ORIGINAL ARTICLE

Process Systems Engineering

Global Optimization of Stiff Dynamical Systems[1]

Matthew E. Wilhelm[1] | Anne V. Le[2] | Matthew D. Stuber[1]

[1]Process Systems and Operations Research Laboratory, Dept. of Chemical and Biomolecular Engineering, University of Connecticut, Storrs, CT, 06269, USA

[2]Dept. of Chemical Engineering, Texas A&M University, College Station, TX, 77843, USA, Current address: Dept. of Chemical Engineering, University of Massachusetts Amherst, Amherst, MA, 01003, USA

Correspondence
Matthew D. Stuber, PhD, University of Connecticut, Storrs, CT, 06269, USA
Email: stuber@alum.mit.edu

Funding information
National Science Foundation, Award No.: 1560072, 1706343, 1932723; University of Connecticut

We present a deterministic global optimization method 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.

KEYWORDS
dynamic simulation, stiff systems, global optimization, implicit functions

[1]Author's final accepted version. Published version: Wilhelm, M.E., Le, A.V., and M.D. Stuber. Global Optimization of Stiff Dynamical Systems. AIChE Journal. 65(12):e16836 (2019). DOI: 10.1002/aic.16836

Introduction

Dynamic optimization problems of the form:

$$\begin{aligned} \phi^* = & \min_{\mathbf{p}\in P\subset\mathbb{R}^{n_p}} \phi(\mathbf{x}(\mathbf{p},t_f),\mathbf{p}) \\ \text{s.t. } & \dot{\mathbf{x}}(\mathbf{p},t) = \mathbf{f}(\mathbf{x}(\mathbf{p},t),\mathbf{p},t), \quad \forall t\in I=[t_0,t_f] \\ & \mathbf{x}(\mathbf{p},t_0) = \mathbf{x}_0(\mathbf{p}) \\ & \mathbf{g}(\mathbf{x}(\mathbf{p},t_f),\mathbf{p}) \le \mathbf{0} \end{aligned} \tag{1}$$

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 (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 (1) is a nonconvex program, in general, and therefore verifying optimality requires deterministic global optimization. The focus of this paper is on solving (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 (1) rigorously to global optimality rely on the spatial branch-and-bound (B&B) framework [1,2] 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 (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 (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 (1) to global optimality were introduced by Papamichail and Adjiman [3] which utilized the αBB convex relaxations [4,5], and by Singer and Barton [6,7] utilizing McCormick-based relaxations [8]. 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 [9,10,11,12,13,14,15] 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 [16]) or state-of-the-art software packages (e.g., CVODES [17]) [7,10,13]. 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 [18,19,20,21]. 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 [22]. More recently, a discrete-time differential inequalities approach was developed utilizing the explicit Euler scheme [23]. 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 [24] 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 [7] and Mitsos et al. [22] to global optimality using the implicit (backward) Euler integration scheme [24].

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. [24] to more accurate higher-order numerically-stable implicit integration schemes. The proposed approach reformulates the dynamic optimization problem (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 (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 [25], ANTIGONE [26]) 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. [24] 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 [24] 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 paper 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.

Background

Interval Arithmetic

Throughout this article, $A = [\mathbf{a}^L, \mathbf{a}^U]$ will represent an n-dimensional interval that is a nonempty compact set defined as $A = \{\mathbf{a} \in \mathbb{R}^n : \mathbf{a}^L \leq \mathbf{a} \leq \mathbf{a}^U\}$ with $\mathbf{a}^L$ and $\mathbf{a}^U$ the lower and upper bounds of the interval, respectively. A set A^n is defined as the Cartesian product $A^n = A \times A \times \cdots \times A$. Further, let $\mathbb{IR}^n = \{[\mathbf{a}^L, \mathbf{a}^U] \subset \mathbb{R}^n\}$ be the set of all n-dimensional real intervals and for any $D \subset \mathbb{R}^n$, $\mathbb{I}D = \{A \in \mathbb{IR}^n : A \subset D\}$ is the set of all interval subsets of D. The interior of an n-dimensional interval is denoted $\text{int}(A)$ and is defined as the open set containing all points which do not strictly lie on the bounds of A. The image of A under the mapping $\mathbf{f} : D \to \mathbb{R}^n$ will be denoted $\mathbf{f}(A)$ whereas an inclusion monotonic interval extension of $\mathbf{f}$ on A will be denoted $F(A)$. From the Fundamental Theorem of Interval Analysis (Thm. 1.4.1 in Neumaier [27]), we have $\mathbf{f}(A) \subset F(A),\ \forall A \in \mathbb{I}D$.

Parametric Ordinary Differential Equations

Consider a system of parametric ODE-IVPs:

$$\begin{aligned} \dot{\mathbf{x}}(\mathbf{p}, t) = \frac{d\mathbf{x}}{dt}(\mathbf{p}, t) &= \mathbf{f}(\mathbf{x}(\mathbf{p}, t), \mathbf{p}, t), \quad t \in I = [t_0, t_f],\ \mathbf{p} \in P \\ \mathbf{x}(\mathbf{p}, t_0) &= \mathbf{x}_0(\mathbf{p}), \quad \mathbf{p} \in P \end{aligned} \tag{2}$$

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 $P \in \mathbb{I}\Pi$ and $I \in \mathbb{I}T$.

Assumption 1 *The parametric ODE-IVP system* (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) 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) on I. Further, it is assumed that a unique solution exists over the time domain I for every $\mathbf{p} \in P$.

Deterministic Global Optimization

Nearly all deterministic global optimization routines rely on a variation of the B&B algorithm [28]. During the course of this routine, the decision space is iteratively partitioned into subdomains which are fathomed based on computed lower and upper bounds as well as subproblem feasibility. The lower bounds are computed by solving relaxed (underestimating) optimization subproblems [2]. These relaxed problems can be constructed from the relaxations of the objective and constraint functions. We follow this approach herein motivating the discussion of relaxations below.

Definition 1 (Convex and Concave Relaxations [22]) *Given a convex set* $Z \subset \mathbb{R}^n$ *and a function* $f : Z \to \mathbb{R}$, *a convex function* $f^{cv} : Z \to \mathbb{R}$ *is a* convex relaxation *of* f *on* Z *if* $f^{cv}(\mathbf{z}) \leq f(\mathbf{z})$ *for every* $\mathbf{z} \in Z$. *A concave function* $f^{cc} : Z \to \mathbb{R}$ *is a* concave relaxation *of* f *on* Z *if* $f^{cc}(\mathbf{z}) \geq f(\mathbf{z})$ *for every* $\mathbf{z} \in Z$.

Note that this definition involves scalar functions. However, convex and concave relaxations of vector valued functions $\mathbf{f} : Z \to \mathbb{R}^n$ are defined by applying the above inequalities componentwise [24].

Definition 2 (McCormick Relaxation [22]) *Relaxations of factorable functions that are formed from the recursive application of univariate composition, binary multiplication, and binary addition from convex and concave relaxations of univariate intrinsic functions, without the introduction of auxiliary variables, are referred to as* McCormick relaxations.

Definition 3 (Subgradients [24]) *Let $Z \subset R^n$ be a nonempty convex set, $f^{cv} : Z \to \mathbb{R}$ be convex and $f^{cc} : Z \to \mathbb{R}$ be concave. A function $\mathbf{s}_f^{cv} : Z \to \mathbb{R}^n$ is a subgradient of f^{cv} on Z if for each $\bar{\mathbf{z}} \in Z, f^{cv}(\mathbf{z}) \leq f^{cv}(\bar{\mathbf{z}}) + \mathbf{s}_f^{cv}(\bar{\mathbf{z}})^T(\mathbf{z}-\bar{\mathbf{z}}), \forall \mathbf{z} \in Z$. Similarly, a function $\mathbf{s}_f^{cc} : Z \to \mathbb{R}^n$ is a subgradient of f^{cc} on Z if for each $\bar{\mathbf{z}} \in Z, f^{cc}(\mathbf{z}) \geq f^{cc}(\bar{\mathbf{z}}) + \mathbf{s}_f^{cc}(\bar{\mathbf{z}})^T(\mathbf{z}-\bar{\mathbf{z}}), \forall \mathbf{z} \in Z$.*

Note that subgradients of vector-valued functions and subgradients of matrix-valued functions will be defined analogously and will be matrix-valued functions and 3rd-order tensor-valued functions, respectively.

Definition 4 (Composite Relaxations [24]: $\mathbf{u}_{\mathcal{G}}, \mathbf{o}_{\mathcal{G}}$) *Let $D \subset \mathbb{R}^n$, $Z \in \mathbb{I}D$, and $P \in \mathbb{IR}^{n_p}$. Define the mapping $\mathcal{G} : D \times P \to \mathbb{R}^n$. The functions $\mathbf{u}_{\mathcal{G}}, \mathbf{o}_{\mathcal{G}} : \mathbb{R}^n \times \mathbb{R}^n \times P \to \mathbb{R}^n$ are called* composite relaxations *of $\mathcal{G}$ on $Z \times P$ if for any $\psi^{cv}, \psi^{cc} : P \to \mathbb{R}^n$, the functions $\mathbf{u}_{\mathcal{G}}(\psi^{cv}(\cdot), \psi^{cc}(\cdot), \cdot)$ and $\mathbf{o}_{\mathcal{G}}(\psi^{cv}(\cdot), \psi^{cc}(\cdot), \cdot)$ are, respectively, convex and concave relaxations of $\mathcal{G}(\mathbf{q}(\cdot), \cdot)$ on P for any function $\mathbf{q} : P \to Z$ and any pair of convex and concave relaxations (ψ^{cv}, ψ^{cc}) of $\mathbf{q}$ on P.*

Remark In this paper, composite relaxations are computed using the generalized McCormick relaxation framework [29].

Definition 5 ($\bar{\mathbf{u}}_{\mathcal{G}}, \bar{\mathbf{o}}_{\mathcal{G}}$ [24]) *Let $\mathbf{u}_{\mathcal{G}}, \mathbf{o}_{\mathcal{G}}$ be composite relaxations of $\mathcal{G} : D \times P \to \mathbb{R}^n$ on $Z \times P$. The functions $\bar{\mathbf{u}}_{\mathcal{G}}, \bar{\mathbf{o}}_{\mathcal{G}} : \mathbb{R}^n \times \mathbb{R}^n \times P \to \mathbb{R}^n$ will be defined as*

$$\bar{\mathbf{u}}_{\mathcal{G}}(\psi^{cv}, \psi^{cc}, \mathbf{p}) \equiv \max\{\psi^{cv}, \mathbf{u}_{\mathcal{G}}(\psi^{cv}, \psi^{cc}, \mathbf{p})\},$$
$$\bar{\mathbf{o}}_{\mathcal{G}}(\psi^{cv}, \psi^{cc}, \mathbf{p}) \equiv \min\{\psi^{cc}, \mathbf{o}_{\mathcal{G}}(\psi^{cv}, \psi^{cc}, \mathbf{p})\},$$

$\forall (\psi^{cv}, \psi^{cc}, \mathbf{p}) \in \mathbb{R}^n \times \mathbb{R}^n \times P$ with the max/min *operations applied componentwise.*

Definition 6 (Composite Subgradients [24]: $\mathcal{S}_{\mathbf{u}_{\mathcal{G}}}, \mathcal{S}_{\mathbf{o}_{\mathcal{G}}}$) *Let $D \subset \mathbb{R}^n$, $P \in \mathbb{IR}^{n_p}$, and $Z \in \mathbb{I}D$. Let $\mathbf{q} : P \to Z$ and $\mathcal{G} : D \times P \to \mathbb{R}^n$. Let $\mathbf{u}_{\mathcal{G}}, \mathbf{o}_{\mathcal{G}}$ be composite relaxations of $\mathcal{G}$ on $Z \times P$. The functions $\mathcal{S}_{\mathbf{u}_{\mathcal{G}}}, \mathcal{S}_{\mathbf{o}_{\mathcal{G}}} : \mathbb{R}^n \times \mathbb{R}^n \times \mathbb{R}^{n_p \times n} \times \mathbb{R}^{n_p \times n} \times P \to \mathbb{R}^{n_p \times n}$ are called* composite subgradients *of $\mathbf{u}_{\mathcal{G}}$ and $\mathbf{o}_{\mathcal{G}}$ on $Z \times P$, respectively, if for any $\psi^{cv}, \psi^{cc} : P \to \mathbb{R}^n$ and $\mathbf{s}_\psi^{cv}, \mathbf{s}_\psi^{cc} : P \to \mathbb{R}^{n_p \times n}$, the functions $\mathcal{S}_{\mathbf{u}_{\mathcal{G}}}(\psi^{cv}(\cdot), \psi^{cc}(\cdot), \mathbf{s}_\psi^{cv}(\cdot), \mathbf{s}_\psi^{cc}(\cdot), \cdot)$ and $\mathcal{S}_{\mathbf{o}_{\mathcal{G}}}(\psi^{cv}(\cdot), \psi^{cc}(\cdot), \mathbf{s}_\psi^{cv}(\cdot), \mathbf{s}_\psi^{cc}(\cdot), \cdot)$ are, respectively, subgradients of $\mathbf{u}_{\mathcal{G}}$ and $\mathbf{o}_{\mathcal{G}}$, provided ψ^{cv} and ψ^{cc} are, respectively, convex and concave relaxations of $\mathbf{q}$ on P and $\mathbf{s}_\psi^{cv}$ and $\mathbf{s}_\psi^{cc}$ are, respectively, subgradients of ψ^{cv} and ψ^{cc} on P.*

Note, that these definitions are included for completeness and future reference. The reader is directed to Stuber et al. [24] for the full definitions of relaxations and subgradients (omitted for brevity) as they pertain to implicit functions and to Scott et al. [29] for the generalized McCormick relaxation theory on which these results are based.

Optimization Problem 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 (1) is generalized as:

$$\begin{aligned} \phi^* = & \min_{\mathbf{p}\in P\subset\mathbb{R}^{n_p}} \phi(\mathbf{x}(\mathbf{p},\tau_0),\mathbf{x}(\mathbf{p},\tau_1),\ldots,\mathbf{x}(\mathbf{p},\tau_q),\ldots,\mathbf{x}(\mathbf{p},\tau_Q),\mathbf{p}) \\ \text{s.t. } & \dot{\mathbf{x}}(\mathbf{p},t) = \mathbf{f}(\mathbf{x}(\mathbf{p},t),\mathbf{p},t), \quad \forall t\in I=[t_0,t_f] \\ & \mathbf{x}(\mathbf{p},t_0) = \mathbf{x}_0(\mathbf{p}) \\ & \mathbf{g}(\mathbf{x}(\mathbf{p},\tau_0),\mathbf{x}(\mathbf{p},\tau_1),\ldots,\mathbf{x}(\mathbf{p},\tau_q),\ldots,\mathbf{x}(\mathbf{p},\tau_Q),\mathbf{p}) \le \mathbf{0} \end{aligned} \tag{3}$$

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 (3) 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 (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, 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 (4). As a result, the interpolated relaxation is subject to order 2 interpolation error and the McCormick relaxation is generally nonexpansive. For $t_{k-1}\le\tau_q\le t_k$, the interpolated state variable is given by:

$$\begin{aligned} \lambda &:= \frac{\tau_q - t_{k-1}}{t_k - t_{k-1}} \\ \mathbf{x}(\mathbf{p},\tau_q) &:= \lambda\mathbf{x}(\mathbf{p},t_k) + (1-\lambda)\mathbf{x}(\mathbf{p},t_{k-1}). \end{aligned} \tag{4}$$

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, (3) is reformulated from a

dynamic optimization problem to a standard (nonconvex) NLP as:

$$\begin{aligned} \phi^* = & \min_{(\hat{\mathbf{z}},\mathbf{p})\in Z\times P} \phi(\hat{\mathbf{z}},\mathbf{p}) \\ \text{s.t. } & \mathbf{h}(\hat{\mathbf{z}},\mathbf{p}) = \mathbf{0} \\ & \hat{\mathbf{z}}_0 = \mathbf{x}_0(\mathbf{p}) \\ & \mathbf{g}(\hat{\mathbf{z}},\mathbf{p}) \leq \mathbf{0} \end{aligned} \tag{5}$$

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 2 *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 2, we can reformulate the discretized problem (5) into the implicit form as:

$$\begin{aligned} \phi^* = & \min_{\mathbf{p}\in P} \phi(\mathbf{z}(\mathbf{p}),\mathbf{p}) \\ \text{s.t. } & \mathbf{g}(\mathbf{z}(\mathbf{p}),\mathbf{p}) \leq \mathbf{0}. \end{aligned} \tag{6}$$

It is worth noting that the full-space equality-constrained formulation (5) has $n_x \times (K+1) + n_p$ decision variables, whereas the reduced-space formulation (6) 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 [24]. 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 (5) is very closely related to that of the collocation approaches [30,31]. However, since this work is motivated by solving (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) 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 $\mathbf{h}$ when evaluated at $\mathbf{p} \in P$.

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). The conceptualization of these relaxations is illustrated in Figure 1. Specifically, in this section we develop methods for relaxing implicit functions via the relaxation of second-order implicit numerical integration schemes.

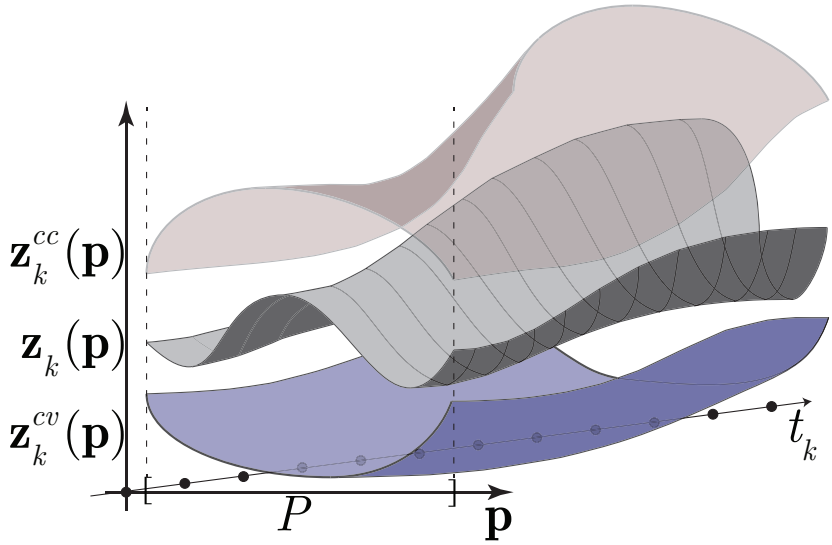

FIGURE 1 The implicit function $\mathbf{z}_k(\mathbf{p})$ is the approximation of the parametric solution to the initial value problem: $\dot{\mathbf{x}}(\mathbf{p},t) = \mathbf{f}(\mathbf{x}(\mathbf{p},t),\mathbf{p},t)$ at the discrete time points t_k illustrated by the curve segments on the center 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 $P^I \subset P$.

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:

$$\hat{\mathbf{z}}_{k+s} + \sum_{i=0}^{s-1} a_i \hat{\mathbf{z}}_{k+i} = \Delta t \sum_{j=0}^{s} b_j \mathbf{f}(\hat{\mathbf{z}}_{k+j}, \mathbf{p}, t_{k+j}). \tag{7}$$

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[32] 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:

$$\zeta_k^s(\hat{\mathbf{z}}_{k+s}, \ldots, \hat{\mathbf{z}}_k, \mathbf{p}) = \hat{\mathbf{z}}_{k+s} - \hat{\mathbf{z}}_{k+s-1} - \Delta t \sum_{j=0}^{s} b_j \mathbf{f}(\hat{\mathbf{z}}_{k+j}, \mathbf{p}, t_{k+j}) = \mathbf{0} \tag{8}$$

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

$$\xi_k^s(\hat{\mathbf{z}}_{k+s}, \ldots, \hat{\mathbf{z}}_k, \mathbf{p}) = \hat{\mathbf{z}}_{k+s} + \sum_{i=0}^{s-1} a_i \hat{\mathbf{z}}_{k+i} - \Delta t b_s \mathbf{f}(\hat{\mathbf{z}}_{k+s}, \mathbf{p}, t_{k+s}) = \mathbf{0} \tag{9}$$

where $\zeta_k^s, \xi_k^s : D \times \cdots \times D \times P \to D$. Note that in the above equations (8) and (9), the values $\hat{\mathbf{z}}_{k+i}$ are known for $i = 0, \ldots, s-1$. Therefore, (8) and (9) 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 (8) and (9) with respect to the $\hat{\mathbf{z}}_{k+s}$ variables are required. The respective Jacobian matrix of either formulation is given below by:

$$\mathbf{J}_k^s(\hat{\mathbf{z}}_{k+s}, \mathbf{p}) = \mathbf{I}_{n_x} - \Delta t b_s \mathbf{J}_{\mathbf{x}}(\hat{\mathbf{z}}_{k+s}, \mathbf{p}) \tag{10}$$

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 2.

Theorem 1 *Suppose the system has been discretized using an s-step parametric BDF or AM method. Let* $b := \min(s, k)$ *and* $\theta_{k-b}^b(\hat{\mathbf{z}}_k, \hat{\mathbf{z}}_{k-1}, \ldots, \hat{\mathbf{z}}_{k-b}, \mathbf{p}) = \mathbf{0}$ *for* $1 \leq k \leq K$ *with* $\theta \in \{\zeta, \xi\}$. *Let* $J_k^v(X, P) \in \mathbb{IR}^{n_x \times n_x}$ *be nonsingular (i.e., con-*

tains no singular matrices) with $X \in \mathbb{I}D$ *for* $1 \leq v \leq s$ *computed from* (10)*. 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* $\boldsymbol{\theta}_{k-b}^b(\hat{\mathbf{z}}_k^*, \hat{\mathbf{z}}_{k-1}^*, \ldots, \hat{\mathbf{z}}_{k-b}^*, \mathbf{p}^*) = \mathbf{0}$*. Further, suppose that for every* $\hat{\mathbf{z}}_k \in X$ *such that* $\hat{\mathbf{z}}_k \notin \text{int}(X)$*, we have* $\boldsymbol{\theta}_{k-b}^b(\hat{\mathbf{z}}_k, \hat{\mathbf{z}}_{k-1} \ldots, \hat{\mathbf{z}}_{k-b}, \mathbf{p}) \neq \mathbf{0}$ *for every* $(\hat{\mathbf{z}}_{k-1} \ldots, \hat{\mathbf{z}}_{k-b}, \mathbf{p}) \in X^b \times P$*. Then Assumption 2 holds with* $\mathbf{h}_k(\mathbf{z}_k(\mathbf{p}), \mathbf{p}) = \boldsymbol{\theta}_{k-b}^b(\mathbf{z}_k(\mathbf{p}), \mathbf{z}_{k-1}(\mathbf{p}), \ldots, \mathbf{z}_{k-b}(\mathbf{p}), \mathbf{p}) = \mathbf{0}$ *for* $1 \leq k \leq K$*.*

Proof This result follows directly from the semilocal implicit function theorem (Thm. 5.1.3 of Neumaier [27]) 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 $\boldsymbol{\theta}_0^1(\hat{\mathbf{z}}_1, \hat{\mathbf{z}}_0, \mathbf{p}) = \boldsymbol{\theta}_0^1(\hat{\mathbf{z}}_1, \mathbf{x}_0(\mathbf{p}), \mathbf{p})$. Continuous differentiability of $\boldsymbol{\xi}_0^1$ 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 [27] is satisfied. By assumption, there exists $(\hat{\mathbf{z}}_1^*, \hat{\mathbf{z}}_0^*, \mathbf{p}^*) \in X^2 \times P$ such that $\boldsymbol{\theta}_0^1(\hat{\mathbf{z}}_1^*, \hat{\mathbf{z}}_0^*, \mathbf{p}^*) = \mathbf{0}$ which may be restated as $\boldsymbol{\theta}_0^1(\hat{\mathbf{z}}_1^*, \mathbf{x}_0(\mathbf{p}^*), \mathbf{p}^*) = \mathbf{0}$. Further, for every $\hat{\mathbf{z}}_1 \in X$ such that $\hat{\mathbf{z}}_1 \notin \text{int}(X)$, we have $\boldsymbol{\theta}_0^1(\hat{\mathbf{z}}_1, \mathbf{x}_0(\mathbf{p}), \mathbf{p}) \neq \mathbf{0}$ for all $\mathbf{p} \in P$ since $\mathbf{x}_0(P) \subset D$. As such, Theorem 5.1.3 of Neumaier [27] 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 $\mathbf{h}_1(\mathbf{z}_1(\mathbf{p}), \mathbf{p}) = \boldsymbol{\theta}_0^1(\mathbf{z}_1(\mathbf{p}), \mathbf{x}_0(\mathbf{p}), \mathbf{p}) = \mathbf{0}$.

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 $\boldsymbol{\theta}_{k-b}^b(\hat{\mathbf{z}}_k, \hat{\mathbf{z}}_{k-1}, \ldots, \hat{\mathbf{z}}_{k-b}, \mathbf{p}) = \boldsymbol{\theta}_{k-b}^b(\hat{\mathbf{z}}_k, \mathbf{z}_{k-1}(\mathbf{p}), \ldots, \mathbf{z}_{k-b}(\mathbf{p}), \mathbf{p})$. Continuous differentiability of $\boldsymbol{\theta}_{k-b}^b$ 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 [27] 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 $\boldsymbol{\theta}_{k-b}^b(\hat{\mathbf{z}}_k^*, \hat{\mathbf{z}}_{k-1}^*, \ldots, \hat{\mathbf{z}}_{k-b}^*, \mathbf{p}^*) = \mathbf{0}$. As a result, there exists $\mathbf{p}^* \in P$ such that $\boldsymbol{\theta}_{k-b}^b(\hat{\mathbf{z}}_k^*, \mathbf{z}_{k-1}(\mathbf{p}^*), \ldots, \mathbf{z}_{k-b}(\mathbf{p}^*), \mathbf{p}^*) = \mathbf{0}$. Additionally, for each $\hat{\mathbf{z}}_k \in X$ such that $\hat{\mathbf{z}}_k \notin \text{int}(X)$, we have $\boldsymbol{\theta}_{k-b}^b(\hat{\mathbf{z}}_k, \mathbf{z}_{k-1}(\mathbf{p}) \ldots, \mathbf{z}_{k-b}(\mathbf{p}), \mathbf{p}) \neq \mathbf{0}$ for every $\mathbf{p} \in P$ and as such Theorem 5.1.3 of Neumaier [27] is satisfied with $\mathbf{J}_k^s(X, P)$ 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 $\mathbf{h}_k(\mathbf{z}_k(\mathbf{p}), \mathbf{p}) = \boldsymbol{\theta}_{k-b}^b(\mathbf{z}_k(\mathbf{p}), \mathbf{z}_{k-1}(\mathbf{p}) \ldots, \mathbf{z}_{k-b}(\mathbf{p}), \mathbf{p}) = \mathbf{0}$. 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 [27], Krawcyzk [33], Hansen-Sengupta [27], 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 1 is given by Theorem 2 below.

Theorem 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* $\mathbf{J}_c \equiv \text{mid}(J_k^s(Z_{k+s}, P))$ *(i.e., the elementwise midpoint of the interval matrix), and* $\Delta^* \equiv (J_k^s(Z_{k+s}, P))^U - \mathbf{J}_c$ *(i.e., the elementwise radius of* J_k^s*). 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* $J_k^s(Z_{k+s}, P)$ *contains no singular matrices.*

Proof This directly follows from Proposition 4.1.1 in Neumaier [27]. Note that b_s is a positive constant which depends on the method of choice and $\Delta t > 0$ by definition. ∎

Remark Theorem 2 provides a computational test for aiding users in choosing appropriate discretizations to avoid violating the hypotheses of Theorem 1 and ensuring satisfaction of Assumption 2.

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) [34,35]. Higher-order PILMS methods often exhibit better stability than explicit methods but they

do not exhibit A-stability [35]. 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 paper we consider only first- and second-order methods which are given by the residual equations:

$$\boldsymbol{\zeta}_k^2(\hat{\mathbf{z}}_{k+2}, \hat{\mathbf{z}}_{k+1}, \hat{\mathbf{z}}_k, \mathbf{p}) = \hat{\mathbf{z}}_{k+2} - \hat{\mathbf{z}}_{k+1} - (\Delta t/2)\,(\mathbf{f}(\hat{\mathbf{z}}_{k+2}, \mathbf{p}, t_{k+2}) + \mathbf{f}(\hat{\mathbf{z}}_{k+1}, t_{k+1})) \tag{11}$$

$$\boldsymbol{\xi}_k^1(\hat{\mathbf{z}}_{k+1}, \hat{\mathbf{z}}_k, \mathbf{p}) = \hat{\mathbf{z}}_{k+1} - \hat{\mathbf{z}}_k - \Delta t\mathbf{f}(\hat{\mathbf{z}}_{k+1}, \mathbf{p}, t_{k+1}) \tag{12}$$

$$\boldsymbol{\xi}_k^2(\hat{\mathbf{z}}_{k+2}, \hat{\mathbf{z}}_{k+1}, \hat{\mathbf{z}}_k, \mathbf{p}) = \hat{\mathbf{z}}_{k+2} - \tfrac{4}{3}\hat{\mathbf{z}}_{k+1} + \tfrac{1}{3}\hat{\mathbf{z}}_k - \tfrac{2}{3}\Delta t\mathbf{f}(\hat{\mathbf{z}}_{k+2}, \mathbf{p}, t_{k+2}) \tag{13}$$

where (11) is the second-order/two-step parametric Adams-Moulton method (i.e., trapezoidal method), (12) is the first-order parametric BDF method (i.e., backward/implicit Euler), and (13) is the second-order/two-step parametric BDF method. With respect to the equality constraints of (5), the equations given by (11)-(13) would form the K blocks of n_x equations of $\mathbf{h}$. This is expressed formally as:

$$\mathbf{h}(\hat{\mathbf{z}}_0, \hat{\mathbf{z}}_1, \ldots, \hat{\mathbf{z}}_K, \mathbf{p}) = \begin{pmatrix} \boldsymbol{\xi}_0^1(\hat{\mathbf{z}}_1, \hat{\mathbf{z}}_0, \mathbf{p}) \\ \boldsymbol{\theta}_0^2(\hat{\mathbf{z}}_2, \hat{\mathbf{z}}_1, \hat{\mathbf{z}}_0, \mathbf{p}) \\ \vdots \\ \boldsymbol{\theta}_{K-2}^2(\hat{\mathbf{z}}_K, \hat{\mathbf{z}}_{K-1}, \hat{\mathbf{z}}_{K-2}, \mathbf{p}) \end{pmatrix} = \mathbf{0} \tag{14}$$

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 2. The corresponding directed graph for the considered two-step methods is shown in Figure 3.

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 (11)-(13), which will also be extensible to general s-step PILMS methods as in (7). The first construction of relaxations and subgradients of implicit functions via an implicit integra-

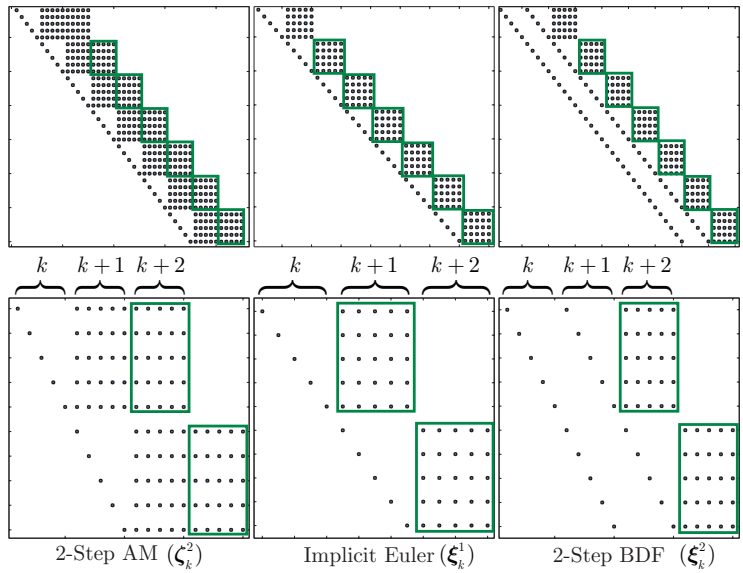

FIGURE 2 The sparsity patterns of the occurrence matrices of the systems of equations are illustrated for each of the three implicit integration schemes with $n_x = 5$. The sparsity patterns exhibit the block-diagonal structure corresponding to the timestep k which is exploited by the numerical equation solver and relaxation algorithms.

$$
\begin{aligned}
\mathbf{h}_1 &= \boldsymbol{\xi}_0^1 = (h_1, h_2, h_3, h_4, h_5) \\
\mathbf{h}_2 &= \boldsymbol{\theta}_0^2 = (h_6, h_7, h_8, h_9, h_{10}) \\
\mathbf{h}_3 &= \boldsymbol{\theta}_1^2 = (h_{11}, h_{12}, h_{13}, h_{14}, h_{15}) \\
&\vdots \qquad\quad \vdots \qquad\qquad\quad \vdots \\
\mathbf{h}_K &= \boldsymbol{\theta}_{K-2}^2 = (h_{5K-4}, h_{5K-3}, h_{5K-2}, h_{5K-2}, h_{5K})
\end{aligned}
$$

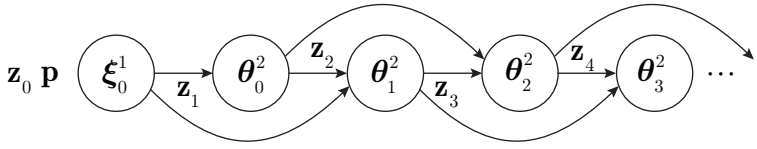

FIGURE 3 The directed graph corresponding to the occurrence matrices in Fig. 2 for the considered 2-step PILMS methods (with $n_x = 5$, $\boldsymbol{\theta} \in \{\boldsymbol{\xi}, \boldsymbol{\zeta}\}$) is depicted illustrating the sequential-block structure exploited when solving numerically the discretized system and constructing bounds and relaxations of the implicit functions $\mathbf{z}_k$ on P.

tion form was demonstrated in Stuber et al.[24]. However, in that work, only the implicit (backward) Euler scheme (12) was presented, and higher-order methods such as (11) and (13) 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.[24] for ensuring that relaxations of implicit functions are computable.

Assumption 3 *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* (10)*) on* $X \times P$.

Note that for consistency with (5), Assumption 2, and (14), 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 1 and 2 together provide a method for determining an appropriate $X \in \mathbb{I}D$ satisfying Assumption 3.1. Parametric interval methods may be used to automatically compute and verify an appropriate X interval satisfying Assumption 3.1 utilizing Theorems 1 and 2. In general, domain specific knowledge is required to furnish a valid initial guess for the interval X such that Assumption 3.1 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 [27] for constructing interval bounds of solutions of nonlinear algebraic systems. The development of this approach starts with the parametric mean value theorem [24].

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

$$\nabla_{\mathbf{x}} h_j(\mathbf{y}^j(\mathbf{p}), \mathbf{p})^{\mathsf{T}} (\mathbf{z}_k(\mathbf{p}) - \boldsymbol{\gamma}(\mathbf{p})) = -h_j(\boldsymbol{\gamma}(\mathbf{p}), \mathbf{p}), \; j = (k-1)n_x + 1, \ldots, k n_x \tag{15}$$

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. [24]).

Definition 7 ($\mathbf{M}_k, \mathbf{B}_k$) *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*

$$\mathbf{M}_k(\cdot) = \mathbf{B}_k(\mathbf{y}^{(k-1)n_x+1}(\cdot), \mathbf{y}^{(k-1)n_x+2}(\cdot), \ldots, \mathbf{y}^{kn_x}(\cdot), \cdot) \equiv \mathbf{Y}_k \begin{pmatrix} \nabla_{\mathbf{x}} h_{(k-1)n_x+1}(\mathbf{y}^{(k-1)n_x+1}(\cdot), \cdot)^{\mathsf{T}} \\ \nabla_{\mathbf{x}} h_{(k-1)n_x+2}(\mathbf{y}^{(k-1)n_x+2}(\cdot), \cdot)^{\mathsf{T}} \\ \vdots \\ \nabla_{\mathbf{x}} h_{kn_x}(\mathbf{y}^{kn_x}(\cdot), \cdot)^{\mathsf{T}} \end{pmatrix}. \tag{16}$$

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 $\mathbf{h}_k(\hat{\mathbf{z}}_k, \mathbf{p}) = (h_{(k-1)n_x+1}, h_{(k-1)n_x+2}, \ldots, h_{kn_x}) = \boldsymbol{\theta}^2_{k-2}(\hat{\mathbf{z}}_k, \mathbf{z}_{k-1}(\mathbf{p}), \mathbf{z}_{k-2}(\mathbf{p}), \mathbf{p})$ *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.* [24]*) when applied to* $h_j : X \times P \to X$, *as illustrated in* (15), *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 3.3.*

Remark Note that the definition of $\mathbf{M}_k$ (and $\mathbf{B}_k$) satisfies Lemma 3.15 of Stuber et al. [24]. 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 (15) and Definition 7, we can now form the n_x-dimensional system of equations for the k^{th} block as:

$$\mathbf{M}_k(\mathbf{p})(\mathbf{z}_k(\mathbf{p}) - \boldsymbol{\gamma}(\mathbf{p})) = -\mathbf{Y}_k \mathbf{h}_k(\boldsymbol{\gamma}(\mathbf{p}), \mathbf{p}) = -\mathbf{Y}_k \boldsymbol{\theta}^2_{k-2}(\boldsymbol{\gamma}(\mathbf{p}), \mathbf{z}_{k-1}(\mathbf{p}), \mathbf{z}_{k-2}(\mathbf{p}), \mathbf{p}), \ \forall \mathbf{p} \in P. \tag{17}$$

From this mean value form, we can now define the function ψ_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 8 (ψ_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 3.3. Define the function* $\psi_k : X \times M_k \times X \times X \times X \times P \to \mathbb{R}^{n_x}$ *such that* $\forall(\tilde{\boldsymbol{\gamma}}, \tilde{\mathbf{M}}, \tilde{\mathbf{z}}_k, \tilde{\mathbf{z}}_{k-1}, \tilde{\mathbf{z}}_{k-2}, \mathbf{p}) \in X \times M \times X \times X \times X \times P$, $\psi_k(\tilde{\boldsymbol{\gamma}}, \tilde{\mathbf{M}}, \tilde{\mathbf{z}}_k, \tilde{\mathbf{z}}_{k-1}, \tilde{\mathbf{z}}_{k-2}, \mathbf{p}) = \tilde{\mathbf{z}}^*_k$, *where the* i^{th} *component of* $\tilde{\mathbf{z}}^*_k$ *is given by the loop:*

for $i = 1, \ldots, n_x$ **do**

$$\tilde{z}^*_{k,i} := \tilde{\gamma}_i - \frac{b_{k,i}(\tilde{\boldsymbol{\gamma}}, \mathbf{z}_{k-1}(\mathbf{p}), \mathbf{z}_{k-2}(\mathbf{p}), \mathbf{p}) + \sum_{j<i} \tilde{m}_{ij}(\tilde{z}^*_{k,j} - \tilde{\gamma}_j) + \sum_{j>i} \tilde{m}_{ij}(\tilde{z}_{k,j} - \tilde{\gamma}_j)}{\tilde{m}_{ii}} \tag{18}$$

end.

Remark Note that this definition is analogous to Definition 3.17 in Stuber et al. [24] 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: $\mathbf{h}_1 = \boldsymbol{\xi}^1_0$, we will still utilize this definition of ψ_1 with $\mathbf{b}_1$ as $\mathbf{b}_1(\tilde{\boldsymbol{\gamma}}, \mathbf{z}_0(\mathbf{p}), \mathbf{z}_0(\mathbf{p}), \mathbf{p}) = \mathbf{Y}_1 \boldsymbol{\xi}^1_0(\tilde{\boldsymbol{\gamma}}, \mathbf{z}_0(\mathbf{p}), \mathbf{p})$.

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 8 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 ψ_k. Stuber et al. [24] 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 [24], which rely on relaxing $\mathbf{z}_k$ through the mean value form approximated using ψ_k. We follow this approach in this work.

With these definitions of ψ_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. [24] 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 ψ_k (Def. 8) and $\mathbf{M}_k$ (Def. 7).

Theorem 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 ψ_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 ψ_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.[24], we will show how relaxations of ψ_k as defined in Definition 8 (and their subgradients) are computable.

By the parametric mean value theorem [24], we have

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

By definition of $\mathbf{y}^i$, $i = (k-1)n_x + 1, \dots, kn_x$, 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 $i = (k-1)n_x + 1, \dots, kn_x$ (see Lemma 3.16 Stuber et al. [24] 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. 7) on P for every $i = (k-1)n_x + 1, \dots, kn_x$.

As a result, relaxations of $\mathbf{M}_k$ on P are calculable as:

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

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 ψ_k.

By Assumption 3.2, we have $0 \notin M_{k,ii} \supset m_{k,ii}(P),\ \forall i$. Then, for $i = 1, \ldots, n_x$, we can write:

$$z_{k,i}(\mathbf{p}) = \gamma_i(\mathbf{p}) - \frac{b_{k,i}(\gamma(\mathbf{p}), \mathbf{z}_{k-1}(\mathbf{p}), \mathbf{z}_{k-2}(\mathbf{p}), \mathbf{p}) + \sum_{j<i} m_{k,ij}(\mathbf{p})(z_{k,j}(\mathbf{p}) - \gamma_j(\mathbf{p})) + \sum_{j>i} m_{k,ij}(\mathbf{p})(z_{k,j}(\mathbf{p}) - \gamma_j(\mathbf{p}))}{m_{k,ii}(\mathbf{p})}.$$

By Definition 8, we can write:

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

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

By hypothesis, we have valid convex and concave relaxations $\mathbf{z}_{k-1}^{cv}(\mathbf{p}), \mathbf{z}_{k-2}^{cv}(\mathbf{p})$ and $\mathbf{z}_{k-1}^{cc}(\mathbf{p}), \mathbf{z}_{k-2}^{cc}(\mathbf{p})$ 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 $\gamma : P \to X$ is affine (both convex and concave). As a result, relaxations of the function $\mathbf{b}_k = \mathbf{Y}_k \theta_{k-2}$ (as defined in Definition 8) on P are calculable as

$$\mathbf{u}_{\mathbf{b}_k}(\gamma(\mathbf{p}), \gamma(\mathbf{p}), \mathbf{z}_{k-1}^{cv}(\mathbf{p}), \mathbf{z}_{k-1}^{cc}(\mathbf{p}), \mathbf{z}_{k-2}^{cv}(\mathbf{p}), \mathbf{z}_{k-2}^{cc}(\mathbf{p}), \mathbf{p}) \tag{19}$$

$$\mathbf{o}_{\mathbf{b}_k}(\gamma(\mathbf{p}), \gamma(\mathbf{p}), \mathbf{z}_{k-1}^{cv}(\mathbf{p}), \mathbf{z}_{k-1}^{cc}(\mathbf{p}), \mathbf{z}_{k-2}^{cv}(\mathbf{p}), \mathbf{z}_{k-2}^{cc}(\mathbf{p}), \mathbf{p}) \tag{20}$$

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 (19) and (20). It follows immediately that if we know $\mathbf{z}_k^{j,cv}$ and $\mathbf{z}_k^{j,cc}$, then

$$\begin{aligned}\mathbf{z}_k^{j+1,cv}(\cdot) &:= \bar{\mathbf{u}}_{\psi_k}(\gamma(\cdot), \gamma(\cdot), \mathbf{M}_k^{j,cv}(\cdot), \mathbf{M}_k^{j,cc}(\cdot), \mathbf{z}_k^{j,cv}(\cdot), \mathbf{z}_k^{j,cc}(\cdot), \mathbf{z}_{k-1}^{cv}(\cdot), \mathbf{z}_{k-1}^{cc}(\cdot), \mathbf{z}_{k-2}^{cv}(\cdot), \mathbf{z}_{k-2}^{cc}(\cdot), \cdot) \\ \mathbf{z}_k^{j+1,cc}(\cdot) &:= \bar{\mathbf{o}}_{\psi_k}(\gamma(\cdot), \gamma(\cdot), \mathbf{M}_k^{j,cv}(\cdot), \mathbf{M}_k^{j,cc}(\cdot), \mathbf{z}_k^{j,cv}(\cdot), \mathbf{z}_k^{j,cc}(\cdot), \mathbf{z}_{k-1}^{cv}(\cdot), \mathbf{z}_{k-1}^{cc}(\cdot), \mathbf{z}_{k-2}^{cv}(\cdot), \mathbf{z}_{k-2}^{cc}(\cdot), \cdot).\end{aligned}$$

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 $\mathbf{s}_{\mathbf{z}}^{cc}$. ∎

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**
    $X_i^a \leftarrow c_i + \sum_{j=1}^{n_p} (\mathbf{s_c}^\mathsf{T})_{ij} (P_j - \bar{p}_j)$ ▷ Interval arithmetic calculation
    $X_i^A \leftarrow C_i + \sum_{j=1}^{n_p} (\mathbf{s_C}^\mathsf{T})_{ij} (P_j - \bar{p}_j)$ ▷ 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 BlockRelax($\mathbf{h}_k$, 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**
    $\mathbf{c}, \mathbf{C}, \mathbf{s_c}, \mathbf{s_C} \leftarrow \mathrm{Aff}(\mathbf{z}_k^{j,cv}(\bar{\mathbf{p}}), \mathbf{z}_k^{j,cc}(\bar{\mathbf{p}}), \mathbf{s}_{\mathbf{z}_k}^{j,cv}(\bar{\mathbf{p}}), \mathbf{s}_{\mathbf{z}_k}^{j,cc}(\bar{\mathbf{p}}), \lambda, X, P, \bar{\mathbf{p}})$
    $\mathbf{z}_k^{j,a}(\mathbf{p}) \leftarrow \mathbf{c} + \mathbf{s_c}^\mathsf{T}(\mathbf{p} - \bar{\mathbf{p}})$ ▷ Affine relaxation lower bound
    $\mathbf{z}_k^{j,A}(\mathbf{p}) \leftarrow \mathbf{C} + \mathbf{s_C}^\mathsf{T}(\mathbf{p} - \bar{\mathbf{p}})$ ▷ Affine relaxation upper bound
    $\gamma^j(\mathbf{p}) \leftarrow \lambda \mathbf{z}_k^{j,a}(\mathbf{p}) + (1-\lambda)\mathbf{z}_k^{j,A}(\mathbf{p})$
    $\mathbf{s}_\gamma^j \leftarrow \lambda \mathbf{s_c} + (1-\lambda)\mathbf{s_C}$
    $\mathbf{M}^{j,cv}(\mathbf{p}) \leftarrow \mathbf{u}_{\mathbf{B}k}(\mathbf{z}_k^{j,a}(\mathbf{p}), \mathbf{z}_k^{j,A}(\mathbf{p}), \ldots, \mathbf{z}_k^{j,a}(\mathbf{p}), \mathbf{z}_k^{j,A}(\mathbf{p}), \mathbf{p})$
    $\mathbf{M}^{j,cc}(\mathbf{p}) \leftarrow \mathbf{o}_{\mathbf{B}k}(\mathbf{z}_k^{j,a}(\mathbf{p}), \mathbf{z}_k^{j,A}(\mathbf{p}), \ldots, \mathbf{z}_k^{j,a}(\mathbf{p}), \mathbf{z}_k^{j,A}(\mathbf{p}), \mathbf{p})$
    $\mathbf{s}_{\mathbf{M}}^{j,cv}(\mathbf{p}) \leftarrow \mathcal{S}_{\mathbf{u}_B}(\mathbf{z}_k^{j,a}(\mathbf{p}), \mathbf{z}_k^{j,A}(\mathbf{p}), \mathbf{s_c}, \mathbf{s_C}, \ldots, \mathbf{z}_k^{j,a}(\mathbf{p}), \mathbf{z}_k^{j,A}(\mathbf{p}), \mathbf{s_c}, \mathbf{s_C}, \mathbf{p})$
    $\mathbf{s}_{\mathbf{M}}^{j,cc}(\mathbf{p}) \leftarrow \mathcal{S}_{\mathbf{o}_B}(\mathbf{z}_k^{j,a}(\mathbf{p}), \mathbf{z}_k^{j,A}(\mathbf{p}), \mathbf{s_c}, \mathbf{s_C}, \ldots, \mathbf{z}_k^{j,a}(\mathbf{p}), \mathbf{z}_k^{j,A}(\mathbf{p}), \mathbf{s_c}, \mathbf{s_C}, \mathbf{p})$
    $\mathbf{z}_k^{j+1,cv}(\mathbf{p}) \leftarrow \bar{\mathbf{u}}_\psi(\gamma^j(\mathbf{p}), \gamma^j(\mathbf{p}), \mathbf{M}^{j,cv}(\mathbf{p}), \mathbf{M}^{j,cc}(\mathbf{p}), \mathbf{z}_k^{j,cv}(\mathbf{p}), \mathbf{z}_k^{j,cc}(\mathbf{p}), \mathbf{z}_{k-1}^{cv}(\mathbf{p}), \mathbf{z}_{k-1}^{cc}(\mathbf{p}), \mathbf{z}_{k-2}^{cv}(\mathbf{p}), \mathbf{z}_{k-2}^{cc}(\mathbf{p}), \mathbf{p})$
    $\mathbf{z}_k^{j+1,cc}(\mathbf{p}) \leftarrow \bar{\mathbf{o}}_\psi(\gamma^j(\mathbf{p}), \gamma^j(\mathbf{p}), \mathbf{M}^{j,cv}(\mathbf{p}), \mathbf{M}^{j,cc}(\mathbf{p}), \mathbf{z}_k^{j,cv}(\mathbf{p}), \mathbf{z}_k^{j,cc}(\mathbf{p}), \mathbf{z}_{k-1}^{cv}(\mathbf{p}), \mathbf{z}_{k-1}^{cc}(\mathbf{p}), \mathbf{z}_{k-2}^{cv}(\mathbf{p}), \mathbf{z}_{k-2}^{cc}(\mathbf{p}), \mathbf{p})$
    $\mathbf{s}_{\mathbf{z}_k}^{j+1,cv}(\mathbf{p}) \leftarrow \mathcal{S}_{\bar{\mathbf{u}}_\psi}(\gamma^j(\mathbf{p}), \gamma^j(\mathbf{p}), \mathbf{s}_\gamma^j, \mathbf{s}_\gamma^j, \mathbf{M}^{j,cv}(\mathbf{p}), \mathbf{M}^{j,cc}(\mathbf{p}), \mathbf{s}_{\mathbf{M}}^{j,cv}(\mathbf{p}), \mathbf{s}_{\mathbf{M}}^{j,cc}(\mathbf{p}), \mathbf{z}_k^{j,cv}(\mathbf{p}), \mathbf{z}_k^{j,cc}(\mathbf{p}), \mathbf{s}_{\mathbf{z}_k}^{j,cv}(\mathbf{p}), \mathbf{s}_{\mathbf{z}_k}^{j,cc}(\mathbf{p}), \mathbf{p})$
    $\mathbf{s}_{\mathbf{z}_k}^{j+1,cc}(\mathbf{p}) \leftarrow \mathcal{S}_{\bar{\mathbf{o}}_\psi}(\gamma^j(\mathbf{p}), \gamma^j(\mathbf{p}), \mathbf{s}_\gamma^j, \mathbf{s}_\gamma^j, \mathbf{M}^{j,cv}(\mathbf{p}), \mathbf{M}^{j,cc}(\mathbf{p}), \mathbf{s}_{\mathbf{M}}^{j,cv}(\mathbf{p}), \mathbf{s}_{\mathbf{M}}^{j,cc}(\mathbf{p}), \mathbf{z}_k^{j,cv}(\mathbf{p}), \mathbf{z}_k^{j,cc}(\mathbf{p}), \mathbf{s}_{\mathbf{z}_k}^{j,cv}(\mathbf{p}), \mathbf{s}_{\mathbf{z}_k}^{j,cc}(\mathbf{p}), \mathbf{p})$
  end for
  return $\mathbf{z}_k^{r,cv}(\mathbf{p}), \mathbf{z}_k^{r,cc}(\mathbf{p}), \mathbf{s}_{\mathbf{z}_k}^{r,cv}(\mathbf{p}), \mathbf{s}_{\mathbf{z}_k}^{r,cc}(\mathbf{p})$
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($X, P, \mathbf{x}_0(\mathbf{p})$) ▷ Standard McCormick relaxation of $\mathbf{x}_0$ on P
  $\mathbf{z}_1^{cv}(\mathbf{p}), \mathbf{z}_1^{cc}(\mathbf{p}), \mathbf{s}_{\mathbf{z}_1}^{cv}(\mathbf{p}), \mathbf{s}_{\mathbf{z}_1}^{cc}(\mathbf{p}) \leftarrow$BlockRelax($\mathbf{h}_1, X, \mathbf{p}, \bar{\mathbf{p}}, P, \mathbf{z}_0^{cv}, \mathbf{z}_0^{cc}, \mathbf{s}_{\mathbf{z}_0}^{cv}, \mathbf{s}_{\mathbf{z}_0}^{cc}, \mathbf{x}^L, \mathbf{x}^U, \mathbf{0}, \mathbf{0}, r, \lambda$)
  for $k \leftarrow 2$ **to** K **do**
    $\mathbf{z}_k^{cv}(\mathbf{p}), \mathbf{z}_k^{cc}(\mathbf{p}), \mathbf{s}_{\mathbf{z}_k}^{cv}(\mathbf{p}), \mathbf{s}_{\mathbf{z}_k}^{cc}(\mathbf{p}) \leftarrow$BlockRelax($\mathbf{h}_k, X, \mathbf{p}, \bar{\mathbf{p}}, P, \mathbf{z}_{k-1}^{cv}, \mathbf{z}_{k-1}^{cc}, \mathbf{s}_{\mathbf{z}_{k-1}}^{cv}, \mathbf{s}_{\mathbf{z}_{k-1}}^{cc}, \mathbf{z}_{k-2}^{cv}, \mathbf{z}_{k-2}^{cc}, \mathbf{s}_{\mathbf{z}_{k-2}}^{cv}, \mathbf{s}_{\mathbf{z}_{k-2}}^{cc}, r, \lambda$)
  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

Partition Convergence

In this section, we show that Algorithm 3 generates a relaxation which exhibits partition convergence given that Assumption 3 is entirely satisfied. This result follows directly from established properties of the implicit relaxation algorithm presented in Stuber et al. [24] and ensures that a B&B algorithm when used in conjunction with the relaxations developed herein will terminate in finite time.

Proposition 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 $\mathbf{h} = \mathbf{h}_1(\hat{\mathbf{z}}_1, \mathbf{p}) = \xi_0^1(\hat{\mathbf{z}}_1, \hat{\mathbf{z}}_0, \mathbf{p})$ trivially reduces to the form considered explicitly in the lower-bounding problem formulation (16) in Stuber et al.[24] and Lemma 4.3 in Stuber et al.[24] 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 [29]. 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 5 *Let X be defined as in Def. 4.1 of Stuber et al.[24] and suppose Assumption 4.2 of Stuber et al.[24] holds. Further, suppose the hypotheses of Proposition 4 hold. Then, after finitely many iterations, κ, the spatial B&B algorithm of Stuber et al.[24] 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.[24] hold from the pointwise convergence property of Proposition 4 and continuity assumptions without further modification. As a result, the finite convergence theorem (Thm. 4.9) in Stuber et al.[24] holds. ∎

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 [37] was created to implement the algorithm detailed above and is located at `https://github.com/PSORLab/EAGODifferential.jl`. The Intel MKL (2019 Update 2) [38] was used to perform all LAPACK [39,40] and BLAS [41] routines.

An affine lower-bounding problem was constructed using the relaxations evaluated at their midpoint and their respective subgradients [22]. 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 [42]. 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 [43], 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 (5). The DifferentialEquations.jl [44] 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.

Illustrative Examples

Example 1 (A Scalar Parametric ODE-IVP) Consider the simple scalar ODE-IVP presented in Section 4.1 of Sahlodin and Chachuat [19] with a single parameter:

$$\begin{aligned} \dot{x}(p,t) &= -x^2 + p, \quad t \in I = [0,1],\ p \in P = [-1,1] \\ x_0(p) &= 9, \quad p \in P. \end{aligned} \tag{21}$$

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 1 above and Theorem 5.1.8 in Neumaier [27], this verifies the existence of a unique implicit function in X. Both the two-step PILMS methods (11) and (13) were

used to construct bounds with and without the use of a parametric interval contraction method. In Figure 4, the relaxations are obtained using only the state bounds specified as X. As illustrated in Figure 4, 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[19]. 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 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.

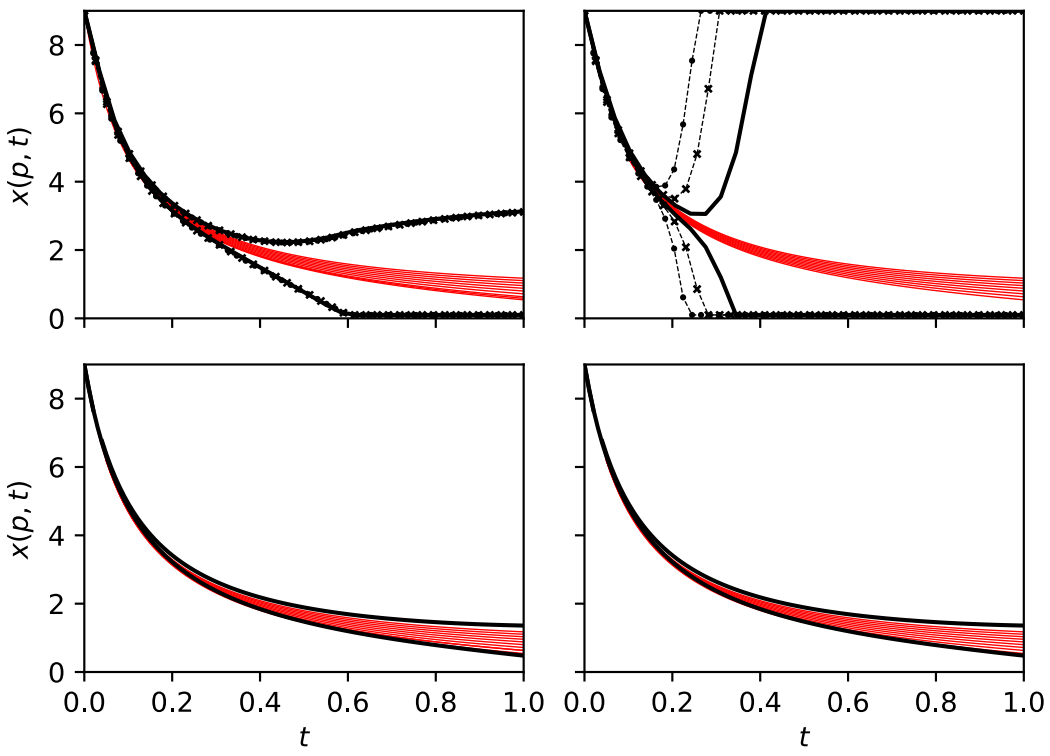

FIGURE 4 The results of Example 1 are illustrated here. Lower and upper bounds shown here are respectively the minimum and maximum values of $z_k^{cv}(\cdot)$ and $z_k^{cc}(\cdot)$ attained on P for each k. **Upper Left:** Bounds on $x(p,t)$ of (21) obtained by using the two-step AM method for $r = 3$ with $K = 30$ (black), $K = 40$ (light gray), and $K = 50$ (gray). **Upper Right:** Bounds on $x(p,t)$ of (21) 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 (21) 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 (21) obtained by using a second-order BDF method for $r = 3$ after applying 5 iterations of the parametric interval-Newton method using $K = 200$.

TABLE 1 The CPU times required to construct parametric interval (PI) bounds and convex/concave relaxation-based bounds (and associated subgradients) for Example 1 as well as the enclosure width at $t = 1$ associated with each variant used to construct relaxations.

Timesteps	$K = 10$	$K = 20$	$K = 30$	$K = 40$	$K = 50$
CPU Time for 2-step AM bounds, PI + relax (s)	152×10^{-6}	288×10^{-6}	436×10^{-6}	607×10^{-6}	731×10^{-6}
CPU Time for 2-step BDF bounds, PI + relax (s)	138×10^{-6}	284×10^{-6}	434×10^{-6}	582×10^{-6}	728×10^{-6}
2-step AM, PI + relax, enclosure width at $t = 1$	0.703	0.806	0.841	0.8550	0.8634
2-step AM, PI only, enclosure width at $t = 1$	1.812	1.906	1.938	1.955	1.955
2-step BDF, PI + relax, enclosure width at $t = 1$	0.703	0.810	0.901	0.937	0.951
2-step BDF, PI only, enclosure width at $t = 1$	0.713	0.822	0.916	0.954	0.969

In order to compare our bounding results with those of Sahlodin and Chachuat [19], we evaluate the enclosure width, $\Delta\omega_k$, at discretization point k corresponding to time t_k, as the 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 1.

The computational performance of the presented methods were compared with the timings presented by Sahlodin and Chachuat [19]. The results are contained in Table 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 [19] (Cinebench R15 score of 114). Therefore, we compute hardware-normalized timings by dividing the results of Sahlodin and Chachuat [19] by the factor 1.51. For $K = 100$ timesteps, both the two-step AM method and the two-step BDF method terminate in only 14×10^{-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 [19] takes a normalized time of between 93×10^{-4} s and 29×10^{-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 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 (22). 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 5, tight bounds on the characteristically-stiff system (22) 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.

$$\begin{aligned}\dot{x}_1(\mathbf{p},t) &= k_2 x_2 - k_1 x_1 \\ \dot{x}_2(\mathbf{p},t) &= k_1 x_1 - k_2 x_2\end{aligned} \tag{22}$$

The sharpness of the bounds provided here can be attributed to the fact that the right-hand side of (22) 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.

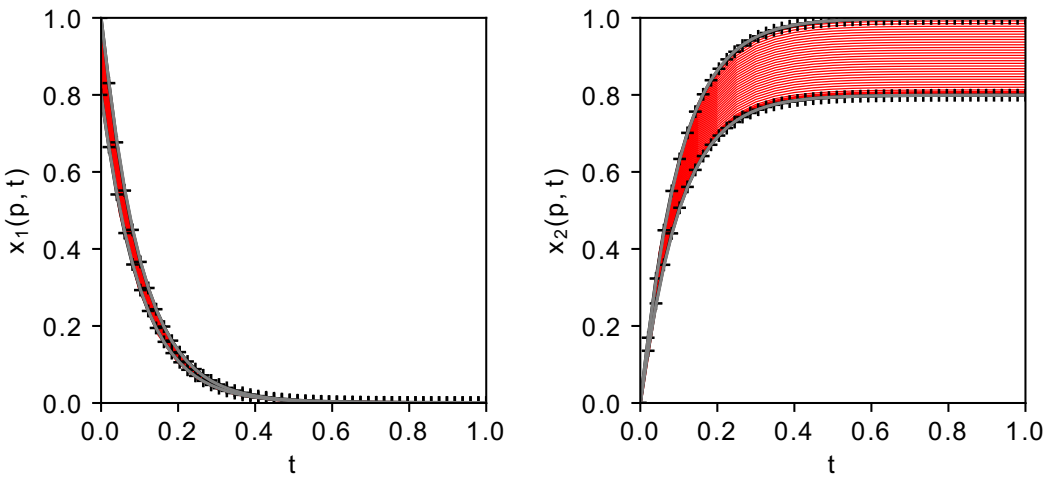

FIGURE 5 The results of Example 2 are illustrated. **(Left):** Bounds on $x_1(p,t)$ of (22) determined using the two-step BDF (black) and Adams-Moulton methods (gray-plus) using $K = 50$ discretization points. **(Right):** Bounds on $x_2(p,t)$ of (22) determined using the two-step BDF (black) and Adams-Moulton methods (gray-plus) using $K = 50$ discretization points.

Example 3 (Kinetic Parameter Estimation) Consider the kinetic mechanism problem first presented in Mitsos et al.[22] as an adaptation of the parameter estimation problem encountered for the oxygen addition to cyclohexadienyl radicals[10,45]. The reaction mechanism can be modeled as the ODE-IVP:

$$\begin{aligned}
\dot{x}_A(\mathbf{p},t) &= k_1 x_Z x_Y - c_{O_2}(k_{2f}+k_{3f})x_A + \frac{k_{2f}}{K_2}x_D + \frac{k_{3f}}{K_3}x_B - k_5 x_A^2 \\
\dot{x}_B(\mathbf{p},t) &= k_{3f} c_{O_2} x_A - \left(\frac{k_{3f}}{K_3} + k_4\right) x_B, \quad \dot{x}_Z(\mathbf{p},t) = -k_1 x_Z x_Y \\
\dot{x}_D(\mathbf{p},t) &= k_{2f} c_{O_2} x_A - \frac{k_{2f}}{K_2} x_D, \quad \dot{x}_Y(\mathbf{p},t) = -k_{1s} x_Z x_Y \\
x_{A,0} &= 0, x_{B,0} = 0, x_{D,0} = 0, x_{Y,0} = 0.4, x_{Z,0} = 140,
\end{aligned}$$

where x_j is the concentration of species $j \in \{A, B, D, Y, Z\}$. The constants are then given by $T = 273$, $K_2 =$

$46\exp(6500/T-18)$, $K_3 = 2K_2$, $k_1 = 53$, $k_{1s} = k_1 \times 10^{-6}$, $k_5 = 1.2\times10^{-3}$, and $c_{O_2} = 2\times10^{-3}$. Intensity versus time data is available in Stuber[46] 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[7]. 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.[22], 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.[24] and solved via the global optimization of implicit functions approach. State variable bounds on X provided in Stuber et al.[24] 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

$$\phi(\hat{\mathbf{z}}, \mathbf{p}) = \sum_{i=1}^{n} \left(I_i^c - I_i^d\right)^2 \tag{23}$$

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 2 and illustrated in Figure 6. 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 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.[24]. An implementation of the explicit Euler approach presented in Mitsos et al.[22] 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).

Example 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):

$$\frac{\partial x}{\partial t} = -\frac{\partial x}{\partial y} - Dax \tag{24}$$

where x is the nondimensional spatiotemporal-varying species concentration, $Da = k\tau$ is the 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

TABLE 2 The CPU times required to solve the kinetic parameter estimation problem (Ex. 3) using each bounding method for various K values after applying five iterations of the parametric interval-Newton method.

Solution Method	K	Iterations	Average time per iteration	Solution time	SSE at Solution
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	>2 h	N/A*
	200	6068	22×10^{-2}s	22.6min	13077.998
2-Step BDF	100	88408	81×10^{-3}s	>2 h	N/A*
	200	27600	26×10^{-2}s	>2 h	N/A*
Explicit Euler	100	>300,000	23×10^{-4}s	>2 h	N/A
	200	>300,000	24×10^{-4}s	>2 h	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.

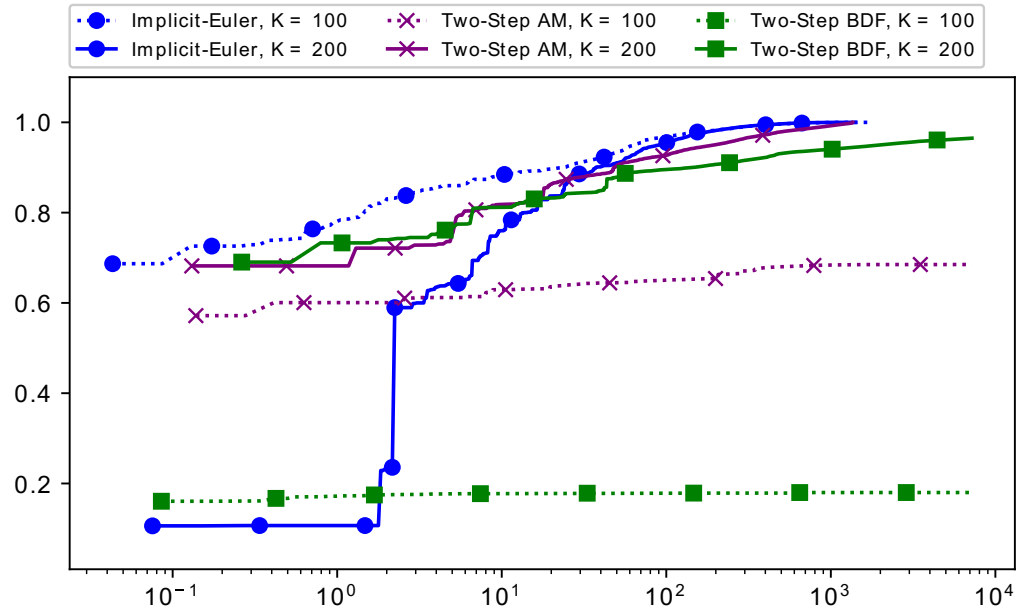

FIGURE 6 A convergence plot of each variation of the global optimization algorithm for timesteps $K = 100$ and $K = 200$ as demonstrated on the kinetic parameter estimation example (Ex. 3). For each K, the algorithm using the two-step AM method produces the tightest upper bound earlier in time. However, 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.

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 $k = 0.35\text{h}^{-1}$. 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:

$$\frac{d\tilde{\mathbf{x}}}{dt}(\mathbf{p}, t) = -\frac{\Delta\tilde{\mathbf{x}}}{\Delta y} - Da\tilde{\mathbf{x}} \tag{25}$$

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 (25) was numerically integrated using both implicit-Euler and an explicit fourth-order Runge-Kutta (RK4). As seen in Figure 7, 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.

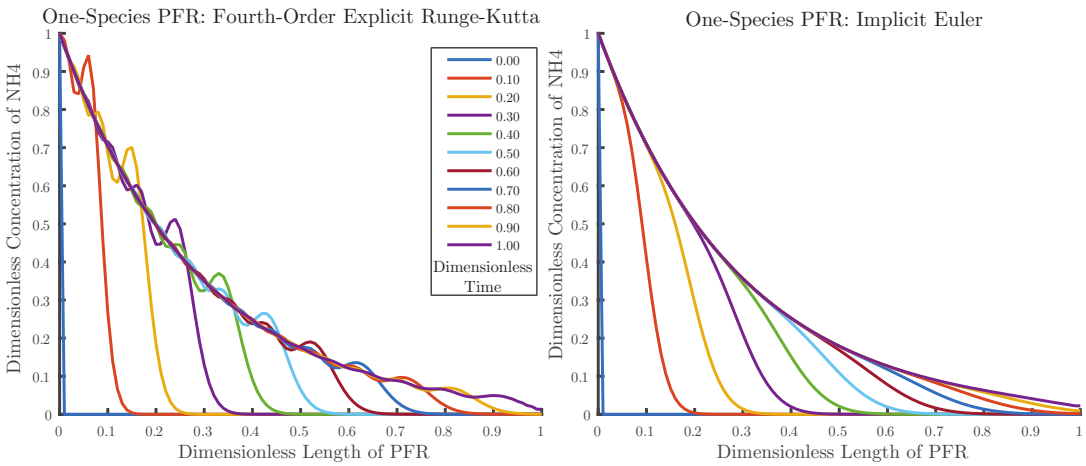

FIGURE 7 The one-species PFR model of Example 4 is simulated using (**Left**) the explicit fourth-order Runge-Kutta method and (**Right**) implicit (backward) Euler. The explicit method results in spurious oscillatory behavior of the concentration profiles which is not present using the implicit method.

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:

$$\begin{aligned} \phi^* &= \min_{p \in P} p \\ \text{s.t.} \quad & z_{K,\text{exit}}(p) - \lambda \leq 0 \end{aligned} \tag{26}$$

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 3.3 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 8. 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 8 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 3 (for which the two-step BDF method had worse performance than the two-step AM method) is significantly less stiff.

Example 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[47]. 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 Grady's[48] first activated sludge

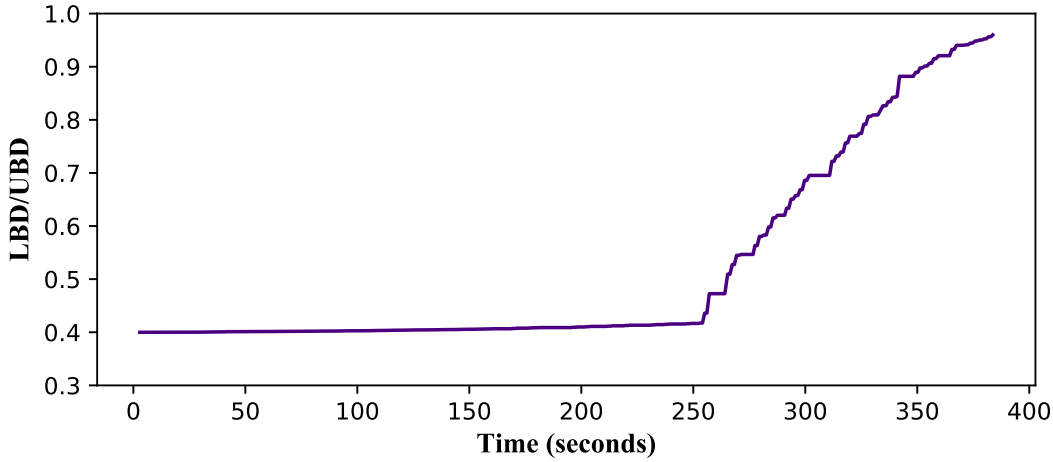

FIGURE 8 In this convergence plot, we see that the improvement of the bounds of (26) generated using the two-step BDF method in Example 4 remains stagnant until around 250 seconds when the these bounds begin to converge and the absolute convergence tolerance of 10^{-2} is satisfied for this example. At convergence, the relative gap is approximately $LBD/UBD = 0.99$.

model (ASM1) provides a suitable foundation for dynamic model development. The ASM1[48] 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[49]. 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 (27) describes the kinetics of the nine species of interest:

$$\begin{aligned}
\dot{x}_1(\mathbf{p},t) &= \tau(x_{in,1} - x_1) + 3.93(10 - x_1) + r_{11}\\
\dot{x}_2(\mathbf{p},t) &= \tau(x_{in,2} - x_2) + 0.484x_2 + r_6\\
\dot{x}_3(\mathbf{p},t) &= \tau(x_{in,3} - x_3) + 0.484x_3 + r_5\\
\dot{x}_4(\mathbf{p},t) &= \tau(x_{in,4} - x_4) + r_2\\
\dot{x}_5(\mathbf{p},t) &= \tau(x_{in,5} - x_5) + r_7\\
\dot{x}_6(\mathbf{p},t) &= \tau(x_{in,6} - x_6) + r_8\\
\dot{x}_7(\mathbf{p},t) &= \tau(x_{in,7} - x_7) + r_9\\
\dot{x}_8(\mathbf{p},t) &= \tau(x_{in,8} - x_8) + 0.484x_8 + r_4\\
\dot{x}_9(\mathbf{p},t) &= \tau(x_{in,9} - x_9) + 0.484x_9 + r_{10}
\end{aligned} \tag{27}$$

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 (28). 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)][48]. 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.

$$
\begin{aligned}
r_2 &= -\frac{\mu_H}{y_H}\frac{x_3 x_4}{K_s + x_4}\left(\frac{x_1}{K_O + x_1} + \frac{0.8K_O}{K_O + x_1}\frac{x_5}{K_{NO} + x_5}\right) + k_H x_3 \frac{x_8/x_3}{K_X + x_8/x_3}\left(\frac{x_1}{K_O + x_1}\right) \\
&\quad \ldots + \frac{0.8K_O}{K_O + x_1}\left(\frac{x_5}{K_{NO} + x_1}\right) \\
r_4 &= 0.92(b_H x_3 + b_A x_2) - k_H x_3 \frac{x_8/x_3}{K_X + x_8/x_3}\left(\frac{x_1}{K_O + x_1}\right) + \frac{0.8K_O}{K_O + x_1}\left(\frac{x_5}{K_{NO} + x_5}\right) \\
r_5 &= \mu_H \frac{x_3 x_4}{K_s + x_4}\left(\frac{x_1}{K_O + x_1} + 0.8\frac{K_O}{K_O + x_1}\frac{x_5}{K_{NO} + x_5}\right) - b_H x_3 \\
r_6 &= \mu_A \frac{x_2 x_6}{K_{NHA} + x_6}\left(\frac{x_1}{K_O + x_1}\right) - b_A x_2 \\
r_7 &= -0.088\mu_H \frac{x_3 x_4}{K_s + x_4}\frac{K_O}{K_O + x_1}\left(\frac{x_5}{K_{NO} + x_5}\right) + \frac{\mu_A}{y_A}\frac{x_2 x_6}{K_{NHA} + x_6}\left(\frac{x_1}{K_O + x_1}\right) \\
r_8 &= -0.068\mu_H \frac{x_3 x_4}{K_s + x_4}\left(\frac{x_1}{K_O + x_1} + \frac{K_O}{K_O + x_1}\frac{0.8x_5}{K_{NO} + x_5}\right) - \mu_A \frac{4.23x_6}{K_{NHA} + x_6}\left(\frac{x_1 x_2}{K_O + x_1}\right) + k_A x_7 x_3 \\
r_9 &= -k_A x_7 x_3 + k_H x_3 \frac{x_9/x_3}{K_X + x_8/x_3}\left(\frac{x_1}{K_O + x_1}\right) + \frac{0.8K_O}{K_O + x_1}\left(\frac{x_5}{K_{NO} + x_5}\right) \\
r_{10} &= 0.063(b_H x_3 + b_A x_2) - k_H \frac{(x_9/x_3)}{K_X + x_8/x_3}\left(\frac{x_1 x_3}{K_O + x_1}\right) + \frac{0.8K_O}{(K_O + x_1)}\left(\frac{x_5}{(K_{NO} + x_5)}\right) \\
r_{11} &= -\left(0.32\mu_H \frac{x_4 x_3}{K_s + x_4} + 18.04\mu_A \frac{x_6 x_2}{K_{NHA} + x_6}\right)\frac{x_1}{K_O + x_1}
\end{aligned}
\tag{28}
$$

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[50]. 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, (27), and (28) 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 9 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.

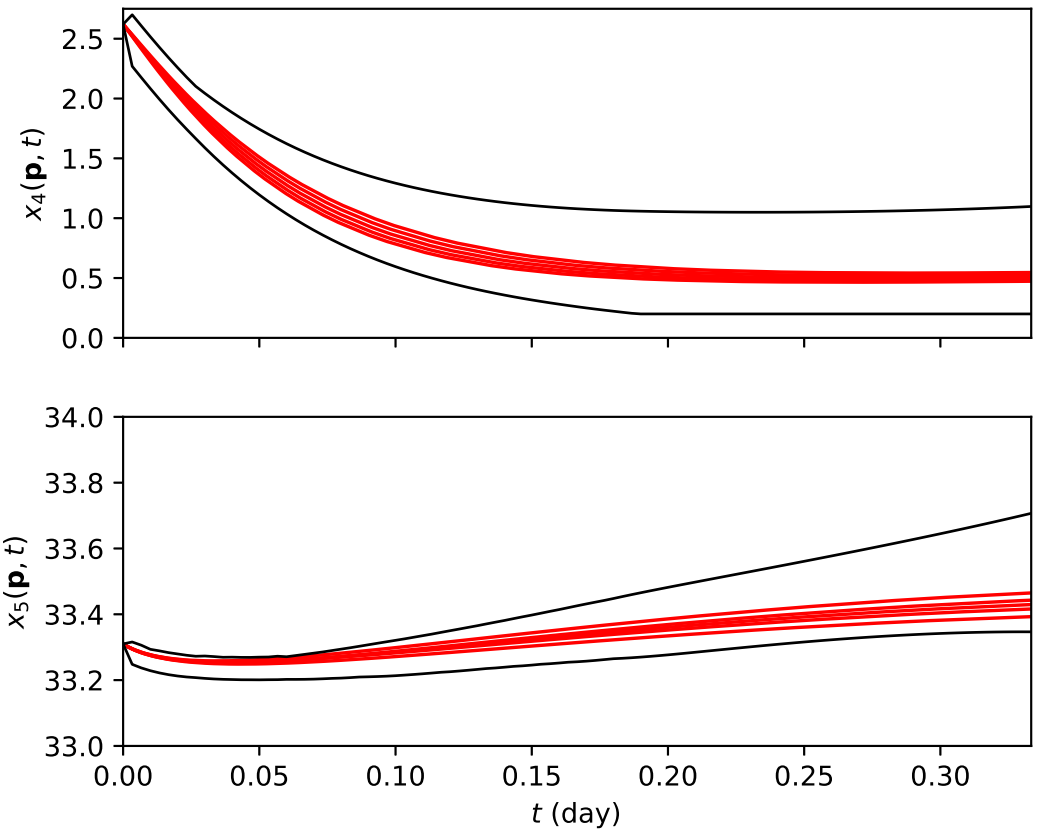

FIGURE 9 The results from Example 5 are illustrated. **(Top):** Bounds on $x_4(\mathbf{p}, t)$ determined using the two-step Adams-Moulton method using $K = 200$ discretization points. **(Bottom):** Bounds on $x_5(\mathbf{p}, t)$ determined using the two-step Adams-Moulton method using $K = 200$ discretization points.

Conclusion

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 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[24] 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.

Acknowledgements

This material is based upon work supported by the National Science Foundation under Grant Nos. 1560072, 1706343, 1932723, as well as the University of Connecticut. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation. The authors would like to thank Jacob Chicano for the original numerical simulation of the stiff PFR model (Example 4) as well as the reviewers for the helpful feedback.

references

1. Falk JE, Soland RM. An Algorithm for Separable Nonconvex Programming Problems. Management Science 1969;15(9):550–569.

2. Horst R, Tuy H. Global Optimization: Deterministic Approaches. Springer Berlin Heidelberg; 2013. https://doi.org/10.1007/978-3-662-03199-5.

3. Papamichail I, Adjiman CS. A rigorous global optimization algorithm for problems with ordinary differential equations. Journal of Global Optimization 2002;24(1):1–33.

4. Adjiman CS, Floudas CA. Rigorous convex underestimators for general twice-differentiable problems. Journal of Global Optimization 1996;9(1):23–40. https://doi.org/10.1007/BF00121749.

5. Adjiman CS, Androulakis IP, Maranas CD, Floudas CA. A global optimization method, αBB, for process design. Computers & Chemical Engineering 1996;20:S419–S424.

6. Singer AB, Barton PI. Global Solution of Optimization Problems with Parameter-Embedded Linear Dynamic Systems. Journal of Optimization Theory and Applications 2004;121:613–646.

7. Singer AB. Global dynamic optimization. PhD thesis, Massachusetts Institute of Technology; 2004.

8. McCormick GP. Computability of global solutions to factorable nonconvex programs: Part I — Convex underestimating problems. Mathematical Programming 1976;10(1):147–175. https://doi.org/10.1007/BF01580665.

9. Singer AB, Barton PI. Global optimization with nonlinear ordinary differential equations. Journal of Global Optimization 2006;34(2):159–190.

10. Singer AB, Taylor JW, Barton PI, Green WH. Global dynamic optimization for parameter estimation in chemical kinetics. The Journal of Physical Chemistry A 2006;110(3):971–976.

11. Scott JK, Barton PI. Convex relaxations for nonconvex optimal control problems. In: 50th IEEE Conference on Decision and Control; 2011. p. 1042–1047.

12. Scott JK, Barton PI. Improved relaxations for the parametric solutions of ODEs using differential inequalities. Journal of Global Optimization 2013 Sep;57(1):143–176. https://doi.org/10.1007/s10898-012-9909-0.

13. Scott JK, Chachuat B, Barton PI. Nonlinear convex and concave relaxations for the solutions of parametric ODEs. Optimal Control Applications and Methods 2013;34(2):145–163.

14. Shen K, Scott JK. Rapid and accurate reachability analysis for nonlinear dynamic systems by exploiting model redundancy. Computers and Chemical Engineering 2017;106:596–608.

15. Shen K, Scott JK. Mean Value Form Enclosures for Nonlinear Reachability Analysis. In: 2018 IEEE Conference on Decision and Control; 2018. p. 7112–7117.

16. Singer AB, Chachuat B, Barton PI, GDOC Version 1.0 manual. Massachusetts Institute of Technology: Cambridge, MA; 2005.

17. Serban R, Hindmarsh AC. CVODES: the sensitivity-enabled ODE solver in SUNDIALS. In: ASME 2005 international design engineering technical conferences and computers and information in engineering conference American Society of Mechanical Engineers; 2005. p. 257–269.

18. Sahlodin AM, Chachuat B. Discretize-Then-Relax Approach For State Relaxactions In Global Dynamic Optimization. Computer Aided Chemical Engineering 2010;p. 427–432.

19. Sahlodin AM, Chachuat B. Discretize-then-relax approach for convex/concave relaxations of the solutions of parametric ODEs. Applied Numerical Mathematics 2011;61(7):803–820.

20. Sahlodin AM, Chachuat B. Convex/concave relaxations of parametric ODEs using Taylor models. Computers and Chemical Engineering 2011;35:844–857.

21. Sahlodin AM, Chachuat B. Tight Convex and Concave Relaxations via Taylor Models for Global Dynamic Optimization. Computer Aided Chemical Engineering 2011;29:537–541.

22. Mitsos A, Chachuat B, Barton PI. McCormick-based relaxations of algorithms. SIAM Journal on Optimization 2009;20(2):573–601.

23. Yang X, Scott JK. Efficient Reachability Bounds for Discrete-Time Nonlinear Systems by Extending the Continuous-Time Theory of Differential Inequalities. In: 2018 Annual American Control Conference IEEE; 2018. p. 6242–6247. https://doi.org/10.23919/acc.2018.8431811.

24. Stuber MD, Scott JK, Barton PI. Convex and concave relaxations of implicit functions. Optimization Methods and Software 2015;30(3):424–460.

25. Sahinidis NV. BARON: A general purpose global optimization software package. Journal of Global Optimization 1996;8(2):201–205. http://dx.doi.org/10.1007/BF00138693.

26. Misener R, Floudas CA. ANTIGONE: algorithms for continuous/integer global optimization of nonlinear equations. Journal of Global Optimization 2014;59(2-3):503–526.

27. Neumaier A. Interval methods for systems of equations, vol. 37. Cambridge university press; 1990.

28. Floudas CA. Deterministic Global Optimization: Theory, Methods and Applications. Nonconvex Optimization and Its Applications, Springer US; 2000. https://doi.org/10.1007/978-1-4757-4949-6.

29. Scott JK, Stuber MD, Barton PI. Generalized McCormick relaxations. Journal of Global Optimization 2011;51(4):569–606.

30. Tsang T, Himmelblau D, Edgar T. Optimal control via collocation and non-linear programming. International Journal of Control 1975;21(5):763–768.

31. Biegler LT. Solution of dynamic optimization problems by successive quadratic programming and orthogonal collocation. Computers & chemical engineering 1984;8(3-4):243–247.

32. Curtiss C, Hirschfelder JO. Integration of stiff equations. Proceedings of the National Academy of Sciences of the United States of America 1952;38(3):235.

33. Krawczyk R. Interval iterations for including a set of solutions. Computing 1984 Mar;32(1):13–31. https://doi.org/10.1007/BF02243016.

34. Gautschi W. Numerical Analysis. Springer Science & Business Media, New York; 2012.

35. Hairer E, Wanner G. Solvig Ordinary Differential Equations II: Stiff and Differential-Algebraic Problems. Springer, Heidelberg; 1991.

36. Bezanson J, Edelman A, Karpinski S, Shah VB. Julia: A fresh approach to numerical computing. SIAM review 2017;59(1):65–98.

37. Wilhelm M, Stuber MD, EAGO: Easy Advanced Global Optimization Julia Package. PSORLab; 2018. https://github.com/PSORLab/EAGO.jl, EAGO: Easy Advanced Global Optimization Julia Package.

38. F G, Nguyen KT. Intel Math Kernel Library 2019 Update 2 Release Notes; 2019, https://software.intel.com/en-us/mkl.

39. Anderson E, Bai Zea. LAPACK Users' guide. SIAM; 1999.

40. Wang E, Zhang Q, Shen B, Zhang G, Lu X, Wu Q, et al. Intel Math Kernel Library. In: High-performance computing on the Intel Xeon Phi Springer; 2014.p. 167–188.

41. Blackford LS, Petitet A, Pozo R, Remington K, Whaley RC, Demmel J, et al. An updated set of basic linear algebra subprograms (BLAS). ACM Transactions on Mathematical Software 2002;28(2):135–151.

42. CPLEX, IBM ILOG, CPLEX Optimizer: User's Manual. International Business Machines Corp.; 2017. urlhttp://www-01.ibm.com/software/integration/optimization/cplex-optimizer/.

43. Ryoo HS, Sahinidis NV. Global optimization of nonconvex NLPs and MINLPs with applications in process design. Computers & Chemical Engineering 1995;19(5):551–566.

44. Rackauckas C, Nie Q. DifferentialEquations.jl–A performant and feature-rich ecosystem for solving differential equations in Julia. Journal of Open Research Software 2017;5(1).

45. Taylor JW, Ehlker G, Carstensen HH, Ruslen L, Field RW, Green WH. Direct measurement of the fast, reversible addition of oxygen to cyclohexadienyl radicals in nonpolar solvents. The Journal of Physical Chemistry A 2004;108(35):7193–7203.

46. Stuber MD. Evaluation of Process Systems Operating Envelopes. PhD dissertation, Massachusetts Institute of Technology; 2013.

47. Rosso D, Larson LE, Stenstrom MK. Aeration of large-scale municipal wastewater treatment plants: state of the art. Water Science and Technology 2008;57(7):973–978.

48. Henze M, Gujer W, Mino T, Matsuo T, Wentzel MC, Marais GvR, et al. Activated sludge model no. 2d, ASM2d. Water Science and Technology 1999;39(1):165–182. https://doi.org/10.2166/wst.1999.0036.

49. Lindberg CF. Control and estimation strategies applied to the activated sludge process. PhD thesis, Uppsala University Stockholm, Sweden; 1997.

50. Alikhania J, Massoudiehb A, Bhowmika UK. 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) 2016;p. 555–560.