max-min problem. *Naval Research Logistics Quarterly*, 24(3):441–450, 1977. doi: 10.1002/nav.3800240307.

[101] James E Falk and Karla L Hoffman. Concave minimization via collapsing polytopes. *Operations research*, 34(6):919–929, 1986. doi: 10.1287/opre.34.6.919.

[102] Lucas J. Fernández-Alcázar, Rodion Kononchuk, and Tsampikos Kottos. Enhanced energy harvesting near exceptional points in systems with (pseudo-)PT-symmetry. *Communications Physics*, 4(1), 2021. doi: 10.1038/s42005-021-00577-5.

[103] Matteo Fischetti and Jason Jo. Deep neural networks and mixed integer linear optimization. *Constraints*, 23(3):296–309, 2018. doi: 10.1007/s10601-018-9285-6.

[104] Christodoulos A. Floudas and Oliver Stein. The adaptive convexification algorithm: A feasible point method for semi-infinite programming. *SIAM Journal on Optimization*, 18 (4):1187–1208, 2008. doi: 10.1137/060657741.

[105] Christodoulos A Floudas, John L Klepeis, and Panos M Pardalos. Global optimization approaches in protein folding and peptide docking. *Mathematical support for molecular biology*, 47:141–172, 1998.

[106] Robert Fourer, David M Gay, and Brian W Kernighan. A modeling language for mathematical programming. *Management Science*, 36(5):519–554, 1990. doi: 10.1287/mnsc.36.5.519.

[107] Fenila Francis-Xavier, Fabian Kubannek, and René Schenkendorf. Hybrid process models in electrochemical syntheses under deep uncertainty. *Processes*, 9(4):704, 2021. doi: 10.3390/pr9040704.

[108] Ken-ichi Funahashi and Yuichi Nakamura. Approximation of dynamical systems by continuous time recurrent neural networks. *Neural networks*, 6(6):801–806, 1993. doi: 10.1016/S0893-6080(05)80125-X.

[109] Walter Gautschi. *Numerical Analysis*. Springer Science & Business Media, New York, 2012.

[110] S Gerschgorin. Über die abrenzung der eigenwerte eiven matrix. *Izv. Akad. Nauk SSSR Ser. Mat*, 7:749–754, 1931.

[111] Ambros M. Gleixner, Timo Berthold, Benjamin Müller, and Stefan Weltge. Three enhancements for optimization-based bound tightening. *Journal of Global Optimization*, 67(4):731–757, Apr 2017. ISSN 1573-2916. doi: 10.1007/s10898-016-0450-4.

[112] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of The Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 249–256, 2010. URL `https://proceedings.mlr.press/v9/glorot10a.html`.

[113] Michael Grant, Stephen Boyd, and Yinyu Ye. *Disciplined convex programming*, pages 155–210. Springer, Boston, MA, 2006. doi: 10.1007/0-387-30528-9_7.

[114] Andreas Griewank and Andrea Walther. *Evaluating derivatives: principles and techniques of algorithmic differentiation*, volume 105. Siam, 2008.

[115] Bjarne Grimstad and Henrik Andersson. ReLU networks as surrogate models in mixed-integer linear programs. *Computers & Chemical Engineering*, 131:106580, 2019. doi: 10.1016/j.compchemeng.2019.106580.

[116] Zeynep H Gümüş and Christodoulos A Floudas. Global optimization of nonlinear bilevel programming problems. *Journal of Global Optimization*, 20(1):1–31, 2001. doi: 10.1023/A:1011268113791.

[117] Gurobi Optimization, LLC. Gurobi optimizer reference manual, 2020. URL `http://www.gurobi.com`.

[118] Eldad Haber and Lars Ruthotto. Stable architectures for deep neural networks. *Inverse problems*, 34(1):014004, 2017. doi: 10.1088/1361-6420/aa9a90.

[119] Ernst Hairer and Gerhard Wanner. *Solvig Ordinary Differential Equations II: Stiff and Differential-Algebraic Problems*. Springer, Heidelberg, 1991.

[120] K. P. Halemane and I. E. Grossmann. Optimal process design under uncertainty. *AIChE Journal*, 29(3):425–433, 1983. ISSN 1547-5905. doi: 10.1002/aic.690290312.

[121] William E. Hart, Jean-Paul Watson, and David L. Woodruff. Pyomo: modeling and solving mathematical programs in Python. *Mathematical Programming Computation*, 3(3):219, Aug 2011. ISSN 1867-2957. doi: 10.1007/s12532-011-0026-8.

[122] Stuart M Harwood and Paul I Barton. Efficient polyhedral enclosures for the reachable set of nonlinear control systems. *Mathematics of Control, Signals, and Systems*, 28(1):8, 2016. doi: 10.1007/s00498-015-0153-2.

[123] Stuart M Harwood and Paul I Barton. Affine relaxations for the solutions of constrained parametric ordinary differential equations. *Optimal Control Applications and Methods*, 39 (2):427–448, 2018. doi: 10.1002/oca.2323.

[124] MM Hasan. An edge-concave underestimator for the global optimization of twice-differentiable nonconvex problems. *Journal of Global Optimization*, 71(4):735–752, 2018. doi: 10.1007/s10898-018-0646-x.

[125] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 1026–1034, Santiago, Chile, 2015. IEEE. doi: 10.1109/iccv.2015.123.

[126] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016. doi: 10.1109/CVPR.2016.90.

[127] Taotao He and Mohit Tawarmalani. A new framework to relax composite functions in nonlinear programs. *Mathematical Programming*, pages 1–40, 2020. doi: 10.1007/s10107-020-01541-x.

[128] Taha Hossein Hejazi, Mirmehdi Seyyed-Esfahani, and Masoud Mahootchi. Quality chain design and optimization by multiple response surface methodology. *The International Journal of Advanced Manufacturing Technology*, 68(1-4):881–893, 2013. doi: 10.1007/s00170-013-4950-9.

[129] Carlos A. Henao and Christos T. Maravelias. Surrogate-based superstructure optimization framework. *AIChE Journal*, 57(5):1216–1232, 2010. doi: 10.1002/aic.12341.

[130] Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (GELUs). *arXiv preprint*, 2016.

[131] Mogens Henze, Willi Gujer, Takahashi Mino, Tomonori Matsuo, Mark C Wentzel, Gerrit v R Marais, and Mark Van Loosdrecht. Activated sludge model no. 2d, ASM2d. *Water Science and Technology*, 39(1):165–182, 1999. doi: 10.1016/S0273-1223(98)00829-4.

[132] M Ch Hermite and M Borchardt. Sur la formule d'interpolation de lagrange. *Journal für die reine und angewandte Mathematik (Crelles Journal)*, 1878(84):70–79, 1878.

[133] Jean-Baptiste Hiriart-Urruty and Claude Lemaréchal. *Fundamentals of Convex Analysis*. Springer Science & Business Media, Heidelberg, 2004. doi: 10.1007/978-3-642-56468-0.

[134] Milan Hladík. On the efficient Gerschgorin inclusion usage in the global optimization αBB method. *Journal of Global Optimization*, 61(2):235–253, 2015. doi: 10.1007/s10898-014-0161-7.

[135] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, 1989. doi: 10.1016/0893-6080(89)90020-8.

[136] Reiner Horst and Hoang Tuy. *Global optimization: Deterministic approaches*. Springer Science & Business Media, 2013.

[137] Boris Houska, Filip Logist, Jan Van Impe, and Moritz Diehl. Robust optimization of nonlinear dynamic systems with application to a jacketed tubular reactor. *Journal of Process Control*, 22(6):1152–1160, 2012. doi: 10.1016/j.jprocont.2012.03.008.

[138] Boris Houska, Mario E Villanueva, and BenoIt Chachuat. Stable set-valued integration of nonlinear dynamic systems using affine set-parameterizations. *SIAM Journal on Numerical Analysis*, 53(5):2307–2328, 2015. doi: 10.1137/140976807.

[139] Guang-Bin Huang, Qin-Yu Zhu, and Chee-Kheong Siew. Extreme learning machine: theory and applications. *Neurocomputing*, 70(1-3):489–501, 2006. doi: 10.1016/j.neucom.2005.12.126.

[140] Haitao Huang, Claire S Adjiman, and Nilay Shah. Quantitative framework for reliable safety analysis. *AIChE Journal*, 48(1):78–96, 2002. doi: 10.1002/aic.690480110.

[141] Xiaowei Huang, Marta Kwiatkowska, Sen Wang, and Min Wu. Safety verification of deep neural networks. In *International conference on computer aided verification*, pages 3–29. Springer, 2017. doi: 10.1007/978-3-319-63387-9_1.

[142] Mohamed Azlan Hussain. Review of the applications of neural networks in chemical process control — simulation and online implementation. *Artificial Intelligence in Engineering*, 13(1):55–68, 1999. doi: 10.1016/s0954-1810(98)00011-9.

[143] IEEE. IEEE Standard for Interval Arithmetic. *IEEE Std 1788-2015*, pages 1–97, June 2015. doi: 10.1109/IEEESTD.2015.7140721.

[144] Christian Jansson. Quasiconvex relaxations based on interval arithmetic. *Linear Algebra and its Applications*, 324(1-3):27–53, 2001. doi: 10.1016/S0024-3795(00)00295-0.

[145] Olaf Kahrs and Wolfgang Marquardt. The validity domain of hybrid models and its application in process optimization. *Chemical Engineering and Processing: Process Intensification*, 46(11):1054–1066, 2007. doi: 10.1016/j.cep.2007.02.031.

[146] Rohit Kannan and Paul I. Barton. The cluster problem in constrained global optimization. *Journal of Global Optimization*, 69(3):629–676, Nov 2017. ISSN 1573-2916. doi: 10.1007/s10898-017-0531-z.

[147] N Kazazakis and Claire S Adjiman. Arbitrarily tight αBB underestimators of general non-linear functions over sub-optimal domains. *Journal of Global Optimization*, 71(4): 815–844, 2018. doi: 10.1007/s10898-018-0632-3.

[148] Ralph Baker Kearfott, Jessie Castille, and Gaurav Tyagi. A general framework for convexity analysis in deterministic global optimization. *Journal of Global Optimization*, 56(3):765–785, 2013. doi: https://doi.org/10.1007/s10898-012-9905-4.

[149] Aida Khajavirad and Nikolaos V Sahinidis. A hybrid LP/NLP paradigm for global optimization relaxations. *Mathematical Programming Computation*, 10(3):383–421, 2018. doi: 10.1007/s12532-018-0138-5.

[150] Aida Khajavirad, Jeremy J. Michalek, and Nikolaos V. Sahinidis. Relaxations of factorable functions with convex-transformable intermediates. *Mathematical Programming*, 144(1): 107–140, Apr 2014. ISSN 1436-4646. doi: 10.1007/s10107-012-0618-8.

[151] Kamil A. Khan, Harry A. J. Watson, and Paul I. Barton. Differentiable McCormick relaxations. *Journal of Global Optimization*, pages 1–43, 2016. ISSN 1573-2916. doi: 10.1007/s10898-016-0440-6.

[152] Kamil A. Khan, Matthew Wilhelm, Matthew D. Stuber, Huiyi Cao, Harry A. J. Watson, and Paul I. Barton. Corrections to: Differentiable McCormick relaxations. *Journal of Global Optimization*, 70(3):705–706, Mar 2018. ISSN 1573-2916. doi: 10.1007/s10898-017-0601-2.

[153] Günter Klambauer, Thomas Unterthiner, Andreas Mayr, and Sepp Hochreiter. Self-normalizing neural networks. In *Advances in Neural Information Processing Systems*, pages 971–980, 2017. doi: https://dl.acm.org/doi/10.5555/3294771.3294864.

[154] R. Krawczyk. Interval iterations for including a set of solutions. *Computing*, 32(1):13–31, Mar 1984. ISSN 1436-5057. doi: 10.1007/BF02243016.

[155] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. ImageNet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2017. doi: 10.1145/3065386.

[156] Ole Kröger, Carleton Coffrin, Hassan Hijazi, and Harsha Nagarajan. Juniper: An Open-Source Nonlinear Branch-and-Bound Solver in Julia. In *International Conference on the Integration of Constraint Programming, Artificial Intelligence, and Operations Research*, pages 377–386. Springer, 2018. doi: 10.1007/978-3-319-93031-2_2.

[157] Fred T Krogh. Recurrence relations for computing with modified divided differences. *Mathematics of Computation*, 33(148):1265–1271, 1979. doi: 10.2307/2006460.

[158] Jan Kronqvist, Ruth Misener, and Calvin Tsay. Between steps: Intermediate relaxations between big-M and convex hull formulations. In *International Conference on Integration of Constraint Programming, Artificial Intelligence, and Operations Research*, pages 299–314. Springer, 2021. doi: 10.1007/978-3-030-78230-6_19.

[159] Gustav Larsson, Michael Maire, and Gregory Shakhnarovich. Fractalnet: Ultra-deep neural networks without residuals. *arXiv preprint arXiv:1605.07648*, 2016.

[160] Can Li and Ignacio E Grossmann. A review of stochastic programming methods for optimization of process systems under uncertainty. *Frontiers in Chemical Engineering*, page 34, 2021. doi: 10.3389/fceng.2020.622241.

[161] Youdong Lin and Mark A. Stadtherr. Validated solutions of initial value problems for parametric odes. *Applied Numerical Mathematics*, 57(10):1145 – 1162, 2007. ISSN 0168-9274. doi: https://doi.org/10.1016/j.apnum.2006.10.006.

[162] Carl-Fredrik Lindberg. *Control and estimation strategies applied to the activated sludge process*. Uppsala University Stockholm, Sweden, 1997.

[163] Rudolf J Lohner. Computation of guaranteed enclosures for the solutions of ordinary initial and boundary value problems. In *Institute of mathematics and its applications conference series*, volume 39, pages 425–425. Oxford University Press, 1992.

[164] Rudolf J Lohner. On the ubiquity of the wrapping effect in the computation of error bounds. In *Perspectives on enclosure methods*, pages 201–216. Springer, 2001. doi: 10.1007/978-3-7091-6282-8_12.

[165] Lu Lu, Yeonjong Shin, Yanhui Su, and George Em Karniadakis. Dying ReLU and initialization: Theory and numerical examples. *Communications in Computational Physics*, 28(5):1671–1706, 2020. doi: 10.4208/cicp.OA-2020-0165.

[166] Yiping Lu, Aoxiao Zhong, Quanzheng Li, and Bin Dong. Beyond finite layer neural networks: Bridging deep architectures and numerical differential equations. In *International Conference on Machine Learning*, pages 3276–3285. PMLR, 2018.

[167] Luis Benet, David P. Sanders. Validatednumerics.jl, 2022. URL `https://github.com/JuliaIntervals/ValidatedNumerics.jl`.

[168] Kyoko Makino and Martin Berz. Suppression of the wrapping effect by Taylor model-based verified integrators: Long-term stabilization by preconditioning. *International Journal of Differential Equations and Applications*, 10(4), 2011.

[169] Costas D Maranas and Christodoulos A Floudas. Finding all solutions of nonlinearly constrained systems of equations. *Journal of Global Optimization*, 7(2):143–182, 1995. doi: 10.1007/BF01097059.

[170] Andrzej Marciniak. Multistep interval methods of nyström and milne-simpson types. *Computational Methods in Science and Technology*, 13(1):23–40, 2007. doi: 10.12921/cmst.2007.13.01.23-29.

[171] Andrzej Marciniak. On multistep interval methods for solving the initial value problem. *Journal of Computational and Applied Mathematics*, 199(2):229–237, 2007. doi: 10.1016/j.cam.2005.07.036.

[172] Andrzej Marciniak. An interval version of the Crank-Nicolson method–the first approach. In *International Workshop on Applied Parallel Computing*, pages 120–126. Springer, 2010. doi: 10.1007/978-3-642-28145-7_12.

[173] Andrzej Marciniak and Malgorzata A Jankowska. Interval versions of Milnes multistep methods. *Numerical Algorithms*, 79(1):87–105, 2018. doi: 10.1007/s11075-017-0429-3.

[174] Andrzej Marciniak, Malgorzata A Jankowska, and Tomasz Hoffmann. On interval predictor-corrector methods. *Numerical Algorithms*, 75(3):777–808, 2017. doi: 10.1007/s11075-016-0220-x.

[175] Rory B Martin. Optimal control drug scheduling of cancer chemotherapy. *Automatica*, 28 (6):1113–1123, 1992. doi: 10.1016/0005-1098(92)90054-J.

[176] Bruce A McCarl, Alex Meeraus, Paul van der Eijk, Michael Bussieck, Steven Dirkse, Pete Steacy, and Franz Nelissen. GAMS user guide. *GAMS Development Corporation*, 2014.

[177] Garth P. McCormick. Computability of global solutions to factorable nonconvex programs: Part i — convex underestimating problems. *Mathematical Programming*, 10(1):147–175, 1976. ISSN 1436-4646. doi: 10.1007/BF01580665.

[178] Larry Medsker and Lakhmi C Jain. *Recurrent Neural Networks: Design and Applications*. CRC press, Boca Raton, 1999. doi: 10.1201/9781003040620.

[179] Frdric Messine. Extensions of affine arithmetic: Application to unconstrained global optimization. *Journal of Universal Computer Science*, 8(11):992–1015, nov 2002. doi: 10.3217/jucs-008-11-0992.

[180] Clifford A Meyer and Christodoulos A Floudas. Trilinear monomials with mixed sign domains: Facets of the convex and concave envelopes. *Journal of Global Optimization*, 29 (2):125–155, 2004. doi: 10.1023/B:JOGO.0000042112.72379.e6.

[181] Clifford A Meyer and Christodoulos A Floudas. Trilinear monomials with positive or negative domains: Facets of the convex and concave envelopes. In *Frontiers in global optimization*, pages 327–352. Springer, 2004. doi: 10.1007/978-1-4613-0251-3_18.

[182] Clifford A Meyer and Christodoulos A Floudas. Convex underestimation of twice continuously differentiable functions by piecewise quadratic perturbation: spline αbb underestimators. *Journal of Global Optimization*, 32(2):221–258, 2005. doi: 10.1007/s10898-004-2704-9.

[183] Ruth Misener and Christodoulos A Floudas. Global optimization of large-scale generalized pooling problems: quadratically constrained MINLP models. *Industrial & Engineering Chemistry Research*, 49(11):5424–5438, 2010. doi: 10.1021/ie100025e.

[184] Ruth Misener and Christodoulos A Floudas. GloMIQO: Global mixed-integer quadratic optimizer. *Journal of Global Optimization*, 57(1):3–50, 2013. doi: 10.1007/s10898-012-9874-7.

[185] Ruth Misener and Christodoulos A Floudas. ANTIGONE: algorithms for continuous/integer global optimization of nonlinear equations. *Journal of Global Optimization*, 59(2-3):503–526, 2014. doi: 10.1007/s10898-014-0166-2.

[186] Ruth Misener, James B Smadbeck, and Christodoulos A Floudas. Dynamically generated cutting planes for mixed-integer quadratically constrained quadratic programs and their incorporation into GloMIQO 2.0. *Optimization Methods and Software*, 30(1):215–249, 2015. doi: 10.1080/10556788.2014.916287.

[187] Miten Mistry and Ruth Misener. Optimising heat exchanger network synthesis using convexity properties of the logarithmic mean temperature difference. *Computers & Chemical Engineering*, 94:1–17, 2016. doi: 10.1016/j.compchemeng.2016.07.001.

[188] Alexander Mitsos. Global optimization of semi-infinite programs via restriction of the right-hand side. *Optimization*, 60(10-11):1291–1308, 2011. doi: 10.1080/02331934.2010.527970.

[189] Alexander Mitsos and Angelos Tsoukalas. Global optimization of generalized semi-infinite programs via restriction of the right-hand side. *Journal of Global Optimization*, 61(1): 1–17, 2015. doi: 10.1080/02331934.2010.527970.

[190] Alexander Mitsos, Panayiotis Lemonidis, and Paul I. Barton. Global solution of bilevel programs with a nonconvex inner program. *Journal of Global Optimization*, 42(4): 475–513, 2008. doi: 10.1007/s10898-007-9260-z.

[191] Alexander Mitsos, Benoît Chachuat, and Paul I Barton. McCormick-based relaxations of algorithms. *SIAM Journal on Optimization*, 20(2):573–601, 2009. doi: 10.1137/080717341.

[192] Ramon E Moore. Interval arithmetic and automatic error analysis in digital computing. *Ph. D. Dissertation, Department of Mathematics, Stanford University*, 1962.

[193] Ramon E Moore, R Baker Kearfott, and Michael J Cloud. *Introduction to interval analysis*, volume 110. Siam, 2009.

[194] R.E. Moore. *Methods and Applications of Interval Analysis*. Studies in Applied and Numerical Mathematics. Society for Industrial and Applied Mathematics (SIAM, 3600 Market Street, Floor 6, Philadelphia, PA 19104), 1979. ISBN 9781611970906.

[195] APS Mosek. The mosek optimization software. *Online at http://www. mosek. com*, 54(2-1): 5, 2010.

[196] Hossein Mostaan, Morteza Shamanian, and Mehdi Safari. Process analysis and optimization for fracture stress of electron beam welded ultra-thin feco-v foils. *The*

International Journal of Advanced Manufacturing Technology, 87(1):1045–1056, 2016. doi: 10.1007/s00170-016-8553-0.

[197] Andrew Nader and Danielle Azar. Searching for activation functions using a self-adaptive evolutionary algorithm. In *Proceedings of the 2020 Genetic and Evolutionary Computation Conference Companion*, pages 145–146, 2020. doi: 10.1145/3377929.3389942.

[198] Harsha Nagarajan, Mowen Lu, Emre Yamangil, and Russell Bent. Tightening McCormick relaxations for nonlinear programs via dynamic multivariate partitioning. In *International Conference on Principles and Practice of Constraint Programming*, pages 369–387. Springer, 2016. doi: 10.1007/978-3-319-44953-1_24.

[199] Harsha Nagarajan, Mowen Lu, Site Wang, Russell Bent, and Kaarthik Sundar. An adaptive, multivariate partitioning algorithm for global optimization of nonconvex programs. *Journal of Global Optimization*, 2019. ISSN 1573-2916. doi: 10.1007/s10898-018-00734-1.

[200] Harsha Nagarajan, Kaarthik Sundar, Hassan Hijazi, and Russell Bent. Convex hull formulations for mixed-integer multilinear functions. In *AIP Conference Proceedings*, volume 2070, page 020037. AIP Publishing LLC, 2019. doi: 10.1063/1.5090004.

[201] Yuko Nagata and Khim Hoong Chu. Optimization of a fermentation medium using neural networks and genetic algorithms. *Biotechnology Letters*, 25(21):1837–1842, 2003. doi: 10.1023/a:1026225526558.

[202] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. *ICML: Proceedings of the 27th International Conference on Machine Learning*, pages 807–814, 2010. doi: 10.5555/3104322.3104425.

[203] Jaromił Najman and Alexander Mitsos. Convergence analysis of multivariate McCormick relaxations. *Journal of Global Optimization*, pages 1–32, 2016. doi: 10.1007/s10898-016-0408-6.

[204] Jaromił Najman and Alexander Mitsos. Convergence order of McCormick relaxations of LMTD function in heat exchanger networks. *Computer Aided Chemical Engineering*, 38: 1605–1610, 2016. doi: 10.1016/B978-0-444-63428-3.50272-1.

[205] Jaromil Najman and Alexander Mitsos. On tightness and anchoring of McCormick and other relaxations. *Journal of Global Optimization*, Dec 2017. ISSN 1573-2916. doi: 10.1007/s10898-017-0598-6.

[206] Jaromił Najman and Alexander Mitsos. Tighter mccormick relaxations through subgradient propagation. *Journal of Global Optimization*, 75(3):565–593, 2019. doi: 10.1007/s10898-019-00791-0.

[207] Jaromił Najman, Dominik Bongartz, Angelos Tsoukalas, and Alexander Mitsos. Erratum to: Multivariate McCormick relaxations. *Journal of Global Optimization*, pages 1–7, 2016. doi: 10.1007/s10898-016-0470-0.

[208] Jaromił Najman, Dominik Bongartz, and Alexander Mitsos. Convex relaxations of componentwise convex functions. *Computers & Chemical Engineering*, 130:106527, 2019. doi: 10.1016/j.compchemeng.2019.106527.

[209] Jaromił Najman, Dominik Bongartz, and Alexander Mitsos. Relaxations of thermodynamic property and costing models in process engineering. *Computers & Chemical Engineering*, 130:106571, 2019. doi: 10.1016/j.compchemeng.2019.106571.

[210] Jaromił Najman, Dominik Bongartz, and Alexander Mitsos. Linearization of McCormick relaxations and hybridization with the auxiliary variable method. *Journal of Global Optimization*, 80(4):731–756, 2021. doi: 10.1007/s10898-020-00977-x.

[211] Cláudio Augusto Oller Nascimento and Reinaldo Giudici. Neural network based approach for optimisation applied to an industrial nylon-6,6 polymerisation process. *Computers & Chemical Engineering*, 22:S595–S600, 1998. doi: 10.1016/s0098-1354(98)00105-7.

[212] Nedialko S Nedialkov and Kenneth R Jackson. An interval Hermite-Obreschkoff method for computing rigorous bounds on the solution of an initial value problem for an ordinary differential equation. *Reliable Computing*, 5(3):289–310, 1999. doi: 10.1007/978-94-017-1247-7_23.

[213] Nedialko S Nedialkov and Kenneth R Jackson. A new perspective on the wrapping effect in interval methods for initial value problems for ordinary differential equations. In *Perspectives on Enclosure Methods*, pages 219–263. Springer, 2001.

[214] Nedialko S Nedialkov, Kenneth R Jackson, and George F Corliss. Validated solutions of initial value problems for ordinary differential equations. *Applied Mathematics and Computation*, 105(1):21–68, 1999. doi: 10.1016/S0096-3003(98)10083-8.

[215] Nedialko S Nedialkov, Kenneth R Jackson, and John 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(6):449–465, 2001. doi: 10.1023/A:1014798618404.

[216] Nedialko S Nedialkov, Vladik Kreinovich, and Scott A Starks. Interval arithmetic, affine arithmetic, Taylor series methods: why, what next? *Numerical Algorithms*, 37(1):325–336, 2004. doi: 10.1023/B:NUMA.0000049478.42605.cf.

[217] Dimitrios Nerantzis and Claire S Adjiman. Tighter \α bb relaxations through a refinement scheme for the scaled Gerschgorin theorem. *Journal of Global Optimization*, 73(3): 467–483, 2019. doi: 10.1007/s10898-018-0718-y.

[218] Arnold Neumaier. Rigorous sensitivity analysis for parameter-dependent systems of equations. *Journal of Mathematical Analysis and Applications*, 144(1):16–25, 1989. doi: 10.1016/0022-247X(89)90357-0.

[219] Arnold Neumaier. *Interval methods for systems of equations*, volume 37. Cambridge university press, 1990.

[220] Arnold Neumaier. Second-order sufficient optimality conditions for local and global nonlinear programming. *Journal of Global Optimization*, 9(2):141–151, 1996. doi: 10.1007/BF00121660.

[221] Arnold Neumaier, Oleg Shcherbina, Waltraud Huyer, and Tamás Vinkó. A comparison of complete global optimization solvers. *Mathematical programming*, 103(2):335–356, 2005. doi: 10.1007/s10107-005-0585-4.

[222] Jordan Ninin, Frédéric Messine, and Pierre Hansen. A reliable affine relaxation method for global optimization. *4OR*, 13(3):247–277, 2015. doi: 10.1007/s10288-014-0269-0.

[223] Carlos J Nohra, Arvind U Raghunathan, and Nikolaos Sahinidis. Spectral relaxations and branching strategies for global optimization of mixed-integer quadratic programs. *SIAM Journal on Optimization*, 31(1):142–171, 2021. doi: 10.1137/19M1271762.

[224] Carlos J Nohra, Arvind U Raghunathan, and Nikolaos V Sahinidis. SDP-quality bounds via convex quadratic relaxations for global optimization of mixed-integer quadratic programs. *Mathematical Programming*, pages 1–31, 2021. doi: 10.1007/s10107-021-01680-9.

[225] Chigozie Enyinna Nwankpa, Winifred Ijomah, Anthony Gachagan, and Stephen Marshall. Activation functions: comparison of trends in practice and research for deep learning. In *2nd International Conference on Computational Sciences and Technology*, pages 124–133, 2021.

[226] Richard Oberdieck, Nikolaos A. Diangelakis, Ioana Nascu, Maria M. Papathanasiou, Muxin Sun, Styliani Avraamidou, and Efstratios N. Pistikopoulos. On multi-parametric programming and its applications in process systems engineering. *Chemical Engineering Research and Design*, 116:61–82, 2016. doi: 10.1016/j.cherd.2016.09.034.

[227] Melis Onel, Chris A. Kieslich, and Efstratios N. Pistikopoulos. A nonlinear support vector machine-based feature selection approach for fault detection and diagnosis: Application to the tennessee eastman process. *AIChE Journal*, 65(3):992–1005, 2019. doi: 10.1002/aic.16497.

[228] Amir Hossein Pakseresht, Ehsan Ghasali, Mehrdad Nejati, Kamyar Shirvanimoghaddam, Amir Hossein Javadi, and Reza Teimouri. Development empirical-intelligent relationship between plasma spray parameters and coating performance of yttria-stabilized zirconia.

The International Journal of Advanced Manufacturing Technology, 76(5):1031–1045, 2015. doi: 10.1007/s00170-014-6212-x.

[229] Ioannis Papamichail and Claire S Adjiman. A rigorous global optimization algorithm for problems with ordinary differential equations. *Journal of Global Optimization*, 24(1):1–33, 2002. doi: 10.1023/A:1016259507911.

[230] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. dÁlché Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32 (NeurIPS 2019)*, pages 8024–8035, Vancouver, Canada, 2019. Curran Associates, Inc. URL `https://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf`.

[231] E. N. Pistikopoulos. Perspectives in multiparametric programming and explicit model predictive control. *AIChE journal*, 55(8):1918–1925, 2009. doi: 10.1002/aic.11965.

[232] Vicenç Puig, Joseba Quevedo, Teresa Escobet, Fatiha Nejjari, and Salvador de las Heras. Passive robust fault detection of dynamic processes using interval models. *IEEE Transactions on Control Systems Technology*, 16(5):1083–1089, 2008. doi: 10.1109/TCST.2007.906339.

[233] Luca Pulina and Armando Tacchella. Challenging SMT solvers to verify neural networks. *AI Communications*, 25(2):117–135, 2012. doi: 10.5555/2350156.2350160.

[234] Yash Puranik and N. V Sahinidis. Domain reduction techniques for global NLP and MINLP optimization. *Constraints*, 22(3):338–376, 2017. doi: 10.1007/s10601-016-9267-5.

[235] Jennifer Puschke, Hatim Djelassi, Johanna Kleinekorte, Ralf Hannemann-Tamás, and Alexander Mitsos. Robust dynamic optimization of batch processes under parametric uncertainty: Utilizing approaches from semi-infinite programs. *Computers & Chemical Engineering*, 116:253–267, 2018. doi: 10.1016/j.compchemeng.2018.05.025.

[236] Marco Quaglio, Eric S. Fraga, Enhong Cao, Asterios Gavriilidis, and Federico Galvanin. A model-based data mining approach for determining the domain of validity of approximated models. *Chemometrics and Intelligent Laboratory Systems*, 172:58–67, 2018. doi: 10.1016/j.chemolab.2017.11.010.

[237] Ignacio Quesada and Ignacio E Grossmann. A global optimization algorithm for linear fractional and bilinear programs. *Journal of Global Optimization*, 6(1):39–76, 1995. doi: 10.1007/BF01106605.

[238] Chris Rackauckas, Mike Innes, Yingbo Ma, Jesse Bettencourt, Lyndon White, and Vaibhav Dixit. Diffeqflux.jl - A Julia library for neural differential equations. *arXiv preprint arXiv:1902.02376*, 2019.

[239] Christopher Rackauckas and Qing Nie. DifferentialEquations.jl–A performant and feature-rich ecosystem for solving differential equations in Julia. *Journal of Open Research Software*, 5(1), 2017. doi: 10.5334/jors.151.

[240] S Rajakumar and V Balasubramanian. Diffusion bonding of titanium and aa 7075 aluminum alloy dissimilar jointsprocess modeling and optimization using desirability approach. *The International Journal of Advanced Manufacturing Technology*, 86(1): 1095–1112, 2016. doi: 10.1007/s00170-015-8223-7.

[241] P Rajesh, U Nagaraju, G Harinath Gowd, and T Vishnu Vardhan. Experimental and parametric studies of nd: Yag laser drilling on austenitic stainless steel. *The International Journal of Advanced Manufacturing Technology*, 93(1):65–71, 2017. doi: 10.1007/s00170-015-7639-4.

[242] T. K. Ralphs and L. Ladanyi. SYMPHONY: A Parallel Framework for Branch and Cut, 1999.

[243] Prajit Ramachandran, Barret Zoph, and Quoc V Le. Searching for activation functions. *arXiv preprint*, 2017.

[244] Jarrett Revels. Cassette.jl. `https://github.com/jrevels/Cassette.jl`, 2018.

[245] Jarrett Revels, Miles Lubin, and Theodore Papamarkou. Forward-mode automatic differentiation in Julia. *arXiv preprint arXiv:1607.07892*, 2016.

[246] Robert Rihm. Interval methods for initial value problems in ODEs. *Topics in validated computations*, pages 173–207, 1994.

[247] Robert Rihm. Implicit methods for enclosing solutions of ODEs. *Journal of universal computer science*, 4(2):202–209, 1998.

[248] Anatoliy D Rikun. A convex envelope formula for multilinear functions. *Journal of Global Optimization*, 10(4):425–437, 1997. doi: 10.1023/A:1008217604285.

[249] Diego Rosso, Lory E Larson, and Michael K Stenstrom. Aeration of large-scale municipal wastewater treatment plants: state of the art. *Water Science and Technology*, 57(7): 973–978, 2008. doi: 10.2166/wst.2008.218.

[250] Lars Ruthotto and Eldad Haber. Deep neural networks motivated by partial differential equations. *Journal of Mathematical Imaging and Vision*, 62(3):352–364, 2020. doi: 10.1007/s10851-019-00903-1.

[251] Hong S Ryoo and Nikolaos V Sahinidis. Global optimization of nonconvex NLPs and MINLPs with applications in process design. *Computers & Chemical Engineering*, 19(5): 551–566, 1995. doi: 10.1016/0098-1354(94)00097-2.

[252] Hong S. Ryoo and Nikolaos V. Sahinidis. A branch-and-reduce approach to global optimization. *Journal of Global Optimization*, 8(2):107–138, 1996. ISSN 1573-2916. doi: 10.1007/BF00138689.

[253] N. V. Sahinidis. *BARON 21.1.13: Global Optimization of Mixed-Integer Nonlinear Programs,* User's Manual, 2017. URL `https://www.minlp.com/downloads/docs/baron%20manual.pdf`.

[254] Nikolaos V. Sahinidis. Baron: A general purpose global optimization software package. *Journal of Global Optimization*, 8(2):201–205, 1996. ISSN 1573-2916. doi: 10.1007/BF00138693.

[255] A. M. Sahlodin and B. Chachuat. Convex/concave relaxations of parametric ODEs using Taylor models. *Computers and Chemical Engineering*, 35:844–857, 2011. ISSN 0098-1354. doi: 10.1016/j.compchemeng.2011.01.031.

[256] Ali M Sahlodin and Benoit Chachuat. Discretize-then-relax approach for convex/concave relaxations of the solutions of parametric odes. *Applied Numerical Mathematics*, 61(7): 803–820, 2011. doi: 10.1016/j.apnum.2011.01.009.

[257] Ali M. Sahlodin and Benot Chachuat. Discretize-then-relax approach for state relaxations in global dynamic optimization. *Computer Aided Chemical Engineering*, pages 427–432, 2010. ISSN 1570-7946. doi: 10.1016/s1570-7946(10)28072-0.

[258] Ali M. Sahlodin and Benot Chachuat. Tight Convex and Concave Relaxations via Taylor Models for Global Dynamic Optimization. *Computer Aided Chemical Engineering*, 29: 537–541, 2011. ISSN 1570-7946. doi: 10.1016/b978-0-444-53711-9.50108-5.

[259] Ali Mohammad Sahlodin. *Global optimization of dynamic process systems using complete search methods*. PhD thesis, McMaster University, 2013.

[260] David P. Sanders, Luis Benet, lucaferranti, Krish Agarwal, Benot Richard, Josua Grawitter, Eeshan Gupta, Michael F. Herbst, Marcelo Forets, yashrajgupta, Eric Hanson, Braam van Dyk, Christopher Rackauckas, Rushabh Vasani, Sebastian Miclua-C¢mpeanu, Sheehan Olver, Twan Koolen, Caroline Wormell, Favio Andr Vzquez, Julia TagBot, Kevin O'Bryant, Kristoffer Carlsson, Morten Piibeleht, Reno, Robin Deits, Tim Holy, Marek Kaluba, and matsueushi. Juliaintervals/intervalarithmetic.jl: v0.18.2, May 2021.

[261] Spencer D Schaber, Joseph K Scott, and Paul I Barton. Convergence-order analysis for differential-inequalities-based bounds and relaxations of the solutions of odes. *Journal of Global Optimization*, 73(1):113–151, 2019. doi: 10.1007/s10898-018-0691-5.

[262] Hermann Schichl and Arnold Neumaier. Interval analysis on directed acyclic graphs for global optimization. *Journal of Global Optimization*, 33(4):541–562, 12 2005. doi: 10.1007/s10898-005-0937-x. Copyright - Springer 2005; Last updated - 2014-08-23.

[263] Artur M Schweidtmann. *Global optimization of processes through machine learning*. PhD thesis, Dissertation, Rheinisch-Westfälische Technische Hochschule Aachen, 2021, 2018.

[264] Artur M. Schweidtmann and Alexander Mitsos. Deterministic global optimization with artificial neural networks embedded. *Journal of Optimization Theory and Applications*, 180(3):925–948, Mar 2019. ISSN 1573-2878. doi: 10.1007/s10957-018-1396-0.

[265] Artur M. Schweidtmann, Dominik Bongartz, Wolfgang R. Huster, and Alexander Mitsos. Deterministic global process optimization: Flash calculations via artificial neural networks. In *Computer Aided Chemical Engineering*, volume 46, pages 937–942. Elsevier, Amsterdam, Netherlands, 2019. doi: 10.1016/b978-0-12-818634-3.50157-0.

[266] Artur M. Schweidtmann, Wolfgang R. Huster, Jannik T. Lthje, and Alexander Mitsos. Deterministic global process optimization: Accurate (single-species) properties via artificial neural networks. *Computers & Chemical Engineering*, 121:67–74, 2019. doi: 10.1016/j.compchemeng.2018.10.007.

[267] Artur M Schweidtmann, Dominik Bongartz, Daniel Grothe, Tim Kerkenhoff, Xiaopeng Lin, Jaromił Najman, and Alexander Mitsos. Deterministic global optimization with gaussian processes embedded. *Mathematical Programming Computation*, 13(3):553–581, 2021. doi: 10.1007/s12532-021-00204-y.

[268] Artur M Schweidtmann, Jana M Weber, Christian Wende, Linus Netze, and Alexander Mitsos. Obey validity limits of data-driven models through topological data analysis and one-class classification. *Optimization and Engineering*, pages 1–22, 2021. doi: https://doi.org/10.1007/s11081-021-09608-0.

[269] Joseph K. Scott and Paul I. Barton. Convex relaxations for nonconvex optimal control problems. In *50$^{\text{th}}$ IEEE Conference on Decision and Control*, pages 1042–1047, 2011. doi: 10.1109/cdc.2011.6160284.

[270] Joseph K. Scott and Paul I. Barton. Improved relaxations for the parametric solutions of ODEs using differential inequalities. *Journal of Global Optimization*, 57(1):143–176, Sep 2013. ISSN 1573-2916. doi: 10.1007/s10898-012-9909-0.

[271] Joseph K Scott, Matthew D Stuber, and Paul I Barton. Generalized McCormick relaxations. *Journal of Global Optimization*, 51(4):569–606, 2011. doi: 10.1007/s10898-011-9664-7.

[272] Joseph K Scott, Benoit Chachuat, and Paul I Barton. Nonlinear convex and concave relaxations for the solutions of parametric ODEs. *Optimal Control Applications and Methods*, 34(2):145–163, 2013. doi: 10.1002/oca.2014.

[273] Yongho Seong, Changhyup Park, Jinho Choi, and Ilsik Jang. Surrogate model with a deep neural network to evaluate gas–liquid flow in a horizontal pipe. *Energies*, 13(4):968, 2020. doi: 10.3390/en13040968.

[274] Radu Serban and Alan C Hindmarsh. CVODES: the sensitivity-enabled ODE solver in SUNDIALS. In *ASME 2005 international design engineering technical conferences and computers and information in engineering conference*, pages 257–269. American Society of Mechanical Engineers, 2005. doi: 10.1115/DETC2005-85597.

[275] Yuanxun Shao and Joseph K Scott. Convex relaxations for global optimization under uncertainty described by continuous random variables. *AIChE Journal*, 64(8):3023–3033, 2018. doi: 10.1002/aic.16064.

[276] Kai Shen and Joseph K. Scott. Rapid and accurate reachability analysis for nonlinear dynamic systems by exploiting model redundancy. *Computers and Chemical Engineering*, 106:596–608, 2017. ISSN 0098-1354. doi: 10.1016/j.compchemeng.2017.08.001.

[277] Kai Shen and Joseph K Scott. Rapid and accurate reachability analysis for nonlinear dynamic systems by exploiting model redundancy. *Computers and Chemical Engineering*, 106:596–608, 2017.

[278] Kai Shen and Joseph K. Scott. Mean value form enclosures for nonlinear reachability analysis. In *2018 IEEE Conference on Decision and Control*, pages 7112–7117, 2018. doi: 10.1109/cdc.2018.8619809.

[279] Xinyue Shen, Steven Diamond, Yuantao Gu, and Stephen Boyd. Disciplined convex-concave programming. In *Decision and Control (CDC), 2016 IEEE 55th Conference on*, pages 1009–1014. IEEE, 2016. doi: 10.1109/CDC.2016.7798400.

[280] Hanif D Sherali. Convex envelopes of multilinear functions over a unit hypercube and over special discrete sets. *Acta mathematica vietnamica*, 22(1):245–270, 1997.

[281] Hanif D Sherali and Warren P Adams. *A reformulation-linearization technique for solving discrete and continuous nonconvex problems*, volume 31. Springer Science & Business Media, 2013. doi: 10.1007/978-1-4757-4388-3.

[282] A. B. Singer and P. I. Barton. Global solution of optimization problems with parameter-embedded linear dynamic systems. *Journal of Optimization Theory and Applications*, 121:613–646, 2004. ISSN 0022-3239. doi: 10.1023/b:jota.0000037606.79050.a7.

[283] Adam B Singer. *Global dynamic optimization*. PhD thesis, Massachusetts Institute of Technology, 2004.

[284] Adam B Singer and Paul I Barton. Global optimization with nonlinear ordinary differential equations. *Journal of Global Optimization*, 34(2):159–190, 2006. doi: 10.1007/s10898-005-7074-4.

[285] Adam B Singer and Paul I Barton. Bounding the solutions of parameter dependent nonlinear ordinary differential equations. *SIAM Journal on Scientific Computing*, 27(6): 2167–2182, 2006.

[286] Adam B Singer, Benoıt Chachuat, and Paul I Barton. GDOC Version 1.0 manual, 2005.

[287] Adam B Singer, James W Taylor, Paul I Barton, and William H Green. Global dynamic optimization for parameter estimation in chemical kinetics. *The Journal of Physical Chemistry A*, 110(3):971–976, 2006. doi: 10.1021/jp0548873.

[288] Anders Skjäl and Tapio Westerlund. New methods for calculating *alpha* bb-type underestimators. *Journal of Global Optimization*, 58(3):411–427, 2014. doi: 10.1007/s10898-013-0057-y.

[289] Anders Skjäl, Tapio Westerlund, Ruth Misener, and Christodoulos A Floudas. A generalization of the classical αbb convex underestimation via diagonal and nondiagonal quadratic terms. *Journal of Optimization Theory and Applications*, 154(2):462–490, 2012. doi: 10.1007/s10957-012-0033-6.

[290] Edward M.B. Smith and Constantinos C. Pantelides. Global optimisation of nonconvex MINLPs. *Computers & Chemical Engineering*, 21:S791 – S796, 1997. ISSN 0098-1354. doi: http://dx.doi.org/10.1016/S0098-1354(97)87599-0.

[291] Edward MB Smith and Constantinos C Pantelides. Global optimisation of nonconvex MINLPs. *Computers & Chemical Engineering*, 21:S791–S796, 1997. doi: 10.1016/0098-1354(94)00097-2.

[292] E.M.B. Smith and C.C. Pantelides. A symbolic reformulation/spatial branch-and-bound algorithm for the global optimisation of nonconvex {MINLPs}. *Computers & Chemical Engineering*, 23(45):457 – 478, 1999. ISSN 0098-1354. doi: https://doi.org/10.1016/S0098-1354(98)00286-5.

[293] Yingkai Song and Kamil A. Khan. Optimization-based convex relaxations for nonconvex parametric systems of ordinary differential equations. *Mathematical Programming*, 2021. doi: https://doi.org/10.1007/s10107-021-01654-x.

[294] Oliver Stein and Paul Steuermann. The adaptive convexification algorithm for semi-infinite programming with arbitrary index sets. *Mathematical programming*, 136(1):183–207, 2012. doi: 10.1007/s10107-012-0556-5.

[295] J. Stolfi and L. H. De Figueiredo. An introduction to affine arithmetic. *TEMA - Tendências em Matemática Aplicada e Computacional*, 4(3):297–312, dec 2003. ISSN 2179-8451. doi: 10.5540/tema.2003.04.03.0297.

[296] M. D. Stuber. *Evaluation of Process Systems Operating Envelopes*. PhD dissertation, Massachusetts Institute of Technology, February 2013.

[297] Matthew D. Stuber. A differentiable model for optimizing hybridization of industrial process heat systems with concentrating solar thermal power. *Processes*, 6(7):76, 2018. doi: 10.3390/pr6070076.

[298] Matthew D. Stuber and Paul I. Barton. Semi-infinite optimization with implicit functions. *Industrial & Engineering Chemistry Research*, 54(1):307–317, 2015. doi: 10.1021/ie5029123.

[299] Matthew D. Stuber, Achim Wechsung, Arul Sundaramoorthy, and Paul I. Barton. Worst-case design of subsea production facilities using semi-infinite programming. *AIChE Journal*, 60(7):2513–2524, 2014. ISSN 1547-5905. doi: 10.1002/aic.14447.

[300] Matthew D Stuber, Joseph K Scott, and Paul I Barton. Convex and concave relaxations of implicit functions. *Optimization Methods and Software*, 30(3):424–460, 2015. doi: 10.1080/10556788.2014.924514.

[301] Kaarthik Sundar, Harsha Nagarajan, Jeff Linderoth, Site Wang, and Russell Bent. Piecewise polyhedral formulations for a multilinear term. *Operations Research Letters*, 49 (1):144–149, 2021. doi: 10.1016/j.orl.2020.12.002.

[302] Kaarthik Sundar, Sujeevraja Sanjeevi, and Harsha Nagarajan. Sequence of polyhedral relaxations for nonlinear univariate functions. *Optimization and Engineering*, pages 1–18, 2021. doi: 10.1007/s11081-021-09609-z.

[303] 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. Springer US, 2002. ISBN 9781402010316.

[304] Mohit Tawarmalani and Nikolaos V. Sahinidis. Global optimization of mixed-integer nonlinear programs: A theoretical and computational study. *Mathematical Programming*, 99(3):563–591, 2004. ISSN 1436-4646. doi: 10.1007/s10107-003-0467-6.

[305] Mohit Tawarmalani and Nikolaos V Sahinidis. A polyhedral branch-and-cut approach to global optimization. *Mathematical Programming*, 103(2):225–249, 2005. ISSN 1436-4646. doi: 10.1007/s10107-005-0581-8.

[306] James W Taylor, Gerhard Ehlker, Hans-Heinrich Carstensen, Leah Ruslen, Robert W Field, and William H Green. Direct measurement of the fast, reversible addition of oxygen to cyclohexadienyl radicals in nonpolar solvents. *The Journal of Physical Chemistry A*, 108(35):7193–7203, 2004.

[307] Jefferson W Tester, Michael Modell, et al. *Thermodynamics and its Applications*. Prentice Hall PTR, 1997.

[308] TH Tsang, DM Himmelblau, and TF Edgar. Optimal control via collocation and non-linear programming. *International Journal of Control*, 21(5):763–768, 1975. doi: 10.1080/00207177508922030.

[309] Calvin Tsay, Jan Kronqvist, Alexander Thebelt, and Ruth Misener. Partition-based formulations for mixed-integer optimization of trained ReLU neural networks. *Advances in Neural Information Processing Systems*, 34, 2021.

[310] Angelos Tsoukalas and Alexander Mitsos. Multivariate McCormick relaxations. *Journal of Global Optimization*, 59(2-3):633–662, 2014. doi: 10.1007/s10898-014-0176-0.

[311] Luis N Vicente and Paul H Calamai. Bilevel and multilevel programming: A bibliography review. *Journal of Global optimization*, 5(3):291–306, oct 1994. ISSN 0925-5001. doi: 10.1007/bf01096458.

[312] Stefan Vigerske. *Decomposition in multistage stochastic programming and a constraint integer programming approach to mixed-integer nonlinear programming*. PhD thesis, Humboldt-Universität zu Berlin, Mathematisch-Naturwissenschaftliche Fakultät II, 2013.

[313] Stefan Vigerske and Ambros Gleixner. SCIP: global optimization of mixed-integer nonlinear programs in a branch-and-cut framework. *Optimization Methods and Software*, 33(3):563–593, 2018. doi: 10.1080/10556788.2017.1335312.

[314] Mario Villanueva, Radoslav Paulen, Boris Houska, and Benoît Chachuat. Enclosing the reachable set of parametric ODEs using taylor models and ellipsoidal calculus. In *Computer Aided Chemical Engineering*, volume 32, pages 979–984. Elsevier, 2013. doi: 10.1016/B978-0-444-63234-0.50164-0.

[315] Mario E Villanueva, Boris Houska, and Benoît Chachuat. Unified framework for the propagation of continuous-time enclosures for parametric nonlinear ODEs. *Journal of Global Optimization*, 62(3):575–613, 2015. doi: 10.1007/s10898-014-0235-6.

[316] Thomas Villmann, Jensun Ravichandran, Andrea Villmann, David Nebel, and Marika Kaden. Investigation of activation functions for generalized learning vector quantization. *International Workshop on Self-Organizing Maps*, pages 179–188, 2019. doi: 10.1007/978-3-030-19642-4_18.

[317] Andreas Wächter and Lorenz T. Biegler. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical Programming*, 106(1):25–57, Mar 2006. ISSN 1436-4646. doi: 10.1007/s10107-004-0559-y.

[318] Olga Walz, Hatim Djelassi, and Alexander Mitsos. Optimal experimental design for optimal process design: A trilevel optimization formulation. *AIChE Journal*, 66(1): e16788, 2020. doi: 10.1002/aic.16788.

[319] Chenyu Wang, Samuel Degnan-Morgenstern, John D. Martin, and Matthew D. Stuber. Optimal therapy design with tumor microenvironment normalization. *AIChE Journal*, page e17747, may 2022. doi: 10.1002/aic.17747.

[320] Endong Wang, Qing Zhang, Bo Shen, Guangyong Zhang, Xiaowei Lu, Qing Wu, and Yajuan Wang. Intel math kernel library. In *High-performance computing on the Intel Xeon Phi*, pages 167–188. Springer, 2014. doi: 10.1007/978-3-319-06486-4_7.

[321] Xueyi Wang, Guiping Du, Mingtao Wu, Qingliang Song, and Yu Chen. Two-step robust design of LLC converter with corner judgement and greedy algorithm. *IEEE Transactions on Power Electronics*, pages 1–1, 2021. doi: 10.1109/tpel.2021.3139049.

[322] Achim Wechsung. *Global optimization in reduced space*. PhD thesis, Massachusetts Institute of Technology, 2014.

[323] Achim Wechsung, Spencer D. Schaber, and Paul I. Barton. The cluster problem revisited. *Journal of Global Optimization*, 58(3):429–438, Mar 2014. ISSN 1573-2916. doi: 10.1007/s10898-013-0059-9.

[324] Achim Wechsung, Joseph K Scott, Harry AJ Watson, and Paul I Barton. Reverse propagation of McCormick relaxations. *Journal of Global Optimization*, 63(1):1–36, 2015. doi: 10.1007/s10898-015-0303-6.

[325] Ue-Pyng Wen and Shuh-Tzy Hsu. Linear bi-level programming problems—a review. *Journal of the Operational Research Society*, 42(2):125–133, 1991. doi: 10.1038/sj/jors/0420204.

[326] Matthew Wilhelm and Matthew D Stuber. Easy advanced global optimization (eago): An open-source platform for robust and global optimization in julia, 2017. AIChE Annual Meeting.

[327] Matthew E. Wilhelm and Matthew D. Stuber. EAGO.jl: easy advanced global optimization in Julia. *Optimization Methods and Software*, pages 1–26, 2020. doi: 10.1080/10556788.2020.1786566.

[328] Matthew E Wilhelm, Anne V Le, and Matthew D Stuber. Global Optimization of Stiff Dynamical Systems. *AIChE Journal*, 2019. doi: 10.1002/aic.16836.

[329] Matthew E. Wilhelm, Robert X. Gottlieb, and Matthew D. Stuber. PSORLab/McCormick.jl, 2020. URL `https://github.com/PSORLab/McCormick.jl`.

[330] Matthew E. Wilhelm, Chenyu Wang, and Matthew D. Stuber. Robust optimization with hybrid first-principles data-driven models, 2021.

[331] Matthew E. Wilhelm, Chenyu Wang, and Matthew D. Stuber. Convex and concave envelopes of artificial neural network activation functions for deterministic global optimization, 2022. under review.

[332] Christopher Williams and Carl Rasmussen. Gaussian processes for regression. *Advances in Neural Information Processing systems*, 8:514–520, 1995. doi: 10.5555/2998828.2998901.

[333] Lili Xu and CL Philip Chen. Comparison and combination of activation functions in broad learning system. In *2020 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 3537–3542. IEEE, 2020. doi: 10.1109/SMC42975.2020.9282871.

[334] Yasutoshi Yajima. Convex envelopes in optimization problems: Convex envelopes in optimization problems. In Christodoulos A. Floudas and Panos M. Pardalos, editors, *Encyclopedia of Optimization*, pages 343–344. Springer US, Boston, MA, 2001. ISBN 978-0-306-48332-5. doi: 10.1007/0-306-48332-7_74.

[335] Xuejiao Yang and Joseph K. Scott. Efficient Reachability Bounds for Discrete-Time Nonlinear Systems by Extending the Continuous-Time Theory of Differential Inequalities. In *2018 Annual American Control Conference*, pages 6242–6247. IEEE, jun 2018. doi: 10.23919/acc.2018.8431811.

[336] Xuejiao Yang and Joseph K Scott. Efficient reachability bounds for discrete-time nonlinear systems by extending the continuous-time theory of differential inequalities. In *2018 Annual American Control Conference (ACC)*, pages 6242–6247. IEEE, 2018. doi: 10.23919/ACC.2018.8431811.

[337] Xuejiao Yang and Joseph K Scott. Accurate uncertainty propagation for discrete-time nonlinear systems using differential inequalities with model redundancy. *IEEE Transactions on Automatic Control*, 65(12):5043–5057, 2020. doi: 10.1109/TAC.2020.2968241.

[338] Zhongbo Yue, Chuanzhen Huang, Hongtao Zhu, Jun Wang, Peng Yao, and ZengWen Liu. Optimization of machining parameters in the abrasive waterjet turning of alumina ceramic based on the response surface methodology. *The International Journal of Advanced Manufacturing Technology*, 71(9-12):2107–2114, 2014. doi: 10.1007/s00170-014-5624-y.

[339] Hao Zheng, Zhanlei Yang, Wenju Liu, Jizhong Liang, and Yanpeng Li. Improving deep neural networks using softplus units. In *2015 International Joint Conference on Neural Networks (IJCNN)*, pages 1–4. IEEE, 2015. doi: 10.1109/IJCNN.2015.7280459.

[340] Keith Zorn and Nikolaos V Sahinidis. Global optimization of general non-convex problems with intermediate bilinear substructures. *Optimization Methods and Software*, 29 (3):442–462, 2014. doi: 10.1080/10556788.2013.783032.