

Two-Phase Thermofluidic Engines for Low-Grade Heat Recovery: System Analysis and Supporting Algorithms

Yukun WANG

Centre for Process Systems Engineering

Department of Chemical Engineering

Imperial College London

This dissertation is submitted in part fulfilment
of the requirements for the degree:
Doctor of Philosophy In Chemical Engineering

Oct, 2020

DECLARATION OF ORIGINALITY

I hereby declare that except where specific reference is made to the work of others, the contents of this dissertation are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other university. This dissertation contains fewer than 100,000 words including appendices, bibliography, footnotes, tables and equations.

Yukun Wang
Oct 2020

DECLARATION OF COPYRIGHT

The copyright of this thesis rests with the author and is made available under a Creative Commons Attribution Non-Commercial No Derivatives license. Researchers are free to copy, distribute or transmit the thesis on the condition that they attribute it, that they do not use it for commercial purposes and that they do not alter, transform or build upon it. For any reuse or redistribution, researchers must make clear to others the license terms of this work.

Acknowledgements

PhD has been a long, fun yet difficult journey. There is a long list of people I would like to thank for their help and care during my PhD studies.

My deepest gratitude and respect go to my supervisor, Dr Benoît Chachuat, who has led my way through the PhD with incredible patience. I have always felt extremely lucky to have his supervision, support and guidance in research, without which it would not have been possible to accomplish this work. All ideas and contributions in this work owe a debt of thanks to his visions and help. I also would like to thank Prof Christos Markides for bringing such an interesting topic of study, and providing adequate information and help along the way. I also want to thank Prof Nilay Shah for mentoring and providing insights throughout the PhD.

I shall also thank the research group seniors, Mario, Channarong, Nikola and Muxin. They were very generous in giving out advice and helped me understand the research topics at early stages of my PhD. I shall not miss my fellow office mates on this list – Amjad, Andres, Andrea, Ariel, Daniel, Lucian, Victoria, Quek, Antonio, Hunnan, Kyungjae and Yanlin. PhD would have been a lot less fun without all the coffee breaks, drinks and inspirational chats with them.

The Emperor's Gate gang – Linda, Yingjian and Xue, made my PhD life much less boring. I will miss all the hangouts and the exchange of thoughts (mostly whining), whether related to research or not. Besides, I really appreciate the help from Xiaowen who was the best and the kindest flatmate ever!

In addition to my supervisor, colleagues and friends, I am also sincerely grateful to my parents. Despite my occasional frustration and temper tantrums, they have never stopped loving and caring for me. I shall share more time with them in the future.

Apart from all these brilliant people around me, I also want to thank the development team for the toolbox CORA, who very kindly helped me construct my numerical cases studies in CORA, clarify my questions and comprehend key observations.

Last but not least, I would like to extend gratitude to my examiners, Dr Vivek Dua and Prof Amparo Galindo, for their comments and corrections to this thesis. The examination was an enjoyable and insightful experience. I am also grateful for the financial support from the Department of Chemical Engineering at Imperial College London for a Doctoral Scholarship.

Publications and Conferences

Parts of this thesis have been published in the following peer-reviewed journal and conference proceedings:

Yukun Wang, Christos N. Markides, and Benoît Chachuat. Optimization-based Investigations of a Thermofluidic Engine for Low-grade Heat Recovery. *IFAC PapersOnLine*, 51(18):690 – 695, 2018;

Yukun Wang, Christos N. Markides, and Benoît Chachuat. Optimization-based investigations of a two-phase thermofluidic oscillator for low-grade heat conversion. *BMC Chemical Engineering*, 1(1):20, 2019;

and presented at the following conferences:

Yukun Wang, Jai Rajyaguru and Benoît Chachuat. Exploiting Structure in Direct Simultaneous Methods for Global Dynamic Optimisation. *In 2017 AIChE Annual Meeting*, Minneapolis, United States. Paper 599a;

Yukun Wang, Christos N. Markides and Benoît Chachuat. Model-Based Analysis of a Thermofluidic Engine for Low-Grade Heat Recovery: Accounting for Irreversible Thermal Losses. *In 2018 AIChE Annual Meeting*, Pittsburgh, United States. Paper 734d.

Abstract

This thesis considers a two-phase thermofluidic oscillator known as the Non-Inertive-Feedback Thermofluidic Engine (NIFTE), that is capable of utilizing heat supplied at a steady temperature to induce persistent thermal-fluid oscillations. The NIFTE is appealing for its simplicity and ability to operate across small temperature differences, as low as 30 °C on current prototypes. But it is also expected for these prototypes to exhibit low efficiencies relative to conventional heat recovery technologies that target higher-grade heat conversion.

The first part of this work involves a system analysis based on a nonlinear model of the NIFTE, which is extended to encompass irreversible thermal losses. The NIFTE is predicted to exhibit multiple cyclic steady states (CSS) for certain design configurations, either stable or unstable, a behavior that had never been hypothesized. A parametric analysis of the main design parameters of the NIFTE is conducted using local optimization techniques based on randomized multistart. The results confirm that failure to include the irreversible thermal losses in the NIFTE model can grossly overpredict its performance, especially over extended parameter domains. The optimization potential of this technology is also assessed by conducting a multi-objective optimization. Our results reveal that most of the optimization potential is achievable via targeted modifications of three design parameters only. The Pareto frontier between exergetic efficiency and power output is also found to be highly sensitive to these optimized parameters.

Rigorous analysis and optimization of the NIFTE calls for the implementation of global optimization due to the presence of multiple cyclic steady state and other uncertainties, and this is notoriously difficult. The second part of the work focuses on improving ODE bounding methods that form an essential step in global dynamic optimization. Unlike most available bounding methods, a discretization approach is followed in this thesis to convert the differential equations to sparse algebraic equation systems; and their block structure is exploited. A simple block diagonal decomposition strategy is shown to result in significant overestimation due to the wrapping effect. Therefore, a recursive block decomposition

strategy is developed, and it fully retains the inter-dependency between the blocks. Numerical case studies reveal that the proposed discretization approach can potentially outperform the state-of-the-art set-propagation methods for nonlinear dynamic ODE bounding, both in terms of bounds tightness and CPU time. However, for larger systems with more complex dynamics, such as the NIFTE, our method may end up being more computationally demanding than the set-propagation method due to the much larger number of operations involved. Moreover, all of these methods are prone to fail when either a large uncertainty set or a long time horizon is considered.

To circumvent the numerical challenges associated with solving nonlinear ODEs, a switched-linear version of the NIFTE model is developed and validated. This model has three distinct modes for its dynamics, between which the NIFTE switches continuously. The conventional approach of bounding such linear hybrid automata using set-propagation methods proves useful for small-scale problems, but not for the NIFTE. Instead, a new optimization-based approach is developed in the final chapter of this thesis. The switched-linear dynamical system is first discretized. The resulting algebraic system is then reformulated using mixed-integer linear constraints, which can be solved using state-of-the-art MILP solvers. This proposed optimization-based approach is shown to generate tight state enclosures, especially in the presence of large uncertainties where set-propagation ODE bounding methods fail. However, the run-time complexity grows exponentially with the length of the time horizon.

All the numerical methods developed in this thesis are implemented in the in-house toolkit CRONOS, which is publicly available.

Contents

List of Figures

List of Tables

List of Algorithms

List of Listings

1	Introduction	1
1.1	Thermofluidic engines and other Heat recovery technologies	3
1.2	Modeling and optimization of thermofluidic engines	4
1.3	Dynamic optimization technology	5
1.4	Main objectives and thesis outline	8
2	NIFTE Principles and Modeling	10
2.1	NIFTE Principles	10
2.2	Review of existing NIFTE models	12
2.3	Mathematical derivation of the NIFTE models	14
2.3.1	Fluid domain	15
2.3.2	Thermal domain	16
2.3.3	Complete NIFTE models	19
2.3.4	Expansion of NIFTE models as switched-linear systems	21
2.3.5	Key performance indicators	23
2.4	Numerical simulation results	23
2.4.1	Case study definition	23
2.4.2	Numerical integration	25
2.4.3	NIFTE model comparison	27

2.5	Summary of the contributions	28
3	Model-Based Analysis and Optimization of NIFTE	30
3.1	Introduction	30
3.2	Methods	31
3.2.1	Cyclic steady-state detection and analysis of the NIFTE	31
3.2.2	Multi-objective optimization of the NIFTE	33
3.3	Results and discussion	34
3.3.1	Cyclic-steady-state multiplicity and stability	35
3.3.2	Parametric analysis of (R_f, L_f)	39
3.3.2.1	NTP model	39
3.3.2.2	NTP-TL model	41
3.3.3	Parametric analysis of (C_p, L_p)	43
3.3.4	Parametric analysis of (C_d, L_d)	44
3.3.5	Parametric analysis of K_{th}	45
3.3.6	Multi-objective optimization by varying multiple parameters	46
3.4	Summary of the contributions	47
4	Background on Set-based Computing	52
4.1	Introduction	52
4.2	Bounding of factorable functions	53
4.2.1	Interval Analysis	54
4.2.2	Polynomial Model Estimators	55
4.2.3	Other Relaxation Techniques	57
4.2.4	Implementations in CRONOS	58
4.2.4.1	Interval arithmetic-based relaxations	58
4.2.4.2	Polynomial model-based relaxation	59
4.3	Polyhedral relaxations of factorable programs	61
4.3.1	Polyhedral relaxations	61
4.3.2	Implementation in MC++	63
4.4	Bounding the Solutions of Implicit Algebraic Equations	66
4.4.1	Iterative methods	68
4.4.2	Gaussian elimination methods	72

4.4.3	Implementations in CRONOS	74
4.5	Bounding the solutions of ordinary differential equations	78
4.5.1	Review of set-propagation approaches	78
4.5.1.1	Continuous-time set propagation approaches	78
4.5.1.2	Discrete-time set-propagation approaches	80
4.5.2	Implementations in CRONOS	81
4.6	Summary	85
5	Discretization and Block Decomposition Approach to Bounding of Nonlinear ODEs	87
5.1	Introduction	87
5.2	Background	88
5.2.1	Review on discretization methods for differential equations	88
5.3	Methodology: Block decomposition strategy	90
5.3.1	Block-diagonal preconditioning	91
5.3.2	Recursive block preconditioning	95
5.3.2.1	Recursive iterative methods	96
5.3.2.2	Recursive Gaussian elimination	101
5.3.3	Implementations in CRONOS	102
5.4	Case study: Series reaction in batch reactor	103
5.4.1	Implementation in CRONOS	104
5.4.2	Reachable set enclosures	104
5.5	Case study: Nonlinear thermofluidic engine	107
5.5.1	Case 1: Reachable set enclosure subject to $\hat{\mathbf{x}}_0 \in [0.9, 1.1]$ $\hat{\mathbf{x}}_{\text{ref}}$	108
5.5.1.1	Implementation and comparison in CRONOS	108
5.5.1.2	Comparison with CORA (release 2018)	110
5.5.2	Case 2: Reachable set enclosure subject to $\hat{\mathbf{x}}_0 \in [0.5, 2]$ $\hat{\mathbf{x}}_{\text{ref}}$	111
5.6	Summary	112
6	Mixed-integer Optimization Approach to Bounding of Linear Switched ODEs	116
6.1	Introduction	116
6.2	Reachability analysis of hybrid automata (HA)	117
6.3	Methodology	119

6.3.1	Time discretization approach	121
6.3.2	Mixed-integer linear optimization formulation	122
6.3.3	Implementation in CRONOS	123
6.4	Case study: PI-controlled DC motor	124
6.4.1	Problem definition	124
6.4.2	Implementation in CRONOS	126
6.4.3	Implementation in CORA	126
6.4.4	Reachable set enclosures	127
6.5	Case study: Piecewise-linear thermofluidic engine	130
6.5.1	Implementation in CRONOS	130
6.5.2	Implementation in CORA	130
6.5.3	Verification of model implementations	130
6.5.4	Reachable set enclosures	131
6.5.4.1	Case 1: reachable set enclosure subject to $\hat{\mathbf{x}}_0 \in [0.9, 1.1] \hat{\mathbf{x}}_{\text{ref}}$.	132
6.5.4.2	Case 2: Reachable set enclosure subject to $\hat{\mathbf{x}}_0 \in [0.5, 2] \hat{\mathbf{x}}_{\text{ref}}$.	134
6.6	Summary	135
7	Conclusions	142
7.1	Summary of contributions	142
7.2	Future directions	144
	References	147
	Appendix A gPROMS ModelBuilder script for the NIFTE PWLTP-TL model	163
	Appendix B Example code snippet for the full discretization set-propagation approach implemented in CRONOS	167
	Appendix C Example code snippet for the mixed-integer optimization approach implemented in CRONOS	170
	Appendix D Copyright permissions	174

List of Figures

1.1	(a): Schematic illustration of a typical Rankine cycle; (b): Temperature (T) - entropy (s) diagram of a typical Rankine cycle.	2
2.1	NIFTE fluid pump schematic	11
2.2	The temperature or power profile along the hot heat exchanger (HHX) and cold heat exchanger (CHX) assumed in the (a) LTP, (b) CTD, (c) NTP / NTP-TL models for the NIFTE device.	13
2.3	Complete NIFTE circuit.	17
2.4	The temperature profile along the hot heat exchanger (HHX) and cold heat exchanger (CHX) assumed in the piecewise affine NTP-TL model for the NIFTE device.	22
2.5	Bode plot of thermal loss impedance.	26
2.6	Time-simulation of P_{ad} , P_d , P_p , U_f , and U_p in the NIFTE device model using the NTP, the NTP-TL, and the PWLTTP-TL models, with $P_{ad,0} = 0.01P^\circ$, $U_{f,0} = 0.01U^\circ$, and the other state variables equal to zero.	28
3.1	Pressure-volume phase diagrams of a NIFTE simulated over an extended time horizon.	38
3.2	Predicted effect of R_f and L_f on NIFTE performance using the NTP model with n-pentane as the working fluid.	40
3.3	Predicted effect of R_f and L_f on NIFTE performance using the NTP-TL model with n-pentane as the working fluid.	42
3.4	Predicted effect of C_p and L_p on NIFTE performance using the NTP-TL model with n-pentane as the working fluid.	49
3.5	Predicted effect of C_d and L_d on NIFTE performance using the NTP-TL model with n-pentane as the working fluid.	50

3.6	Predicted Pareto frontiers for various parameter subsets of the NTP-TL model with n-pentane as the working fluid.	51
4.1	Illustration of a polynomial model estimator for a function $f(x)$ on $X = [x^L, x^U]$	56
4.2	Relaxations for $f(x) = \frac{x-x^2}{\exp(x)}$ with $x \in [0, 2]$ using interval arithmetic and Chebyshev polynomial model.	62
4.3	Polyhedral relaxations for $f(x) = \frac{x-x^2}{\exp(x)}$ for $x \in [0, 2]$	66
4.4	Comparison of reachable set enclosures to (4.5.5) with $x_1(0) = 1.2, x_2(0) = 1.2, p_1 \in [2.99, 3.01]$ and $p_2 \in [0.99, 1.01]$ using: inner approximation (INNER); continuous set-valued integration with interval-based estimators (DINEQI) and continuous set-valued integration with 5th-order Chebyshev model-based estimators (DINEQPM).	85
5.1	Comparison of bounds on the solution set of (5.3.7) for $u_0 \in [-0.05, 0.05]$, $v_0 \in [0.95, 1.05]$, $h = 0.05$ and $N = 50$	94
5.2	Comparison of reachable set enclosures to (5.4.1) with <i>a priori</i> state bounds $\mathbf{X}^0 = [0, 1]^2$ using: inner approximation (INNER); continuous set-valued integration (DINEQ) with Chebyshev model-based estimators; discretized set-valued integration with recursive block decomposition (DISCR/RECUR); and discretized set-valued integration with diagonal block decomposition (DISCR/DIAG). A step size of either 0.02 or 0.01 is used in the discretized set-valued integration approaches.	105
5.3	Reachable set enclosures for the thermodynamic variables $\hat{P}_{ad}, \hat{P}_d, \hat{P}_p, \hat{U}_f$, and $\hat{U}_p$ for problem (5.5.1) subject to $\hat{\mathbf{x}}_0 \in [0.9, 1.1] \hat{\mathbf{x}}_{ref}$ using: inner approximation (INNER); continuous set-valued integration (DINEQ); and discretized set-valued integration with recursive block decomposition (DISCR/RECUR). 4th-order Chebyshev model-based estimators are used in both DINEQ and DISCR/RECUR; a step size of $0.005\hat{T}_{ref}$ is used in DISCR/RECUR.	113
5.4	Reachable sets for the thermodynamic variables $\hat{P}_{ad}, \hat{P}_d, \hat{P}_p, \hat{U}_f$, and $\hat{U}_p$ for problem (5.5.1) subject to $\hat{\mathbf{x}}_0 \in [0.9, 1.1] \hat{\mathbf{x}}_{ref}$	114
5.5	Reachable sets for the thermodynamic variables $\hat{P}_{ad}, \hat{P}_d, \hat{P}_p, \hat{U}_f$, and $\hat{U}_p$ for problem (5.5.1) subject to $\hat{\mathbf{x}}_0 \in [0.5, 2] \hat{\mathbf{x}}_{ref}$	115
6.1	An illustration of the reachable set of a 2-dimensional system.	118

6.2	Linear automaton representation of the nonsmooth dynamic system 6.3.2 with saturations.	121
6.3	DC motor and equivalent circuit.	124
6.4	Flowpipe enclosure for $\dot{\theta}$ in (6.4.7) computed as a linear hybrid automaton (LHA) in CORA (release 2018) and as a mixed-integer optimization problem (MIP) in CRONOS, subject to $w \in [-0.1, 0.1]$. The classical 4-th order Runge-Kutta scheme with a step size of 0.02 s is adopted in the MIP approach	127
6.5	CPU time (t_{CPU}) to solve for the upper bound of $\dot{\theta}$ in (6.4.7) at various time instances (t) with the mixed-integer optimization approach, subject to $w \in [-0.1, 0.1]$	129
6.6	Comparing the predicted trajectories of $\hat{P}_{\text{ad}}$ using the original nonlinear NTP-TL model (2.3.7a'–2.3.7f, 2.3.8) (NLIN); the PWLTP-TL model (Equations 2.3.7a'–2.3.7f, 2.3.10) as a linear hybrid automaton (LHA) in CORA (release 2018); and the PWLTP-TL model (2.3.7a'–2.3.7f, 2.3.10) as a mixed-integer optimization problem (MIP) in CRONOS.	131
6.7	The effects of various discretization schemes on reachable set predictions for $\hat{P}_{\text{ad}}$. The linearized NTP-TL model (Equations 2.3.7a'–2.3.7f with 2.3.10) is assumed and the proposed min-max approach is applied.	132
6.8	Reachable sets for $\hat{P}_{\text{ad}}$, $\hat{P}_{\text{d}}$, $\hat{P}_{\text{p}}$, $\hat{U}_{\text{f}}$, and $\hat{U}_{\text{p}}$ in the NIFTE subject to $\hat{\mathbf{x}}_0 \in [0.9, 1.1] \hat{\mathbf{x}}_{\text{ref}}$ using: (1) the NTP-TL model with inner-approximation (INNER); (2) the NTP-TL model with set-propagation methods (DINEQ); and (3) the PWLTP-TL model with the mixed-integer optimization approach (MIP).	137
6.9	CPU time (t_{CPU}) to solve for the upper bound of $\hat{P}_{\text{ad}}$ in the PWLTP-TL model (2.3.7a'–2.3.7f, 2.3.10) at various time instances $\hat{t}$ with the mixed-integer optimization approach, subject to $\hat{\mathbf{x}}_0 \in [0.9, 1.1] \hat{\mathbf{x}}_{\text{ref}}$	138
6.10	Flowpipe enclosure for $\hat{P}_{\text{ad}}$ using the PWLTP-TL model (2.3.7a'–2.3.7f, 2.3.10) computed with the toolbox CORA (release 2018), subject to $\hat{\mathbf{x}}_0 \in [0.9, 1.1] \hat{\mathbf{x}}_{\text{ref}}$	139
6.11	Reachable sets for $\hat{P}_{\text{ad}}$, $\hat{P}_{\text{d}}$, $\hat{P}_{\text{p}}$, $\hat{U}_{\text{f}}$, and $\hat{U}_{\text{p}}$ in the NIFTE subject to $\hat{\mathbf{x}}_0 \in [0.5, 2] \hat{\mathbf{x}}_{\text{ref}}$ using: (1) the NTP-TL model with inner-approximation (INNER); and (2) the PWLTP-TL model with the mixed-integer optimization approach (MIP).	140

6.12 Flowpipe enclosures for $\hat{P}_{\text{ad}}$ using the PWLTP-TL model (2.3.7a'–2.3.7f, 2.3.10) computed with the toolbox CORA (release 2018), subject to $\hat{\mathbf{x}}_0 \in [0.5, 2] \hat{\mathbf{x}}_{\text{ref}}$	141
--	-----

List of Tables

2.1	Nominal parameter values in NTP and NTP-TL models of the NIFTE.	24
2.2	The key performance indicators of the NIFTE device predicted using the NTP, the NTP-TL and the PWLTP-TL models.	29
3.1	Asymptotic stability and performance of a NIFTE with nominal parameter values (Table 2.1), except $L_f = 4.74 \times 10^5 \text{ kg m}^{-4} \text{ s}^{-1}$	36
4.1	Code list and the natural interval extension of $f(x) = \frac{x-x^2}{\exp(x)}$ with $x \in [0, 2]$. . .	55
4.2	Code list and dependent variable enclosures of $f(x) = \frac{x-x^2}{\exp(x)}$ with $x \in [0, 2]$. Interval analysis is used to generate the natural interval extension, $[z_k]$; and 4th-order Chebyshev model is used to generate the remainder term, $\mathcal{R}_{z_k}(x)$. . .	57
5.1	The general Butcher's tableau for explicit Runge-Kutta methods	89
5.2	The Butcher's tableau for: (a) the forward Euler method; (b) the backward Euler method; and (c) the classical 4th-order Runge-Kutta method	90
5.3	Performance of various techniques towards enclosing solution sets to (5.4.1) at $t = 1$ subject to <i>a priori</i> state bounds $\mathbf{X}^0$	107
5.4	Width of remainder bounds computed for nine thermodynamic state variables in problem 5.5.1 at various time instances subject to $\hat{\mathbf{x}}_0 \in [0.9, 1.1] \hat{\mathbf{x}}_{\text{ref}}$	109
5.5	Performance of various techniques towards enclosing solution sets to problem (5.5.1) at $\hat{t} = 0.2\hat{T}_{\text{ref}}$	111
6.1	Parameter values in the model for the DC motor.	125
6.2	Performance of using a linear hybrid automaton approach (LHA) in CORA (release 2018) and the mixed-integer optimization (MIP) approach in CRONOS towards enclosing solution sets to (6.4.7) at $t = 1 \text{ s}$	128
6.3	Bound and CPU time comparison in the uncertainty scenario $\hat{\mathbf{x}}_0 \in [0.9, 1.1] \hat{\mathbf{x}}_{\text{ref}}$.	133
D.1	License numbers for figures that are reproduced and protected by copyright . .	174

List of Algorithms

1	Multi-objective optimization procedure for NIFTE design.	35
2	Interval-Based Gauss-Seidel Method	69
3	Polynomial Model-Based Gauss-Seidel Iterative Algorithm	71
4	Interval-Based Gauss-Seidel Method with Block-Diagonal Preconditioning . .	92
5	Interval-Based Gauss-Seidel Method with Recursive Block Preconditioning . .	97
6	Polynomial Model-Based Gauss-Seidel Method with Recursive Block Preconditioning	100

Listings

4.1	Example code for generating interval arithmetic-based bounds using MC++ . . .	58
4.2	Program output for Listing 4.1 in MC++	59
4.3	Example code for generating polynomial model-based bounds using MC++ . . .	60
4.4	Program output for Listing 4.3 in MC++	61
4.5	Example code for generating polyhedral relaxations for factorable programs using MC++	64
4.6	Program output for Listing 4.5 in MC++	65
4.7	Example code for computing polynomial model-based bounds for algebraic equations using CRONOS	75
4.8	Program output for Listing 4.7 in CRONOS	76
4.9	Example code for generating polynomial model-based bounds for ODE-IVPs using set-propagation methods implemented in CRONOS	81
4.10	Program output for Listing 4.9 in CRONOS	83
A.1	Script for the Models entity PWLTP_TL in gPROMS ModelBuilder for the NIFTE PWLTP-TL model	163
A.2	Script for the Processes entity PWLTP_TL_proc in gPROMS ModelBuilder for the NIFTE PWLTP-TL model	165
B.1	Implementation for computing polynomial model-based enclosures for (5.4.1) using the header file odebend_expand.hpp in CRONOS	167
C.1	Script for reachability analysis of the PI-controlled DC motor using the mixed- integer optimization approach implemented in CRONOS	170

Nomenclature for the NIFTE models

Symbols

A	cross-section area (m^2)
C	capacitance ($\text{m}^4 \text{s}^2 \text{kg}^{-1}$)
d	diameter (m)
g	gravitational acceleration ($\text{m}^2 \text{s}^{-1}$)
h	heat transfer coefficient ($\text{W m}^{-2} \text{K}^{-1}$)
K	gain (–)
L	inductance (kg m^{-4})
l	length (m)
P	pressure (Pa)
$\mathbf{p}$	parameter vector
R	resistance ($\text{kg m}^{-4} \text{s}^{-1}$)
$\dot{S}$	rate of change of entropy (W K^{-1})
s	specific entropy ($\text{J kg}^{-1} \text{K}^{-1}$) complex frequency (rad s^{-1})
t	time (s)
T	temperature (K) cyclic steady-state period (s)
U	volumetric flowrate ($\text{m}^{-3} \text{s}^{-1}$)
V	volume (m^3)
$\dot{W}$	work power (W)
$\dot{X}$	thermal exergy power (W)
$\mathbf{x}$	state vector
y	position of vapor-liquid interface (m)
z	displacement of vapor-liquid interface from the equilibrium position (m)
Z	impedance ($\text{kg m}^{-4} \text{s}^{-1}$)
α	thermal diffusivity ($\text{m}^2 \text{s}^{-1}$)
Δ	difference
η	thermodynamic efficiency (–)

γ	adiabatic index (–)
κ	thermal conductivity ($\text{W m}^{-1} \text{K}^{-1}$)
λ	eigenvalue
μ	dynamic viscosity ($\text{kg m}^{-1} \text{s}^{-1}$)
ρ	density (kg m^{-3})
Σ	surface area (m^2)
τ	reference time (s)

Subscripts and superscripts

$\wedge$	dimensionless variable
$-$	time-averaged value
L	lower bound
U	upper bound
$\circ$	reference or equilibrium value
0	initial value
ad	adiabatic vapor chamber
d	displacer cylinder
ex	exergetic
f	feedback connection
gl	glass
l	load
p	power cylinder
pf	pumped fluid
ref	reference position
sat	fluid in saturation region
th	thermal domain
tl	thermal loss
vap	vapor
wf	working fluid

Nomenclature for the PI-controlled DC motor

<i>Symbols</i>			
b	viscous resistance (Nm s rad^{-1})	q	integral of error (rad)
i	current (A)	v	voltage of the source (V)
J	total moment of inertia (kg m^2)	w	disturbance in the torque (Nm)
K	back-EMF constant (V s rad^{-1})	$\dot{\theta}$	rotational speed (rad s^{-1})
K_P	proportional gain (–)	<i>Subscripts and superscripts</i>	
K_I	integral gain (–)	min	minimum
L	inductance (H)	max	maximum
R	resistance (Ω)	ref	reference

Abbreviations and notations

<i>Abbreviations</i>			
AE	algebraic equation	NLP	nonlinear program
CHX	cold heat-exchanger	NTP	nonlinear temperature profile
CSS	cyclic steady states	ORC	organic Rankine cycle
CTD	constant temperature difference	ODE	ordinary differential equation
DAG	directed acyclic graph	PWLTP	piecewise-affine linear temperature profile
DHX	dynamic heat exchange	SBB	spatial branch-and-bound
DO	dynamic optimization	STN	state-transition network
GDO	global dynamic optimization	TL	thermal loss
HA	hybrid automaton	<i>Notations</i>	
HHX	hot heat-exchanger	$\mathbb{I}$	natural interval extension
ISA	interval superposition arithmetics	$\mathbb{II}$	polynomial model extension
IVP	initial value problem	$\mathbb{IR}$	interval real space
LTP	linear temperature profile	$\mathcal{P}$	polynomial part of a polynomial model
MIP	mixed-integer program	$\mathcal{R}$	remainder part of a polynomial model
NIFTE	non-inertive-feedback thermofluidic engine		

Chapter 1

Introduction

High-temperature heat engines are ubiquitous in the power generation industry, stationary distributed power (co)generation, the transport and many other sectors. This demand for high-grade heat has led to ever increasing rates of fossil-fuel consumption and resource depletion, while at the same time concerns exist in relation to the emissions associated with the combustion of these fuels. There is therefore a pressing need for exploiting alternative heat sources.

Broadly speaking, heat available at a relatively low temperature (typically under 250 °C) and for which there is no direct use is classified as “low-grade”. Such heat is abundant in the process and chemical industries, where it is often “wasted” by release into the environment without further utilization. Typically, in an industrial process 20-50% of its energy is rejected as waste heat, and exhaust gas and cooling water carry most of them [1, 2]. Studies show that more than 60% of this waste heat is discharged at temperature below 200 °C, thus falling in the category of low-grade heat. Large amounts are also generated and available in solar collectors and geothermal wells, among other applications. Unfortunately, all technologies for low-grade heat conversion to power are associated with low thermal efficiencies [3, 4], a restriction imposed by the second law of thermodynamics [5].

The efficiency of an ideal heat engine, which operates on the Carnot cycle, is determined by the temperatures of its heat source and heat sink as follows,

$$\eta_{\text{Carnot}} = 1 - \frac{T_C}{T_H},$$

where η_{Carnot} is the Carnot efficiency of the engine, and T_C and T_H are the temperatures of the heat sink and the heat source respectively. It is clear that the greater the temperature difference, the higher the Carnot efficiency; whereas a relatively small temperature difference is usually found in low-grade heat recovery processes. Furthermore, the Carnot cycle is impossible to achieve in practice, hence leading to inherently low efficiencies associated with such processes.

Most mechanical heat engines operate instead on a Rankine cycle, which consists of a pump, an evaporator, a turbine and a condenser. A schematic representation of a typical Rankine cycle is shown in Figure 1.1a, and the corresponding temperature-entropy phase diagram is depicted in Figure 1.1b.

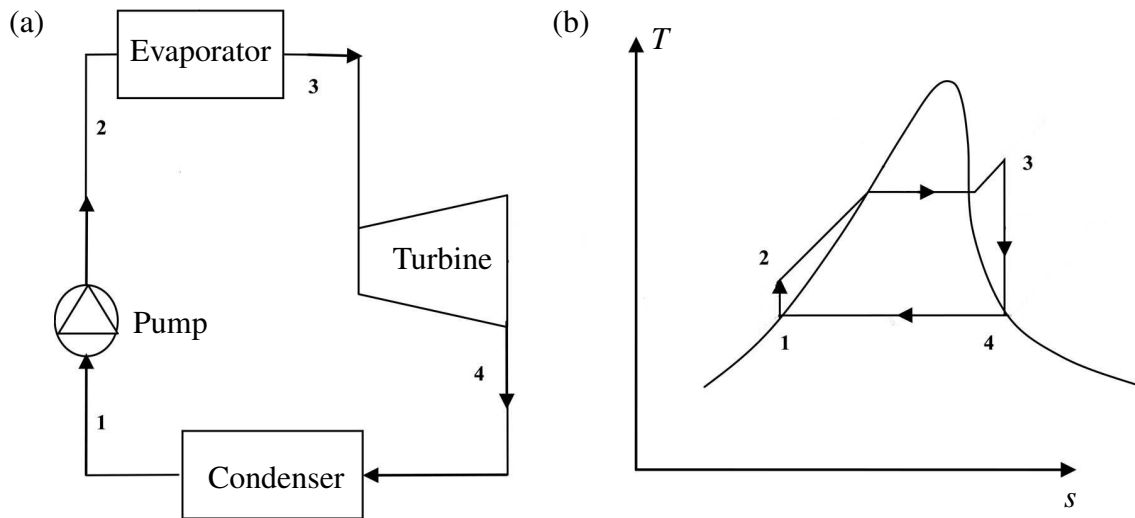


Figure 1.1: (a): Schematic illustration of a typical Rankine cycle; (b): Temperature (T) - entropy (s) diagram of a typical Rankine cycle.

The key stages of the Rankine cycle as shown in Figure 1.1 are:

- 1 $\rightarrow$ 2 isentropic compression;
- 2 $\rightarrow$ 3 isobaric heat addition;
- 3 $\rightarrow$ 4 isentropic expansion;
- 4 $\rightarrow$ 1 isobaric heat removal.

The thermal efficiency of a Rankine cycle is defined as the fraction of total heat input that is converted to useful work output. This efficiency is lower than that of a Carnot cycle with the same heat source and heat sink of the same temperatures respectively. To evaluate the performance of a heat engine, in particular for low-grade heat recovery, the concept of exergetic efficiency η_{ex} is introduced,

$$\eta_{\text{ex}} = \frac{\eta_{\text{th,Rankine}}}{\eta_{\text{Carnot}}}.$$

which measures the relative ratio between the thermal efficiencies of the actual Rankine cycle $\eta_{\text{th,Rankine}}$, and that of the ideal Carnot cycle η_{Carnot} .

1.1 Thermofluidic engines and other Heat recovery technologies

Thermofluidic oscillators are a class of Rankine-cycle engines for converting heat into useful work via the creation of oscillatory fluid motion. The non-inertive-feedback thermofluidic engine (NIFTE) is a particular type of two-phase thermofluidic oscillator [6, 7, 8]. It has been established experimentally on early prototypes [9] that a NIFTE is capable of operating with a temperature difference as small as 30 °C between the heat source and sink, and more recent versions of the technology may be able to operate with temperature differences even smaller than this. An early prototype was reported as achieving thermal efficiencies up to 1% and exergetic efficiencies—the thermal efficiency as a fraction of the Carnot efficiency—between 1–2%, when operated with a heat source at 65–90 °C, a heat sink at 4–12 °C, and n-pentane as the working fluid [8, 10].

Such efficiencies can be considered low in comparison to those of other heat conversion technologies that are typically designed and intended for higher-grade heat sources, such as organic Rankine cycle (ORC) systems, Stirling and thermoacoustic engines [5]. In general, thermal efficiencies in the approximate range of 10–20% are expected from ORC systems, also depending on the system size, the lower end achievable at heat source temperatures of about 100–150 °C and the upper end at up to 400–500 °C [11]; while Stirling and thermoacoustic engines have much lower efficiencies at those temperatures, but can reach thermal efficiencies between 20–30% at heat-source temperatures of 600–800 °C [5]. In one

study [12], a small-scale regenerative ORC power system comprising a scroll expander and with R123 as the working fluid was reported as having a thermal efficiency upwards of 7% at a heat-source temperature of 165–183 °C and an ambient air temperature of 22.5 °C cooling the condenser. In a traveling-wave thermoacoustic engine with air as the working gas [13], a thermal efficiency of 14.5% was reported under 25 bar, with a heat-source temperature of 650 °C and a cooling temperature of 12 °C. Importantly, it can be seen that these conventional power generation systems are neither technically suitable nor economically feasible when the heat-source temperature drops below 100 °C. Furthermore, the construction and operation of these systems are invariably more complicated than those for a thermofluidic oscillator, which has no or few moving parts, mechanical components or dynamic seals.

An alternative type of two-phase thermofluidic oscillator, known as the Up-THERM engine, has also been recently investigated for heat conversion applications [14, 15]. A lab-scale Up-THERM engine had a predicted exergetic efficiency of about 0.8%, when operating with a heat source and sink at 450 °C and 10 °C, respectively, and *n*-pentane as the working fluid [16]. It is also worth noting that an optimized pilot-scale Up-THERM engine with the same working fluid was predicted to yield promising exergetic efficiencies between 10–40% for a heat source temperature in the range 200–400 °C [15], which is close to the efficiency of its ORC counterparts and could even be higher with optimally selected working fluids [17]. The readers are directed to Reference [18] for a detailed review of small-scale thermal-mechanical systems for heat recoveries.

1.2 Modeling and optimization of thermofluidic engines

Given their attractive features of being cheap and simple to construct and maintain, thermofluidic engines, in particular the NIFTE, remain a topic of interest for many studies. Most works on thermofluidic engines, especially the earlier ones, [19, 20, 8, 9, 21, 10, 22, 23] focused on constructing linear dynamic models, and investigations were conducted by analysing the corresponding transfer functions. However, such linear models were insufficient in describing the key oscillatory features and limit cycles exhibited by thermofluidic engines, which are indeed a characteristic of nonlinear systems only. In response to this, nonlinear models were developed for the NIFTE engine [24]; and likewise for the UP-THERM engine

in recent years [15, 16].

A number of studies have been performed to optimize the performance of such engines [9, 21, 24, 22, 10, 15, 16]. However, a majority of these studies have used parametric sensitivity analysis, which may only identify locally optimal solutions. However, only local solutions can be found in this way. A more systematic approach is to adopt an optimization-based approach, which calls for the solution of challenging dynamic optimization problems.

1.3 Dynamic optimization technology

A dynamic optimization problem consists of an objective function, a dynamic model of the physical system, and a set of constraints. The general formulation is presented as follows,

$$\begin{aligned}
 & \min_{\mathbf{u}(t), \mathbf{p}} : \Phi(\mathbf{x}(t_f), \mathbf{u}(t_f), \mathbf{p}) \\
 & \text{s.t. : } \dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t), \mathbf{p}), \\
 & \quad \psi_j(t, \mathbf{x}(t), \mathbf{p}) \stackrel{=}{\leq} 0, \quad j = 1, \dots, n_\psi, \\
 & \quad \mathbf{x}(t_0) = \mathbf{h}(\mathbf{p}), \\
 & \quad \mathbf{x}(t) \in [\mathbf{x}^L, \mathbf{x}^U], \quad \mathbf{u} \in [\mathbf{u}^L, \mathbf{u}^U], \quad \mathbf{p} \in [\mathbf{p}^L, \mathbf{p}^U], \quad \forall t \in [t_0, t_f]
 \end{aligned}$$

where $\mathbf{x}$ represents the state variables, $\mathbf{u}$ for the control variables, and $\mathbf{p}$ for the time-invariant system parameters. $\mathbf{x}^L$ and $\mathbf{x}^U$ denote the lower and the upper bounds of $\mathbf{x}$ respectively; similarly, $\mathbf{u}^L$ and $\mathbf{u}^U$ for $\mathbf{u}$; and $\mathbf{p}^L$ and $\mathbf{p}^U$ for $\mathbf{p}$. The objective function $\Phi(\mathbf{x}(t_f), \mathbf{u}(t_f), \mathbf{p})$ is to be minimized at t_f subject to a dynamical model described by the ordinary differential equations (ODEs) associated with $\mathbf{x}$, and a set of equality and inequality constraints represented by ψ .

Both *direct* and *indirect methods* may be used to compute a numerical solution to such dynamic optimization (DO) problems. The former class of methods parameterizes the control profiles to approximate the problems as finite-dimensional optimization. Powerful nonlinear programming (NLP) techniques can be applied afterwards to solve the transcribed problems. In the indirect methods, no *a priori* discretization is performed and one seeks a solution to the system of the necessary conditions for optimality. Popular indirect methods are based on the

Pontryagin Maximum Principle [25] and the Hamilton-Jacobi-Bellman equation [26]. Indirect methods solve the problems in their original infinite dimensional setting [27, 28]. But their application to constrained problems can be different due to the need to identify the optimal set of active constraints [29].

There are furthermore two major classes of direct methods for dynamic optimization, which entail very different transcription approaches. In the direct sequential approach [30], parameterization of the control trajectories is employed to transcribe the DO problem to a finite-dimensional NLP with ODEs embedded. The ODEs and their sensitivities or adjoints are then solved by numerical integrators, which provide accurate state values but many require a large computational effort in large-scale dynamic systems. In the direct simultaneous approach [31, 29], the DO problem undergoes full discretization and is transcribed to an NLP without ODEs embedded. Both the control and state trajectories are parameterized. However, this full discretization results in a much larger NLP problem than the direct sequential methods, the solution of which calls for exploiting special structures in the problem.

Most dynamic optimization implementations rely on gradient-based NLP methods, which offer no guarantee of convergence to a global optimum for nonlinear dynamic systems. This is especially problematic in the optimization of periodic systems such as thermofluidic engines, which can present multiple cyclic steady states and therefore multiple local optima.

Stochastic optimization offers a wide class of NLP methods that are derivative-free, and has become increasingly popular in many practical applications. The most famous algorithms include the Nelder-Mead simplex method [32], simulated annealing [33], genetic algorithm [34] and Bayesian optimization [35]. They are particularly useful when the problem is highly nonlinear and the initial guesses are poor. While gradient-based methods would result in convergence to local optimum only, stochastic methods allow for escape from local optimum because of the use of multiple random starting points in the search space. However, it should be noted that stochastic methods provide no means to guarantee the finding of global optimal solution.

The search for global optimum relies on the use of complete search methods. The complete search methods developed for global dynamic optimization (GDO) are not as developed as their NLP optimization counterparts. Most available methods follow the spatial branch-and-

bound (SBB) framework [36]. As the name suggest, *branching* first takes place to partition the problem recursively into subproblems, called *nodes*, The role of the *bounding* step is to obtain upper and lower bounds of each node. The nodes giving rise to infeasible or inferior solutions are fathomed. The remaining nodes are then further partitioned and bounded, thus forming a search tree converging to the optimal value.

Both the direct sequential and simultaneous methods may be contrived with the SBB framework. In the sequential approach on which most research has focused, ODE bounding techniques are needed to enclose of the reachable sets of the parametric ODEs. The α BB method which was originally developed for standard algebraic NLPs has been extended to dynamic embedded NLPs [37, 38, 38, 39], so that convex underestimators or concave overestimators are created. However, these methods are computationally demanding in general due to the bounding of first- and second-order sensitivity or adjoint variables. The state-of-the-art is the family of set-propagation methods, whose key idea is to construct an auxiliary system of ODEs whose solution encloses the reachable set of original ODEs. The theory of differential inequalities [27, 40], viability theory [41], or other set-theoretic approaches including ellipsoidal calculus [42] are used to generate the auxiliary system of ODEs. More recently, a unified framework was proposed for enclosing the reachable set of nonlinear ODEs, known as the theory of generalized differential inequality [43]. Effective ODE bounding methods tailored to GDO algorithms offer significant research opportunities in order to reduce the computational burden.

Considerably less research has been devoted to solving GDO problems using the direct simultaneous approach due to the large size of the NLP arising from full discretization. Special algorithms are need to exploit the special structure of these NLP problems. For example, a reduced space NLP interior point algorithm was developed based on reduced-Hessian successive quadratic programming and the interior point method [44]. This algorithm was effective in locating optimal solutions for large-scale NLPs, but these solutions are not guaranteed to be global. On the other hand, multiple complete search methods have been extended to GDO by approximating the state trajectories using a variety of polynomial basis functions with different orthogonal collocation schemes [45, 46, 47, 48].

Despite these advances in GDO, the capability of the state-of-the-art solvers is limited to solving small decision problems (typically 5-10 variables) with simple dynamics [49, 50, 51].

More effort is thus needed to develop effective specialized solvers that can tackle a significantly broader class of optimal control problems with practical relevance to global optimality.

1.4 Main objectives and thesis outline

The current state-of-the-art in thermofluidic engines for low-grade heat recovery and in global dynamic optimization motivate the following two main objectives of this thesis.

1. The first aim is to enhance modeling of the NIFTE. The available NIFTE models, be they linear or nonlinear, can only describe the NIFTE behavior qualitatively. For instance, the current nonlinear NIFTE model fails to account for irreversible thermal losses, resulting in inaccurate quantitative performance estimation.
2. The other aim is to develop more efficient and effective global dynamic optimization techniques. Thermofluidic engines are highly complex systems with multiple limit cycles, for which finding the global solutions is a challenging task.

To accomplish these two goals, multiple directions and topics have been explored and the main findings are presented in this thesis. The structure of this thesis is described as follows.

In Chapter 2, the principles and existing models of the NIFTE are first reviewed, and their limitations are analyzed. A new nonlinear model with thermal losses is then presented, as well as a switched-linear counterpart. The performance predictions from the existing and the newly developed models are elaborated and compared for the purpose of model validation.

Equipped with the improved models, the periodic behavior of the NIFTE by computing their limit cycles is then characterized. A systematic approach based on dynamic optimization is proposed instead of the simulation-based approach adopted in current studies of thermofluidic engines. Furthermore, a parametric sensitivity analysis is conducted to identify the most sensitive parameters and use these results toward a multiobjective optimization of the NIFTE performance. The results of this analysis are presented in Chapter 3.

The use of gradient-based methods to analyze and optimize the NIFTE is unreliable due to nonconvexity, thereby calling for efficient global dynamic optimization methods. One focus in

the second part of the thesis is on enhancing bounding techniques for parametric ODEs, which form the basis of direct methods in GDO.

In Chapter 4, the focus is on reviewing methods that lay the foundation of GDO. These include bounding techniques for both algebraic and differential equations, which form essential steps of solving dynamic optimization problems. Furthermore, existing implementations of these techniques in the in-house C++ toolkits MC++ and CRONOS are briefly demonstrated in this chapter.

Most efforts towards reachability analysis of ODEs follow an set-propagation approach. In Chapter 5, a discretization approach is adopted, and this is a much less explored direction. The sparse structure of the resulting algebraic system allows the application of block decomposition. Two different decomposition strategies are investigated; and the bounding techniques for algebraic equation systems are adapted correspondingly. The first is a naïve strategy which leads to diagonal decomposition. A more sophisticated recursive strategy is later proposed. In the last part of this chapter, the impact of different decomposition strategies on the required CPU time and the resulting bounds tightness are reported and compared for a small-scale case study, as well as the larger-scale NIFTE system.

The bounding techniques for dynamical systems in Chapter 5 are developed based on continuous nonlinear models. The switched-linear reformulation of the NIFTE model, that is developed in Chapter 2, transforms the goal to reachability analysis of linear hybrid dynamical systems. In Chapter 6, an optimization-based approach that relies on solving a number of mixed integer linear problems is proposed and elaborated. The results using the proposed approach are then compared against those using the conventional hybrid automata approach and using the set-propagation ODE bounding approach. Moreover, the time complexity of the proposed approach is also studied.

Finally, Chapter 7 contains the summary of main contributions of this thesis, and discussions on how the aforementioned objectives are met. Some potential future directions in both fields of thermofluidic engines and dynamic optimization technology are also discussed.

Chapter 2

NIFTE Principles and Modeling

The thermofluidic engine of interest in this thesis is the so-called non-inertive-feedback thermofluidic engine (NIFTE). This chapter presents a brief introduction and an overview of the state-of-the-art modeling efforts towards the NIFTE. In Section 2.1, the engine's configuration and principles are first described in details. Existing NIFTE models are then reviewed in Section 2.2, and their respective advantages and limitations are discussed. Building on the efforts of the previous models, a new nonlinear dynamic model is proposed in Section 2.3.3; and a switched-linear counterpart in Section 2.3.4. Dynamic simulation results using the newly developed models are presented in Section 2.4. Part of these results were published in Reference [52].

2.1 NIFTE Principles

The description of the configuration and step-by-step operation of the NIFTE are given as part of Figure 2.1. The working fluid exists in both the liquid and vapor phases, and it changes phase periodically during the operation (evaporation and condensation). It can be tailored to suit the available external heat source and sink temperatures. A NIFTE is simple to construct and has few moving parts in comparison with conventional engines, making it both a reliable and low capital-cost technology, that is capable of converting low-grade heat into mechanical work.

It has been established experimentally on early prototypes [9] that a NIFTE is capable of

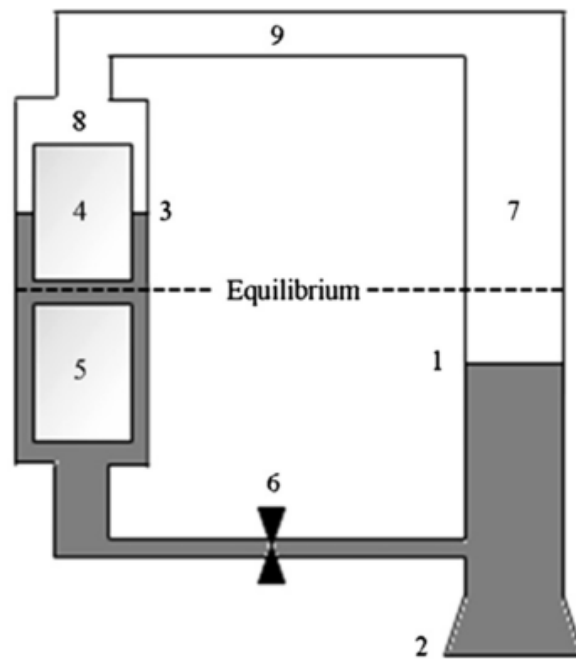


Figure 2.1: NIFTE fluid pump schematic, adapted from Reference [24].

Configuration: A NIFTE consists of four major elements: the displacer cylinder ⑧ that contains the hot ④ and cold ⑤ heat-exchangers; the adiabatic vapor region ⑨; the power cylinder ⑦ that is connected to the load ②; and the feedback connection tube ⑥. The white area denotes the vapor phase, while the grey area denotes the liquid phase of the working fluid.

Operational cycle: Low-grade heat from an external source is supplied to the system via the hot heat-exchanger (HHX) ④ to vaporize the working fluid. The pressure in the adiabatic vapor region ⑨ thus increases, causing the liquid level ① in the power cylinder ⑦ to fall below the equilibrium level. The liquid level ③ in the displacer cylinder ⑧ drops as a result of the hydrostatic pressure via the feedback connection ⑥. The vapor comes into contact with the cold heat-exchanger (CHX) ⑤, where heat is removed from the system, causing the vapor to condense and the liquid height to rise back above the equilibrium position. Sustained oscillations in the load ② are thereby created by the cyclic condensation and evaporation of the working fluid in the cylinders.

operating with a temperature difference as small as 30 °C between the heat source and sink, and more recent versions of the technology may be able to operate with temperature differences even smaller than this. An early prototype was reported as achieving thermal efficiencies up to 1% and exergetic efficiencies—the thermal efficiency as a fraction of the Carnot efficiency—between 1–2%, when operated with a heat source at 65–90 °C, a heat sink at 4–12 °C, and n-pentane as the working fluid [8, 10].

2.2 Review of existing NIFTE models

Nearly all of the mathematical models proposed so far to analyze the dynamical behavior of the NIFTE feature linear dynamics. Smith [6, 7, 8] first developed a linear model, based upon earlier work on thermoacoustic engines [20] and electrical analogies for thermal and fluid processes [19, 53]. In this early work, the NIFTE components were described by linear equations in analogy with electrical components—namely, resistors to account for fluid drag and thermal resistance, and capacitors to represent hydrostatic pressure and vapor compressibility—and the heat-exchange rate between the hot and cold elements and the working fluid was assumed to be either linear or constant. A complete engine model then connects these electrical components into a circuit according to the actual thermal-fluid technology layout. Inductors were later introduced in the electrical analogy to account for fluid inertia [9, 21], which allowed improvements to the accuracy of the predictions, and an effort to consider performance over wide parameter ranges was also made [23].

The main difference among these linear models lie in their assumptions for the heat transfer process within the displacer cylinder. Figures 2.2a, 2.2b and 2.2c show the three variations, i.e. the linear temperature profile (LTP) [19, 20, 9, 21, 10, 23], the constant temperature difference (CTD) [22], and the dynamic heat exchange (DHX) [22, 10]. Note that in Figure 2.2, z represents the height of the working fluid relative to the equilibrium position in the displacer cylinder; and T_{hx} denotes the temperature at the surface of the heat exchanger. The equilibrium position is taken to be the midpoint between the hot heat exchanger (HHX) and the cold heat exchanger (CHX). The HHX and CHX sides are marked accordingly in the temperature and heat transfer profiles in Figure 2.2. In the LTP model, T_{hx} is assumed to vary proportionately to the relative distance from the hot and cold elements. Constant temperatures, on the other hand, are imposed in regions near the HHX or the CHX in the CTD model. Despite being very easy to implement, these two models are too simplistic; the LTP version fails to account for the fact that T_{hx} reaches a maximum or a minimum near the hot or the cold heat exchangers, and the CTD version assumes an infinitely steep transition in T_{hx} . A more accurate model was proposed by Solanki et al. [22], which allows the heat exchangers to store and release dynamically upon interaction with the working fluid. This is illustrated in Figure 2.2c, where the power input to the heat exchangers $\dot{Q}$ is assumed to vary linearly with

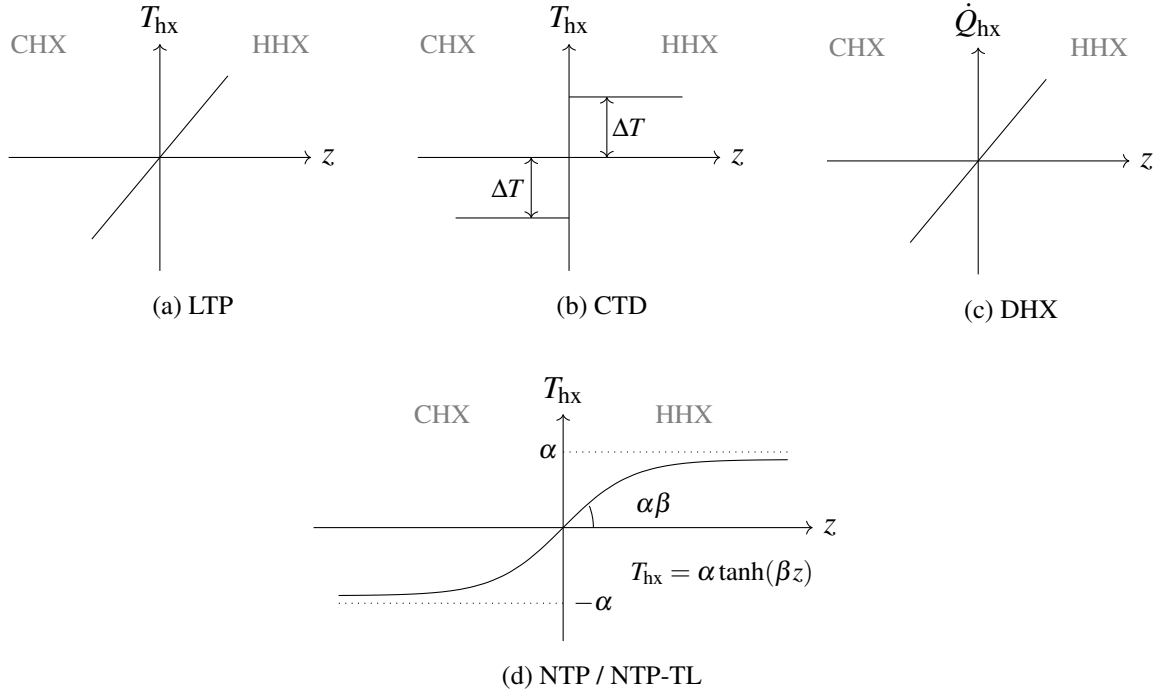


Figure 2.2: The temperature or power profile along the hot heat exchanger (HHX) and cold heat exchanger (CHX) assumed in the (a) LTP, (b) CTD, (c) NTP / NTP-TL models for the NIFTE device [22, 24]. z is the displacement of the vapor-liquid interface from the equilibrium position, that is taken to be the midpoint between HHX and CHX. T_{hx} is the temperature deviation around the reference temperature at $z = 0$.

z . The effects of phase changes are considered in this model; heat is deposited onto the heat exchanger units during condensation, and removed from those units during evaporation of the working fluid.

Notwithstanding their ability to provide quantitative predictions of various performance indicators of the NIFTE performance via numerical simulation at marginal stability, these linear models cannot describe the sustained and robust periodic oscillations of the working fluid as observed in practice, which in essence is a nonlinear phenomenon. Subject to a steady temperature difference between the two heat-exchangers, the engine reaches a cyclic steady state (CSS) after a sufficiently long settling time, where the amplitude and frequency of the oscillations remain constant. When variations in the heat-exchanger temperature difference or other external disturbances occur, which is inevitable in practice, the engine will eventually again return to a CSS after some time. This is the signature of an asymptotically stable limit

cycle in the dynamics, which may only occur in nonlinear dynamic systems.

Inspired by earlier work on exploiting the nonlinearity of core physical elements to predict the efficiency and robustness of thermodynamic engines [54, 55, 56, 57], Markides and co-workers [24] were the first to propose a nonlinear inertive model of the NIFTE, referred to as the NTP model. The key nonlinearity in this model corresponds to the physical saturation in heat transfer rate between the hot and cold elements and the working fluid, whereas the rest of the engine components are described using the same linear electrical analogy as in earlier work. The assumed temperature profile is depicted in Figure 2.2d, and the relation between T_{hx} and z is given as follows,

$$T_{hx} = \alpha \tanh(\beta z), \quad (2.2.1)$$

where α and $-\alpha$ are the maximum and minimum temperature deviations around the reference temperature at $z = 0$ respectively; and $\alpha\beta$ is the gradient at $z = 0$.

Failure to account for various irreversible thermal losses in the early NIFTE models also leads to an overestimation of the engine's efficiency and work output by an order of magnitude or more. Smith [8] identified the two predominant exergy losses to be: (i) shuttle losses, associated with the irreversible two-way heat transfer between the working fluid and the cylinder walls; and (ii) entrance and retrograde condensation losses, caused by the alternating parasitic condensation and evaporation of the working fluid. Solanki and Markides [10, 58] were the first to account explicitly for these losses in a linear NIFTE model, which led to much more reliable efficiency predictions. However, a similar extension of the NTP model to encompass exergy losses has not been reported in the literature as of yet.

2.3 Mathematical derivation of the NIFTE models

The two main thermal-fluid transport processes that govern the operation of a NIFTE device are: (i) pressure differences, driving working-fluid flows through fluid components; and (ii) temperature differences, responsible for heat flows in thermal components. This section presents a summary of the models developed to describe these processes in the fluid and thermal domains first, followed by a complete statement of both the NTP and NTP-TL device models, and a description of the performance indicators considered for the assessment.

2.3.1 Fluid domain

The main components ②, ⑥, ⑦, ⑧, ⑨ of the engine (cf. labels in Figure 2.1) are modeled based on electrical analogies. The drags experienced by the working fluid in the load ② (due to viscosity) and in the feedback connection ⑥ (due to small diameter and/or a valve) are represented by resistors, R_i . In particular, the model assumes that the fluid flow in these elements is akin to viscous laminar flow in a smooth pipe. The inertive effects experienced by the oscillating liquid flow in the load ②, feedback connection ⑥, power cylinder ⑦, and displacer cylinder ⑧ are modeled by inductors, L_i . The hydrostatic pressures at the bottom of the power cylinder ⑦ and displacer cylinder ⑧, and the adiabatic compression/expansion in the vapor region ⑨ are represented by capacitors, C_i . The processes of vapor compression and expansion undergone by the working fluid in the vapor region ⑨ are assumed to be isentropic (i.e. both adiabatic and reversible).

These R, L, C parameters may be expressed in terms of the physical and geometric properties of the NIFTE, namely density, ρ and dynamic viscosity, μ of the working and pumped fluids (indexed with $_{wf}$, $_{pf}$, respectively), and diameter, d , length, l and cross-sectional area, A of the tubes and valves (indexed with $_l$, $_f$, $_p$, $_d$, $_{ad}$ in reference to the components ②, ⑥, ⑦, ⑧, ⑨, respectively) [59]:

$$\begin{aligned} R_l &= \frac{128\mu_{pf}l_l}{\pi d_l^4}, & R_f &= \frac{128\mu_{wf}l_f}{\pi d_f^4}, & L_p &= \frac{\rho_{wf}l_p}{A_p}, & L_d &= \frac{\rho_{wf}l_d}{A_d}, \\ L_l &= \frac{\rho_{pf}l_l}{A_l}, & L_f &= \frac{\rho_{wf}l_f}{A_f}, & C_p &= \frac{A_p}{\rho_{wf}g}, & C_d &= \frac{A_d}{\rho_{wf}g}, \\ C_p &= \frac{A_p}{\rho_{wf}g}, & C_d &= \frac{A_d}{\rho_{wf}g}, & C_{ad} &= \frac{V^\circ}{\gamma P^\circ}, \end{aligned} \quad (2.3.1)$$

with V° and P° , the equilibrium (time-averaged) volume and pressure in the gas phase; γ , the adiabatic index; and g , the gravitational acceleration. The detailed derivations of these R, L, C parameters can be found in Reference [59]. All of the physical properties, and therefore the R, L, C parameters themselves, are assumed to be temperature independent. But for a particular fluid, the R, L, C parameter values can be adjusted by varying the geometric properties (length l , diameter d , and/or cross-sectional area A) of the corresponding elements, which provides the basis for the parametric analysis and optimization in the Results section below.

The electrical analogy assumes linearized first-order governing equation between the volumetric flowrate of the working fluid, U_i through a given component i and the pressure difference, ΔP_i across that component:

$$\Delta P_i = U_i Z_i, \quad (2.3.2)$$

with Z_i the impedance of component i . The impedance for each of the main components of the engine is expressed in terms of resistance, inductance and capacitance in the Laplace domain, using the principle of superposition to describe multiple effects occurring within the same component:

Load ②:	$Z_l = R_l + sL_l$
Feedback connection ⑥:	$Z_f = R_f + sL_f$
Power cylinder ⑦:	$Z_p = sL_p + \frac{1}{sC_p}$
Displacer cylinder ⑧:	$Z_d = sL_d + \frac{1}{sC_d}$
Adiabatic vapour region ⑨:	$Z_{ad} = \frac{1}{sC_{ad}}$

These R, L, C elements, as arranged in a complete NIFTE circuit, may be visualized in Figure 2.3, for which further details will be provided below.

2.3.2 Thermal domain

The thermal domain, which comprises the hot and cold heat-exchangers ④, ⑤ within the displacer cylinder ⑧, converts heat into pressurization and fluid flow. The usual assumptions adopted here include: (i) the spatial temperature profile along the vertical walls of the heat-exchangers is externally imposed as a boundary condition to the device and does not vary with time; and (ii) the convective heat transfer associated with phase change dominates over the forced convection taking place away from an active region near the vapor-liquid interface of the working fluid within which phase-change heat transfer occurs. Under these conditions,

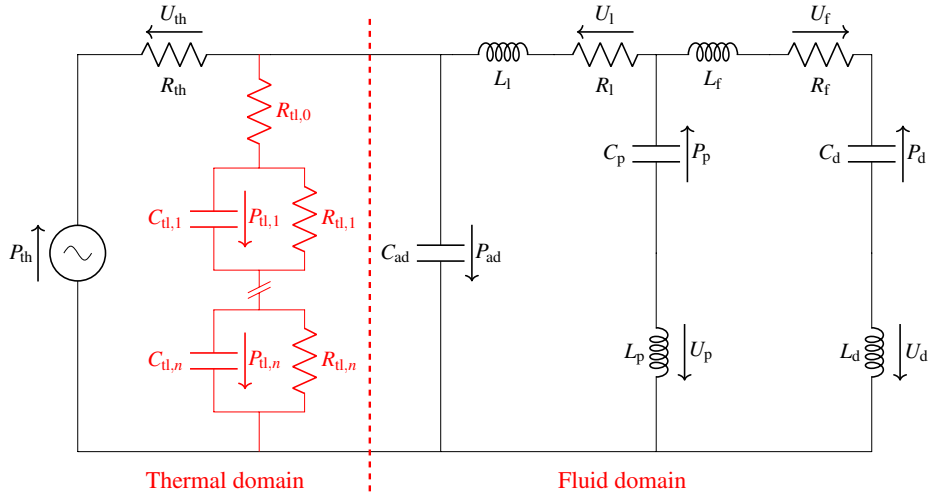


Figure 2.3: Complete NIFTE circuit. R_i denotes a resistance, C_i a capacitance, L_i an inductance, P_i a pressure, and U_i a volumetric flowrate. The subscript _{th} denotes the thermal domain, _{ad} the adiabatic domain, _l the load, _p the power cylinder, _d the displacer (heat-exchanger) cylinder, and _f the combined effect of the feedback connection and/or valve. The components in red relate to the part of the circuit describing irreversible thermal losses, only relevant to the NTP-TL model.

the heat transferred to the working fluid, $\dot{X}_{th}$ is given by:

$$\dot{X}_{th} = T^\circ \dot{S}_{th} = h_{th} A_{th} [T_{th}(y) - T_{ad}] , \quad (2.3.3)$$

with $\dot{S}_{th}$, the associated rate of change of entropy; T° , the equilibrium (time-averaged) temperature; h_{th} , the (constant) phase-change convective heat transfer coefficient; A_{th} , the surface area over which the heat transfer takes place; T_{ad} , the temperature of the working fluid; and T_{th} , the heat-exchanger wall temperature experienced by the working fluid at the vapor-liquid interface ③, y .

The connection between Equation (2.3.3) and the fluid domain proceeds by: (i) relating the entropy rate of change, $\dot{S}_{th}$ to an equivalent volumetric flow-rate of vapor (due to phase change), U_{th} via

$$\dot{S}_{th} = \rho_{vap}^\circ \Delta s_{vap} U_{th} ,$$

with ρ_{vap}° , the equilibrium (time-averaged) density of the vapor, and Δs_{vap} , the specific entropy

of vaporization; and (ii) relating temperatures to pressures as

$$T_{\text{th}} = \left(\frac{dT}{dP} \right)_{\text{sat}} P_{\text{th}}, \quad T_{\text{ad}} = \left(\frac{dT}{dP} \right)_{\text{sat}} P_{\text{ad}},$$

with $\left(\frac{dT}{dP} \right)_{\text{sat}}$, the rate of change in saturation temperature of the working fluid with respect to the saturation pressure. Using the same electrical analogy as in Equation (2.3.2), we obtain:

$$P_{\text{th}} - P_{\text{ad}} = f(P_{\text{d}}) - P_{\text{ad}} = U_{\text{th}} Z_{\text{th}}, \quad (2.3.4)$$

where the thermal impedance Z_{th} consists of the thermal resistance between the working fluid and the walls of the heat-exchangers,

$$\text{Displacer cylinder } \textcircled{8}: \quad Z_{\text{th}} = R_{\text{th}} = \frac{\rho_{\text{vap}}^{\circ} \Delta s_{\text{vap}} T^{\circ}}{h_{\text{th}} A_{\text{th}} \left(\frac{dT}{dP} \right)_{\text{sat}}}$$

The heat-exchanger wall temperature T_{th} , and the pressure P_{th} likewise, vary with the vertical position y of the vapor-liquid interface $\textcircled{3}$, and can be related to the pressure variable $P_{\text{d}} = \rho_{\text{wf}} g y$. In particular, the function f in Equation (2.3.4) may taken as either linear [8, 9] or nonlinear [24] depending on modeling assumptions, as depicted in Figure 2.2.

In addition to the thermal resistance R_{th} , two important thermal loss mechanisms in the displacer cylinder $\textcircled{8}$ are the shuttle loss and the entrance and retrograde condensation loss [8]. The former is associated with irreversible alternating conductive heat transfer between the working fluid and the cylinder walls, via a heat diffusion equation with either an isothermal or adiabatic boundary condition. The latter is caused by the alternating parasitic condensation and evaporation of the working fluid, which may be described as a convective heat transfer. Following the derivations in References [10, 58], a total impedance representing these thermal losses can be solved in the Laplace domain, to arrive at the expressions:

$$\text{Isothermal boundary condition:} \quad Z_{\text{tl}} = \frac{A_{\text{tl}}}{\sqrt{s}} \tanh(B_{\text{tl}} \sqrt{s}) + R_{\text{tl},0} \quad (2.3.5a)$$

$$\text{Adiabatic boundary condition:} \quad Z_{\text{tl}} = \frac{A_{\text{tl}}}{\sqrt{s}} \coth(B_{\text{tl}} \sqrt{s}) + R_{\text{tl},0} \quad (2.3.5b)$$

where the thermal loss parameters A_{tl} , B_{tl} and $R_{tl,0}$ are given by

$$A_{tl} = \frac{\rho_{\text{vap}}^{\circ} T^{\circ} \Delta s_{\text{vap}} \sqrt{\alpha_{\text{gl}}}}{\Sigma_{\text{gl}} \left(\frac{dT}{dP} \right)_{\text{sat}} \kappa_{\text{gl}}}, \quad B_{tl} = \frac{l_{\text{gl}}}{\sqrt{\alpha_{\text{gl}}}}, \quad R_{tl,0} = \frac{\rho_{\text{vap}}^{\circ} T^{\circ} \Delta s_{\text{vap}}}{h_{\text{th}} \Sigma_{\text{gl}} \left(\frac{dT}{dP} \right)_{\text{sat}}}$$

with α_{gl} , κ_{gl} , l_{gl} and Σ_{gl} , the thermal diffusivity, thermal conductivity, length and surface area of the glass component, respectively. Again, the model assumes the physical properties to be temperature independent.

In order to facilitate expressing the dynamic system in the time domain, an RC-circuit approximation is considered for the nonlinear expression of Z_{tl} herein, in the form:

$$Z_{tl} \approx R_{tl,0} + \sum_{i=0}^n \left[\frac{1}{R_{tl,i}} + C_{tl,i} s \right]^{-1}, \quad (2.3.6)$$

where n denotes the number of parallel R-C components in series (see RC circuit depicted in red on Figure 2.3). The resistance and capacitance values $R_{tl,i}$ and $C_{tl,i}$ may be estimated to match Z_{tl} over a suitable frequency range.

2.3.3 Complete NIFTE models

A graphical depiction of the R, L, C elements in both the fluid and thermal domains is shown in Figure 2.3. The original NTP model [24] comprises the components shown in black only. Addition of the components in red describing irreversible thermal losses, gives rise to the extended NTP-TL model.

A mathematical formulation of the NTP model (Equations 2.3.7a–2.3.7e) from its electrical circuit analogue is obtained by applying Kirchhoff's voltage and current laws. A minimum of five independent thermodynamic variables is required to fully describe the system, here taken as: the pressure in the adiabatic vapor region, $\hat{P}_{\text{ad}}$; the hydrostatic pressures in the displacer cylinder, $\hat{P}_{\text{d}}$ and in the power cylinders, $\hat{P}_{\text{p}}$; and the volumetric flowrates in the feedback tube, $\hat{U}_{\text{f}}$ and in the power cylinder $\hat{U}_{\text{p}}$. The resulting system of ordinary differential equations (ODEs) is furthermore non-dimensionalized (denoted with $\hat{\cdot}$), by using P° , U° and τ as reference values

for the pressures, flowrates and time, respectively.

$$\frac{d\hat{P}_{ad}}{d\hat{t}} = \frac{\tau[f(\hat{P}_d) - \hat{P}_{ad}]}{R_{th}C_{ad}} + \frac{\tau U^\circ(\hat{U}_f + \hat{U}_p)}{P^\circ C_{ad}} \quad (2.3.7a)$$

$$\frac{d\hat{P}_d}{d\hat{t}} = \frac{\tau U^\circ \hat{U}_f}{P^\circ C_d} \quad (2.3.7b)$$

$$\frac{d\hat{P}_p}{d\hat{t}} = \frac{\tau U^\circ \hat{U}_p}{P^\circ C_p} \quad (2.3.7c)$$

$$\begin{aligned} \frac{d\hat{U}_f}{d\hat{t}} = & \frac{(\tau P^\circ / U^\circ)[L_1 \hat{P}_p - L_p \hat{P}_{ad} - (L_p + L_1) \hat{P}_d]}{L_1(L_d + L_f) + L_p(L_d + L_f + L_1)} \\ & - \frac{\tau[L_1 R_f + L_p(R_f + R_1)]\hat{U}_f + \tau L_p R_1 \hat{U}_p}{L_1(L_d + L_f) + L_p(L_d + L_f + L_1)} \end{aligned} \quad (2.3.7d)$$

$$\begin{aligned} \frac{d\hat{U}_p}{d\hat{t}} = & \frac{(\tau P^\circ / U^\circ)[L_1 \hat{P}_d - (L_d + L_f) \hat{P}_{ad} - (L_d + L_f + L_1) \hat{P}_p]}{L_1(L_d + L_f) + L_p(L_d + L_f + L_1)} \\ & + \frac{\tau[L_1 R_f - (L_d + L_f) R_1] \hat{U}_f - \tau(L_d + L_f) R_1 \hat{U}_p}{L_1(L_d + L_f) + L_p(L_d + L_f + L_1)} \end{aligned} \quad (2.3.7e)$$

The derivation of the NTP-TL model (Equations 2.3.7a'–2.3.7f) proceeds similarly to that of the NTP model above, yet with n extra thermodynamic variables $\hat{P}_{tl,i}$ representing the pressures associated with each thermal loss capacitance $C_{tl,i}$. Equation (2.3.7a') describing the dynamics of pressure $\hat{P}_{ad}$ also comprises an extra term compared with Equation (2.3.7a), to account for the thermal losses.

$$\frac{d\hat{P}_{ad}}{d\hat{t}} = \frac{\tau[f(\hat{P}_d) - \hat{P}_{ad}]}{R_{th}C_{ad}} + \frac{\tau U^\circ(\hat{U}_f + \hat{U}_p)}{P^\circ C_{ad}} - \frac{\tau(\hat{P}_{ad} - \sum_{i=1}^n \hat{P}_{tl,i})}{R_{tl,0}C_{ad}} \quad (2.3.7a')$$

$$\frac{d\hat{P}_{tl,i}}{d\hat{t}} = \frac{\tau(\hat{P}_{ad} - \sum_{i=1}^n \hat{P}_{tl,i})}{R_{tl,0}C_{tl,i}} - \frac{\tau \hat{P}_{tl,i}}{R_{tl,i}C_{tl,i}}, \quad i = 1 \dots n \quad (2.3.7f)$$

Observe that both NTP and NTP-TL models consist of linear ODEs in the state variables, with the exception of the term $f(\hat{P}_d)$ in Equations (2.3.7a) and (2.3.7a') describing the transfer function between $\hat{P}_d$ and $\hat{P}_{th}$. Early models of the NIFTE [6, 7, 8, 9, 21] have considered LTP models (as depicted in Figure 2.2a), in which linear (static) transfer functions of the form $\hat{P}_{th} = K_{th} \hat{P}_d$ is assumed, where K_{th} is a proportional gain. An alternative approach that also preserves linearity uses the DHX models (as depicted in Figure 2.2c) in which (dynamic) transfer function $\hat{P}_{th}s = K_{th} \hat{P}_d$ is imposed [22, 10]. But because they are fully linear, none of

these models can capture the robust periodic oscillations corresponding to a (stable) limit cycle.

To account for the fact that the temperature on the heat-exchanger walls T_{th} cannot increase or decrease indefinitely with the liquid level ③ in the heat-exchanger block, one can introduce a nonlinearity in the form of a saturation function as the liquid level in the displacer cylinder moves away from the equilibrium position (which lies halfway between HHX ④ and CHX ⑤). Markides and co-workers [24] proposed the NTP model. In this model, $\hat{P}_{\text{th}}$ is described using the saturation function

$$f(\hat{P}_{\text{d}}) = \hat{P}_{\text{th}}^{\text{max}} \tanh(K_{\text{th}} \hat{P}_{\text{d}}), \quad (2.3.8)$$

where the parameter $\hat{P}_{\text{th}}^{\text{max}}$ sets the maximum (saturation) temperature amplitude on the heat-exchanger walls; and the nonlinear gain K_{th} determines the maximum rate of heat transfer and is a function of the physical distance between the hot and cold heat elements. More specifically, K_{th} is inversely proportional to the distance between the HHX and the CHX; and directly proportional to the temperature difference between the heat source and sink. The resulting NIFTE models are comprised of nonlinear ODEs, which have been shown to feature limit cycles [24].

2.3.4 Expansion of NIFTE models as switched-linear systems

The sole contributor of the nonlinearity in the NTP-TL model is its temperature profile as depicted in Figure 2.2d, in which the temperature deviation from the reference temperature at the equilibrium position, i.e. T_{hx} , is expressed with a hyperbolic tangent function in Equation (2.2.1).

An alternative model is considered here, where the temperature profile is expressed as a piecewise-affine function of the liquid height z , as shown in Figure 2.4. Specifically, the working fluid is assumed to follow a linear transition between the maximum and the minimum

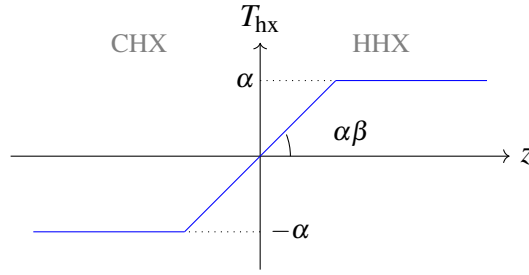


Figure 2.4: The temperature profile along the hot heat exchanger (HHX) and cold heat exchanger (CHX) assumed in the PWLTP and PWLTP-TL models for the NIFTE device. z is the displacement of the vapor-liquid interface from the equilibrium position, that is taken to be the midpoint between HHX and CHX. T_{hx} is the temperature deviation around the reference temperature at $z = 0$.

achievable temperatures, α and $-\alpha$, with a gradient equal to $\alpha\beta$,

$$T_{hx} = \begin{cases} -\alpha, & \text{if } \beta z \leq -1 \\ \alpha\beta z, & \text{if } -1 < \beta z < 1 \\ \alpha, & \text{if } \beta z \geq 1 \end{cases} \quad (2.3.9)$$

As a result, the thermal pressure, P_{th} is transformed into a piecewise affine function of the pressure in the displacer cylinder, P_d ,

$$\hat{P}_{th} = f(\hat{P}_d) = \begin{cases} -\hat{P}_{th}^{\max}, & \text{if } K_{th}\hat{P}_d \leq -1 \\ \hat{P}_{th}^{\max} K_{th}\hat{P}_d, & \text{if } -1 < K_{th}\hat{P}_d < 1 \\ \hat{P}_{th}^{\max}, & \text{if } K_{th}\hat{P}_d \geq 1 \end{cases} \quad (2.3.10)$$

The resulting models comprise the same Equations (2.3.7a–2.3.7e) for NTP, or Equations (2.3.7a’–2.3.7f) for NTP-TL, where $f(\hat{P}_d)$ is now defined in Equation (2.3.10). The definition of $f(\hat{P}_d)$ marks the only difference between the reformulated model and the original NTP and NTP-TL models in Section 2.3.3. These new switched-linear dynamic models are referred to as PWLTP and PWLTP-TL in the rest of the thesis. It is worth noting that such models fall into the category of linear hybrid automaton, for which a variety of tools have been developed for the analysis [60, 61].

2.3.5 Key performance indicators

An important indicator of the performance of a NIFTE is the average useful power output from the load, $\dot{W}_l$ given by

$$\dot{W}_l = \overline{P_l(t)U_l(t)} \quad (2.3.11)$$

where P_l and U_l denote the pressure and volumetric flowrate in the load, respectively; and the average value is calculated over a steady-state cycle of the engine.

A second key performance criterion is the exergetic (or, second law) efficiency of the engine, η_{ex} defined as

$$\eta_{ex} = \frac{\dot{W}_l}{\dot{X}_{th}} = \frac{\overline{P_l(t)U_l(t)}}{\overline{P_{th}(t)U_{th}(t)}} \quad (2.3.12)$$

where P_{th} and U_{th} are the input thermal pressure and volumetric flowrate associated with the phase change, respectively; and $\dot{X}_{th}$ the exergy power input into the device.

For given temperatures of the heat source and sink, recall that the Carnot efficiency, η_{ca} is constant, and therefore η_{ex} is proportional to the thermal efficiency of the engine, $\eta_{th} = \eta_{ex} \eta_{ca}$. The extra pressure and volumetric flowrate states in Equations (2.3.11)–(2.3.12) are readily determined from the application of Kirchoff's circuit laws in the electrical circuit analogue (Figure 2.3).

2.4 Numerical simulation results

2.4.1 Case study definition

The assessment presented in this section is based on a working prototype of the NIFTE. A temperature difference of 80 °C is applied across the heat-exchanger, with the cold source at ambient temperature and a corresponding Carnot efficiency of about 20%. The pumped and working fluids in this prototype are water and n-pentane, respectively. The load corresponds to a delivery pipe of 1 m in length and with a diameter of 0.01 m that would in practice be

connecting a supply tank of water to the destination tank. Further details concerning this prototype may be found in References [8, 9, 21].

Table 2.1: Nominal parameter values in NTP and NTP-TL models of the NIFTE. Values of parameters not associated with thermal loss are in accordance with Ref. [24]; while those of parameters associated with thermal loss are estimated by fitting thermal loss impedance with isothermal boundary conditions using fourth-order RC circuits.

Parameter	Nominal value	Units	Parameter	Nominal value	Units
$\hat{P}_{th}^{max}$	1.37	–	A_{tl}	8.03×10^7	$\text{kg m}^{-4} \text{s}^{-\frac{3}{2}}$
K_{th}	3.30×10^2	–	B_{tl}	4.31	$\text{s}^{\frac{1}{2}}$
R_f	2.13×10^6	$\text{kg m}^{-4} \text{s}^{-1}$	$R_{tl,0}$	3.15×10^7	$\text{kg m}^{-4} \text{s}^{-1}$
R_l	4.08×10^6	$\text{kg m}^{-4} \text{s}^{-1}$	$R_{tl,1}$	8.82×10^6	$\text{kg m}^{-4} \text{s}^{-1}$
R_{th}	5.02×10^8	$\text{kg m}^{-4} \text{s}^{-1}$	$R_{tl,2}$	1.72×10^7	$\text{kg m}^{-4} \text{s}^{-1}$
L_d	1.80×10^5	kg m^{-4}	$R_{tl,3}$	3.83×10^7	$\text{kg m}^{-4} \text{s}^{-1}$
L_f	4.74×10^6	kg m^{-4}	$R_{tl,4}$	2.81×10^8	$\text{kg m}^{-4} \text{s}^{-1}$
L_l	1.27×10^7	kg m^{-4}	$C_{tl,1}$	8.61×10^{-10}	$\text{m}^4 \text{s}^2 \text{kg}^{-1}$
L_p	3.77×10^5	kg m^{-4}	$C_{tl,2}$	5.95×10^{-9}	$\text{m}^4 \text{s}^2 \text{kg}^{-1}$
C_{ad}	1.76×10^{-9}	$\text{m}^4 \text{s}^2 \text{kg}^{-1}$	$C_{tl,3}$	1.79×10^{-8}	$\text{m}^4 \text{s}^2 \text{kg}^{-1}$
C_d	7.35×10^{-8}	$\text{m}^4 \text{s}^2 \text{kg}^{-1}$	$C_{tl,4}$	2.66×10^{-8}	$\text{m}^4 \text{s}^2 \text{kg}^{-1}$
C_p	7.43×10^{-8}	$\text{m}^4 \text{s}^2 \text{kg}^{-1}$	P°	1.013×10^5	Pa
τ	5.00	s	U°	8.00×10^{-4}	$\text{m}^3 \text{s}^{-1}$

Table 2.1 reports the nominal values for the electrical analogues R_i , L_i and C_i in the NTP model (Equations 2.3.7a–2.3.7e & 2.3.8) as well as the parameters K_{th} , $\hat{P}_{th}^{max}$ in the heat transfer function (Equation 2.3.8) and the reference values P° , U° , τ . These were calculated based on the prototype’s physical properties, and it is noteworthy that the NTP model was tested against experimental data with these parameter values (referred to as Set-I in Ref. [24]). In spite of its simplicity compared to complexity of the actual device, the NTP model is capable of early-stage engineering predictions of the NIFTE’s operation and performance.

The same parameter values are used in the NTP-TL (Equations 2.3.7a’–2.3.7f & 2.3.8) and the PWLT-TL models (Equations 2.3.7a’–2.3.7f & 2.3.10). Values for the extra parameters n , $R_{tl,i}$, $C_{tl,i}$ in the approximate thermal-loss impedance (Equation 2.3.6) were determined by fitting the thermal-loss impedance with isothermal boundary condition (Equation 2.3.5a), using

standard MATLAB tools. Specifically, two built-in functions in the System Identification Toolbox in MATLAB are used in conjunction, namely `idtf` and `tfest`. Firstly, an `idtf` model constructs a transfer function with identifiable coefficients. Subsequently, `tfest` estimates the transfer function and computes those coefficients. In order to obtain accurate gain and phase predictions in the frequency range of interest, $0.1\text{--}100\text{ rad s}^{-1}$ for the NIFTE prototype at hand, a minimum of $n = 4$ RC-circuits in series is necessary here (Figure 2.5). The fitted values of $R_{tl,i}$ and $C_{tl,i}$ for $i = 1 \dots 4$ are reported in Table 2.1. Notice that the values of the parameters A_{tl} , B_{tl} , $R_{tl,0}$ in Equation (2.3.5) used for the calibration were also calculated based on the prototype's physical properties, reported in Table 2.1 for completeness. Interestingly, the use of RC-circuit in series to approximate the thermal-loss impedance with isothermal boundary condition also yields an excellent agreement for its counterpart with adiabatic boundary condition (Equation 2.3.5b) in the frequency range $0.2\text{--}100\text{ rad s}^{-1}$. Such close agreement suggests that both types of boundary conditions in Equation (2.3.5) to express the thermal-loss impedance should lead to minimal differences in quantitative predictions by the NTP-TL and PWLTP-TL models. Both types of boundary conditions are indeed relevant as they are representative of different applications of the NIFTE technology.

2.4.2 Numerical integration

The commercially available modeling tool gPROMS ModelBuilder [62, 63, 64] is deployed for direct temporal simulations using the three NIFTE models, i.e. the NTP, the NTP-TL, and the PWLTP-TL models. It is important to note that the NTP and the NTP-TL models are continuous processes, while PWLTP-TL is a hybrid discrete/continuous process.

Numerical integration for continuous processes is implemented in gPROMS ModelBuilder using solvers based on Backward Difference Formulae methods. Effective implicit numerical methods are implemented in this tool, and they can handle stiff systems of ODEs like the NTP and the NTP-TL models. Hybrid dynamical processes require more complicated methods, and are represented as so-called *state-transition networks* (STNs). Barton and co-workers developed a sophisticated two-phase algorithm [65, 66] for such hybrid processes, which forms the basis of solving STNs in gPROMS. gPROMS provides a simple and effective interface to define STNs, and is efficient in handling complicated hybrid systems [67, 68]. In the first phase, state events, i.e. discrete transitions, are detected and the

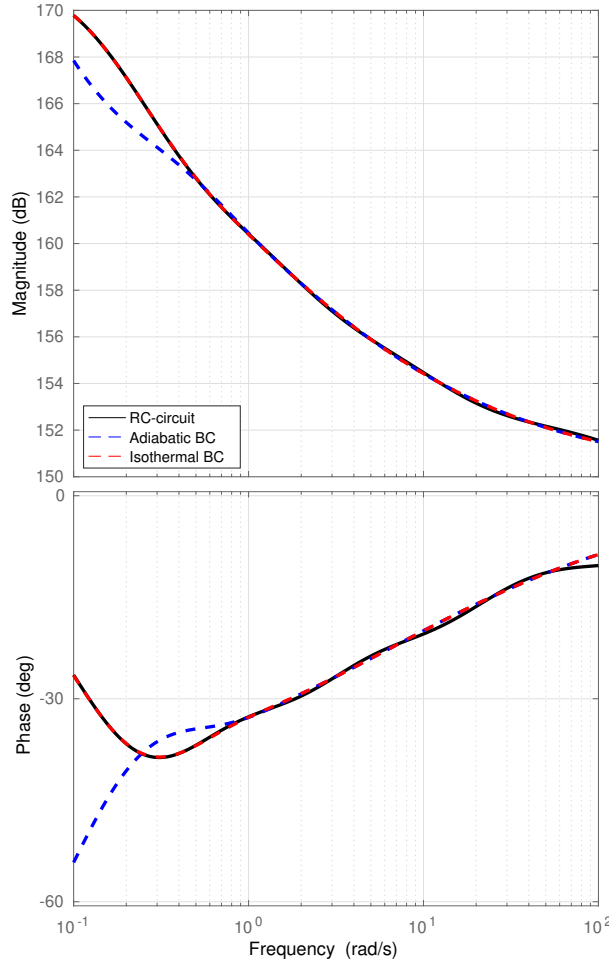


Figure 2.5: Bode plot of thermal loss impedance. The approximate thermal-loss impedance (Equation 2.3.6) with $n = 4$ and values of $R_{tl,i}$, $C_{tl,i}$ as in Table 2.1 is plotted in black solid line. The thermal-loss impedances with isothermal boundary condition (Equation 2.3.5a) and adiabatic boundary condition (Equation 2.3.5b) are plotted in blue and red dashed lines, respectively.

limiting state conditions which triggers the change are determined. This is achieved by constructing interpolation polynomials for the so-called *discontinuity functions* by appending an auxiliary system of differential equations to the original dynamical system, and integrating the augmented system over a small interval. The interpolation polynomials are then solved to find roots that leads to zero crossing of the discontinuity function. If no root is found, no discontinuity is detected; and vice versa. These roots give indications of the exact locations for state events. Note that the system dynamics is locked per integration step, and the discontinuity detection only occurs at the end of the step. Such state events requires reinitialization of the following integration steps. The second phase of the algorithm then ensures consistency in the values of the state variables when discontinuity happens.

There are several key components that need to be specified in gPROMS ModelBuilder. The

types of the variables, including their respective default values and upper and lower limits, if any, are defined in the `Variable Types` entity. The physical mechanisms of the system are then described in the `Model` entity, encapsulating both continuous and discrete parts. The state variables and the time-invariant parameters are also defined here. In this entity, the differential equations associated with continuous state variables are defined to describe the continuous parts of the process. As for the discrete parts, if a STN is being investigated, a `SELECTOR` variable is added to indicate transitions between different state events, or *cases*. Lastly, the `Process` entity defines a particular instance for investigation. For direct temporal simulations of dynamical systems, the initial conditions are provided here.

The code scripts for the `Model` and `Process` entities for the PWLTP-TL model can be found in Appendix A. As for the continuous NTP and NTP-TL models, the code scripts need only be modified slightly to remove the `SELECTOR` variable, and to add a unified mechanism for the state variable P_{ad} .

2.4.3 NIFTE model comparison

Initial conditions to the NIFTE device are specified as $P_{ad}(0) = 0.01P^\circ$, and $U_d(0) = 0.01U^\circ$, while all other thermodynamic variables are set to zero. A comparison of the predicted trajectories for the five key thermodynamic states, P_{ad} , P_d , P_p , U_f and U_p , is shown in Figure 2.6. It is clear that accounting for thermal losses in the model leads to large reductions in the amplitudes of P_{ad} and U_p ; while the effects on the other three states are more limited. Notice that the beating phenomenon, which is also observed experimentally, is predicted by all three models. The trajectories of the NTP-TL and PWLTP-TL models are in excellent agreement overall, with the latter taking slightly longer to settle to a cyclic steady state than the former.

These observations are confirmed by the detailed key performance indicators listed in Table 2.2. While adding extra terms for thermal losses has minimal impact on the oscillation frequency f , the useful power output $\dot{W}_l$ and the exergetic efficiency η_{ex} reduce by over 70%. Finally, the indicators computed with the NTP-TL and PWLTP-TL models are again in excellent agreement, which confirms that either of these models can be used for analysis and optimization of a NIFTE device.

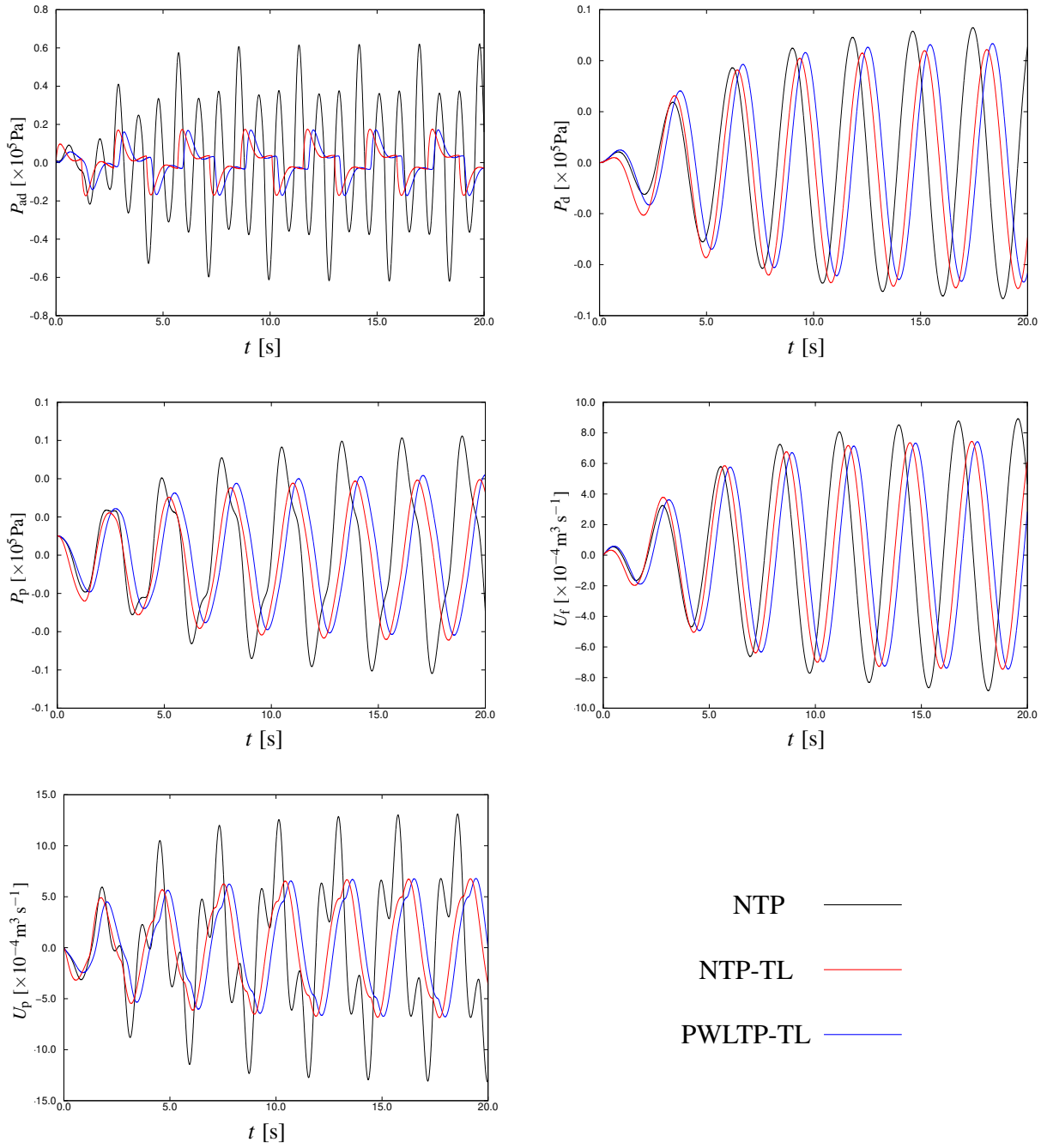


Figure 2.6: Time-simulation of P_{ad} , P_d , P_p , U_f , and U_p in the NIFTE device model using the NTP, the NTP-TL, and the PWLTP-TL models, with $P_{ad,0} = 0.01P^\circ$, $U_{f,0} = 0.01U^\circ$, and the other state variables equal to zero.

2.5 Summary of the contributions

An extension of the existing NTP models of the NIFTE, called NTP-TL, was presented in order to incorporate a description of irreversible thermal losses in the device. A new expression of the

Table 2.2: The key performance indicators of the NIFTE device predicted using the NTP, the NTP-TL and the PWLTP-TL models.

	f (Hz)	$\dot{W}_l$ (W)	η_{ex} (%)
NTP	0.3559	2.4151	2.60
NTP-TL	0.3433	0.6835	0.67
PWLTP-TL	0.3433	0.6869	0.65

temperature profile was introduced in the form of a piecewise-affine function, which results in a switched-linear dynamic system, called PWLTP-TL. A comparison between the NTP, NTP-TL and PWLTP-TL models on the case study of a working prototype as described by parameters in Table 2.1 has revealed that the thermal losses have a large impact on the predicted work output and exergetic efficiency. Moreover, the excellent agreement between the NTP-TL and PWLTP-TL model predictions confirms that either of them might be used to conduct model-based analysis and optimization of the NIFTE.

Chapter 3

Model-Based Analysis and Optimization of NIFTE

3.1 Introduction

The main objective of this chapter is to conduct a detailed assessment of the performance of the NIFTE based on the existing NTP model and the new NTP-TL model described in Chapter 2. The methodology entails a model-based analysis of key design parameters—via the resistors, capacitors, inductors and other gain values in the electrical analogy—in terms of exergetic efficiency and average power output under cyclic steady state (CSS) operation, that leads to new insight into the operation and limits of the NIFTE. A multi-objective optimization based on the NTP-TL model is performed to explore the optimal trade-off between efficiency and power output, and gain further insight into NIFTE’s optimization potential.

The chapter is outlined as below. Section 3.2.1 outlines a dynamic optimization-based approach to CSS detection and analysis. Then Section 3.2.2 embeds this approach into a multi-objective optimization formulation that seeks to maximize the useful work output and the exergetic efficiency of the NIFTE device simultaneously. The main findings from the CSS detection and the associated stability analyses are presented in Section 3.3.1. Extensive sensitivity analyses by varying different sets of key design parameters are performed using both the existing NTP and the new NTP-TL models, and the results are presented and

compared in Sections 3.3.2 to 3.3.5. The most sensitive parameters become the decision variables in the multi-objective optimization problem whose solutions are presented and discussed in Section 3.3.6. Finally, Section 3.4 concludes the chapter. The main findings of this chapter have been published in [69, 52].

3.2 Methods

3.2.1 Cyclic steady-state detection and analysis of the NIFTE

The search for a CSS of the NIFTE can be automated by solving the following constrained dynamic optimization problem,

$$\begin{aligned} \min_{\mathbf{x}_0, T} T & \quad (3.2.1) \\ \text{s.t. } \dot{\mathbf{x}}(t) &= \mathbf{f}(\mathbf{x}(t), \mathbf{p}) \\ \mathbf{x}(0) &= \mathbf{x}(T) = \mathbf{x}_0 \end{aligned}$$

where T denotes a cycle period; the ODEs' right-hand side, $\mathbf{f}$ corresponds to either the NTP model (Equations 2.3.7a–2.3.7e) with the state vector $\mathbf{x}$ comprised of $(\hat{P}_{\text{ad}}, \hat{P}_{\text{d}}, \hat{P}_{\text{p}}, \hat{U}_{\text{f}}, \hat{U}_{\text{p}})$, or the NTP-TL model (Equations 2.3.7a'–2.3.7f) with $\mathbf{x}$ comprised of $(\hat{P}_{\text{ad}}, \hat{P}_{\text{d}}, \hat{P}_{\text{p}}, \hat{P}_{\text{tl}}, \hat{U}_{\text{f}}, \hat{U}_{\text{p}})$; and $\mathbf{p}$ is the vector of model parameters (cf. Table 2.1).

Any solution $\bar{\mathbf{x}}(\cdot, \mathbf{p})$ to Problem (3.2.1) with $T > 0$ corresponds to a CSS of the NIFTE. However, this optimization problem is over-parameterized, in the sense that there exists infinitely many initial conditions $\mathbf{x}_0$ describing a given CSS with $T > 0$. This issue is easily circumvented – and without loss of generality – by forcing one of the initial states to zero; e.g., $\hat{P}_{\text{ad}}(0) = 0$.

The optimization problem (3.2.1) is nonconvex and may therefore present multiple (local) optima. These optima may correspond to different CSS or equilibrium points of the engine, but they could also be “parasitic” local optima corresponding to multiple periods, e.g. $2T, 3T, \dots$, of the same CSS. Herein, a simple single-shooting approach as implemented in the in-house optimization toolkit CRONOS is used to compute such local optima, and a multi-start

search heuristic based on Sobol' sampling is applied to detect the presence of multiple local optima. CRONOS provides an interface to the nonlinear programming (NLP) solver IPOPT [70] coupled with the ODE integrator CVODES in SUNDIALS [71] for computing the objective/constraint functions and their gradients. Both tools are state-of-the-art and widely used in the literature for solving nonlinear problems and ODEs respectively. A local optimization typically converges within a few seconds of CPU time.

The stability of the limit cycle corresponding to a solution $\bar{\mathbf{x}}(\cdot, \mathbf{p})$ can be characterized by analyzing the so-called monodromy matrix $\Phi_{\mathbf{p}}(T, 0)$, which is the fundamental matrix solution to the variational equation [72, 73]

$$\forall \tau, t \in [0, T], \quad \frac{\partial \Phi_{\mathbf{p}}(t, \tau)}{\partial t} = \frac{\partial \mathbf{f}(t, \bar{\mathbf{x}}(t, \mathbf{p}), \mathbf{p})}{\partial \mathbf{x}} \Phi_{\mathbf{p}}(t, \tau) \quad (3.2.2a)$$

$$\text{with } \Phi_{\mathbf{p}}(\tau, \tau) = I \quad (3.2.2b)$$

In practice, the monodromy matrix can be conveniently computed as $\Phi_{\mathbf{p}}(T, 0) = \frac{\partial \bar{\mathbf{x}}(T, \mathbf{p})}{\partial \mathbf{x}_0}$, e.g. using the sensitivity analysis capabilities of the integrator CVODES.

One (at least) of the eigenvalues of the monodromy matrix shall always be equal to 1. The limit cycle $\bar{\mathbf{x}}(\cdot, \mathbf{p})$ is asymptotically stable (attractor) if all of the remaining eigenvalues lie within the open unit disk,

$$\max_i |\lambda_i(\Phi_{\mathbf{p}}(T, 0))| < 1 \quad (3.2.3)$$

If any of these eigenvalues has a modulus greater than 1, the limit cycle is otherwise unstable (repeller).

Bifurcation occurs when a small change made to the values of a system's parameters $\mathbf{p}$ causes a sudden qualitative change in its asymptotic behavior [73]. Of particular interest in the NIFTE are so-called Hopf bifurcations, whereat the equilibrium position $\mathbf{x} = \mathbf{0}$ (origin) loses stability and a periodic solution arises. Such bifurcations can be detected upon repeating the solution of the optimization problem (3.2.1) at a large number of parameter values. The parameters are investigated individually, and their values span over a wide range from 10^{-2} to 10^2 times of their nominal values. One can then use dedicated codes to confirm the presence of a bifurcation and refine its location, such as MatCont [74, 75] which implements a numerical continuation approach in MATLAB. MatCont is an effective tool that is commonly

used for bifurcation analysis, e.g. in Reference [24]. The presence of multiple Hopf bifurcations for a particular parameter can be an indication for the co-existence of multiple limit cycles for certain engine configurations.

3.2.2 Multi-objective optimization of the NIFTE

Finding the design parameters that maximize both the exergetic efficiency and the average power output of a NIFTE leads to a multi-objective optimization (MOO) problem in the form of

$$\begin{aligned} \min_{\mathbf{x}_0, T, \mathbf{p}} \quad & \{\dot{W}_I(\mathbf{x}(\cdot)), \eta_{\text{ex}}(\mathbf{x}(\cdot))\} \\ \text{s.t.} \quad & \dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{p}) \\ & \mathbf{x}(0) = \mathbf{x}(T) = \mathbf{x}_0 \\ & \mathbf{p}^L \leq \mathbf{p} \leq \mathbf{p}^U \end{aligned} \tag{3.2.4}$$

with $\dot{W}_I$ and η_{ex} as defined in Equations (2.3.11)–(2.3.12); and using the same notation as in the problem (3.2.1). Notice that the initial conditions $\mathbf{x}_0$ and cycle time T are decision variables alongside the design variables $\mathbf{p}$ in order to enforce optimal performance under CSS operation. The solution set to the problem (3.2.4) gives rise to a so-called Pareto optimal frontier [76], whereby each point is non-dominated in the sense that no better feasible solution exist in terms of the two objectives simultaneously: betterment of the exergetic efficiency compared to a point on the Pareto frontier results in worsening of the average power output, and vice versa.

Herein, the Pareto frontier of problem (3.2.4) is approximated via scalarization, and the following single-objective optimization problem is subsequently solved for a collection of scalars $\alpha \in [0, 1]$:

$$\begin{aligned} \min_{\mathbf{x}_0, T, \mathbf{p}} \quad & (1 - \alpha)\dot{W}_I(\mathbf{x}(\cdot)) + \alpha\eta_{\text{ex}}(\mathbf{x}(\cdot)) \\ \text{s.t.} \quad & \dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{p}) \\ & \mathbf{x}(0) = \mathbf{x}(T) = \mathbf{x}_0 \\ & \mathbf{p}^L \leq \mathbf{p} \leq \mathbf{p}^U \end{aligned} \tag{3.2.5}$$

Clearly, the problems with $\alpha = 0$ and $\alpha = 1$ correspond to the two extremes of maximizing the average power output and the exergetic efficiency, respectively. It is noteworthy that such a simple scalarization approach fails to describe nonconvex portions of a Pareto frontier in general [76], but the results later on will confirm that this situation does not occur for the Pareto frontiers of interest herein.

In order to tackle the single-objective optimization problems (3.2.5), a two-level decomposition is applied: for a given design vector $\mathbf{p}$, the inner problem determines the values of $\mathbf{x}_0$ and T that yield the CSS with best performance, e.g. in case multiple CSS coexist, or detects that no (nontrivial) CSS is possible; whereas the outer optimization problem operates on the design vector $\mathbf{p}$ only, and solves the inner optimization problem at each iteration in order to calculate the performance under CSS operation for a particular instance of $\mathbf{p}$. The inner optimization problems are solved in the same way as the CSS detection problem (3.2.1) earlier, i.e. using a combination of local search and multi-start search heuristic. As for the outer problem, a simple Nelder-Mead simplex algorithm [77], that is a derivative-free approach with slow convergence but tolerant to numerical noise, is implemented with a termination tolerance on the minimum simplex diameter (set to 10^{-3} in log-scale here). In order to rule out convergence to local (suboptimal) solutions and initialize the Nelder-Mead simplex algorithm, the search space is probed using low-discrepancy sampling based on Sobol' sequences [78] (with 10,000 points here), and the NIFTE performance under CSS operation is computed at each sampling point. Algorithm 1 provides a summary of the multi-objective optimization procedure. The parameter range is set to ± 3 orders of magnitude around their nominal values (see below), and the set of weights considered in the scalarized cost of the problem (3.2.5) are $\{0, \frac{1}{2000}, \frac{1}{1000}, \frac{1}{500}, \frac{1}{200}, \frac{1}{100}, \frac{1}{50}, \frac{1}{20}, \frac{1}{10}, \frac{1}{5}, 1\}$. Each run of the Nelder-Mead simplex algorithm in order to determine one point on the Pareto frontier takes up to a few hundred iterations to converge.

3.3 Results and discussion

The same case study as previously defined in Section 2.4.1 is considered. The nominal parameter values used in both the NTP and NTP-TL models can be found in Table 2.1.

Algorithm 1 Multi-objective optimization procedure for NIFTE design.

Input: Design parameter subset $\mathbf{p} \in \mathbb{R}^n$, and parameter range $[\mathbf{p}^L, \mathbf{p}^U]$

Exploration:

- 1 Generate sample points $\mathbf{p} \in [\mathbf{p}^L, \mathbf{p}^U]$ using Sobol' sampling
- 2 For each sample point, apply multi-start optimization to solve the CSS problem (3.2.1), and record the corresponding initial conditions $\mathbf{x}_0$, cycle time T , average power output $\dot{W}_l$, and exergetic efficiency η_{ex}

Refinement: Repeat for different weights $\alpha_i \in [0, 1]$, $i = 1, 2, \dots$

- 3 Identify the $n + 1$ sampling points yielding the maximal scalarized cost, $(1 - \alpha_i)\dot{W}_l + \alpha_i\eta_{ex}$, and form an initial simplex with these points
- 4 **Nelder-Mead simplex:** Repeat until convergence
 - 4.1 Generate new candidate points using the Nelder-Mead simplex rules (dilation, contraction, reflection)
 - 4.2 Apply multi-start optimization to solve the CSS problem (3.2.1) at each candidate point, and record the corresponding initial conditions $\mathbf{x}_0$, cycle time T , average power output $\dot{W}_l$, and exergetic efficiency η_{ex}
 - 4.3 Form a new simplex by updating one of the vertices in the current simplex

Output: Points on the Pareto frontier of the problem (3.2.4), corresponding to the selected weights

3.3.1 Cyclic-steady-state multiplicity and stability

This subsection investigates the asymptotic behavior of a NIFTE based on the NTP and NTP-TL models. It has already been established [24], based on the NTP model, that a given engine may or may not exhibit a limit cycle. A key finding herein is that a given engine could in fact exhibit *multiple* CSS, which could be either stable or unstable, an observation that had never been reported before.

To illustrate this behavior, a particular case study for the NIFTE is first considered with all its parameters at their nominal values (Table 2.1), except the feedback inductance $L_f = 4.74 \times 10^5 \text{ kg m}^{-4} \text{ s}^{-1}$ (10-fold smaller than the nominal value). When using the NTP model and solving the problem (3.2.1) as described earlier, three possible asymptotic behaviors are identified for this NIFTE configuration. The first one is a trivial equilibrium point at the origin,

which is unstable. The other two are limit cycles with periods $T = 1.30$ s (CSS1) and $T = 0.88$ s (CSS2). The spectra of the monodromy matrices (Equation 3.2.2) at CSS1 and CSS2 are reported in Table 3.1. Since all the eigenvalues of the monodromy matrix at CSS1 lie within the unit disk (Equation 3.2.3), this CSS describes a stable limit cycle. By contrast, the monodromy matrix at CSS2 has two of its eigenvalues outside the unit disk (shown in red), and so this is an unstable limit cycle.

Table 3.1: Asymptotic stability and performance of a NIFTE with nominal parameter values (Table 2.1), except $L_f = 4.74 \times 10^5 \text{ kg m}^{-4} \text{ s}^{-1}$. The eigenvalues causing instability are shown in red. The period, exergetic efficiency, power output and thermal input of the cyclic steady states are also reported for comparison. CSS1: stable limit cycle of NTP model. CSS2 unstable limit cycle of NTP model. CSS1': (unique) stable limit cycle of NTP-TL model. The origin is an unstable equilibrium point of both NTP and NTP-TL models.

	CSS1	CSS2	CSS1'
	$-0.4704+0.2959i$	$-0.3848+1.2078i$	1.0000
	$-0.4704-0.2959i$	$-0.3848-1.2078i$	0.4661
	0.1028	0.0348	$-0.0971 + 0.0681i$
	0.3093	0.7958	$-0.0971 - 0.0681i$
Spectrum	1.0000	1.0000	0.0467
	—	—	0.0358
	—	—	0.0001
	—	—	0.0000
	—	—	0.0000
CSS period, T [s]	1.31	0.88	1.69
Exergetic efficiency, η_{ex} [%]	3.07	24.4	1.05
Average power output, $\dot{W}_1$ [W]	0.79	1.33	0.26
Average exergy input, $\dot{X}_{\text{th}}$ [W]	25.7	5.44	24.5

The pressure-volume phase diagrams in Figure 3.1 are obtained by simulating the NIFTE over an extended time horizon, from different initial states. They represent the pressure P_1 against the volume V_1 of the working fluid, given by $\dot{V}_1 = U_1$, and so the perimeter of the cycle

limit is proportional to the work output, W_1 over one complete cycle:

$$W_1 = \int_0^T P_1(t) U_1(t) dt = \oint P_1 dV_1$$

It is clear that CSS1 acts as an attractor for the trajectories, and the origin as a repeller (Figure 3.1A). Furthermore, trajectories initialized near CSS2 eventually escape away from that limit cycle and are again attracted by CSS1, thereby showing two clusters of trajectories (Figure 3.1B). All of these observations are in agreement with the numerical stability results in Table 3.1.

The comparison of performance indicators in the bottom half of Table 3.1 reveals that a 70%-greater average useful power $\dot{W}_1$ (Equation 2.3.11) is generated by operating the NIFTE at the (unstable) CSS2 compared with the (stable) CSS1. Because the engine is attracted to CSS1 when initialized near the origin (Figure 3.1A), reaching (and then remaining around) an alternative, potentially better limit cycle such as CSS2 would require a tailored control strategy. This could be done, for example, by manipulating the valve in the feedback connection ⑥ (Figure 2.1), a dynamic control that has never been applied in practice however. Notice also that CSS2 has a 30%-shorter period T and an 8-fold greater exergetic efficiency η_{ex} (Equation 2.3.12) than CSS1. The latter is not only the result of the greater useful power output at CSS2, but also a 5-fold lower exergy power input $\dot{X}_{\text{th}}$ (Equation 2.3.12) into the engine from the heat source. Coincidentally, the combination of a shorter period and a larger power output at CSS2 results in about the same work output during a complete cycle of CSS1 ($W_1 = 1.03\text{J}$) or CSS2 ($W_1 = 1.07\text{J}$). This can be seen on Figure 3.1B, where the limit cycles for CSS1 and CSS2 appear to have identical perimeters.

Unlike the NTP model, a simulation of the same NIFTE with the NTP-TL model in Problem (3.2.1) predicts a unique, stable limit cycle (CSS1'). This CSS has a slightly longer period than CSS1, while both the exergetic efficiency and power output are reduced by about 3-fold (Table 3.1). This dramatic reduction is of course attributed to the extra thermal losses in NTP-TL, confirming that their inclusion is paramount to the accuracy of the model predictions. The perimeter of the limit cycle CSS1' on the pressure-volume phase diagrams in Figure 3.1C is accordingly much shorter in comparison with that of CSS1 on Figure 3.1A.

In the following Subsections 3.3.2 to 3.3.5, a parametric analysis of the NIFTE is carried

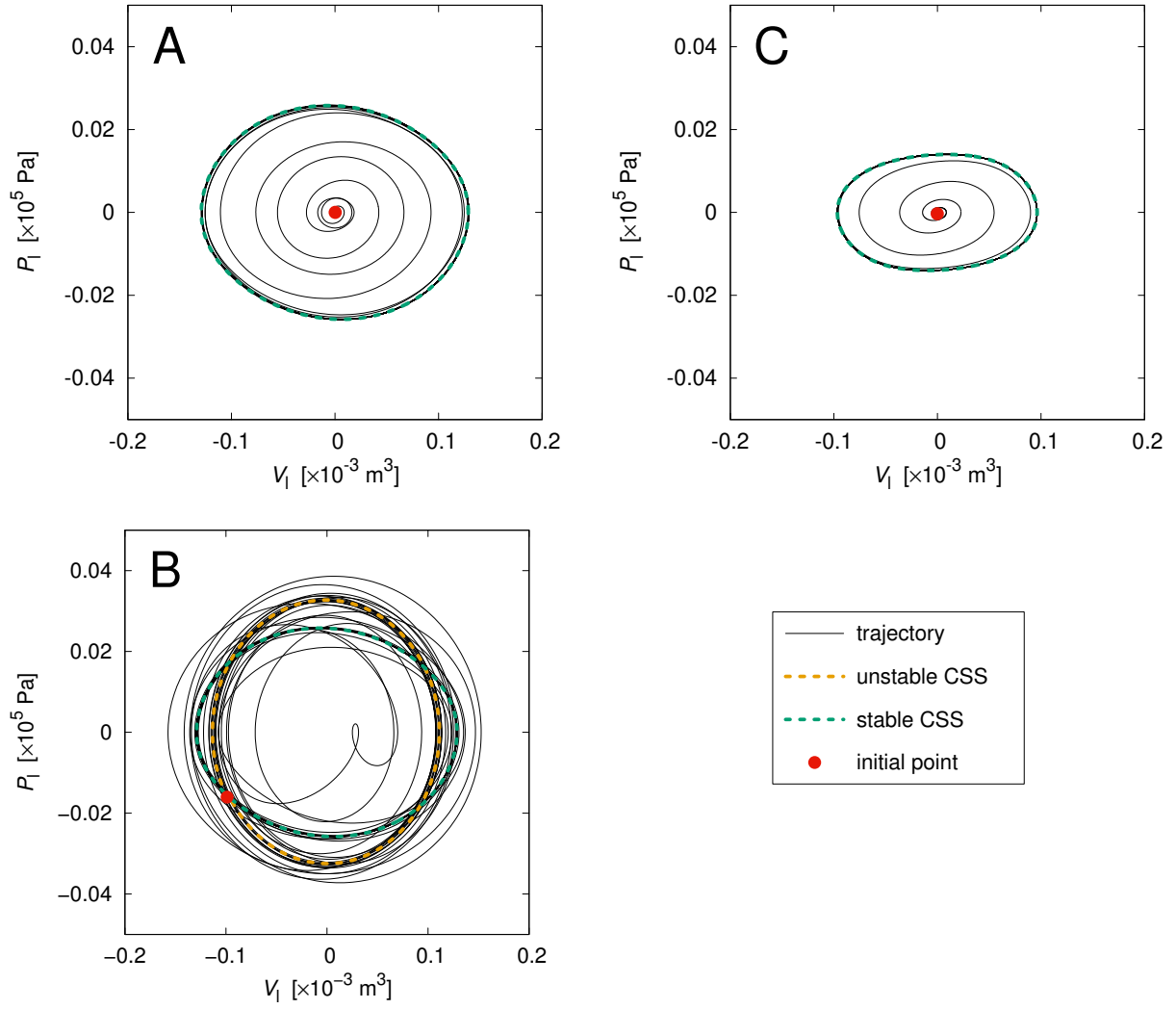


Figure 3.1: Pressure-volume phase diagrams of a NIFTE simulated over an extended time horizon. The NIFTE is simulated using either the NTP or NTP-TL model with nominal parameters values (Table 2.1), except except $L_f = 4.74 \times 10^5 \text{ kg m}^{-4} \text{ s}^{-1}$. A: Trajectory of the NTP model repelled from the origin and attracted to CSS1. B: Trajectory of the NTP model repelled by CSS2 and attracted to CSS1. C: Trajectory of the NTP-TL model repelled from the origin and attracted to CSS1'.

out by varying (combinations of) the design parameters in the feedback, displacer and power cylinders and heat-exchanger, and analyzing the performance of the corresponding CSS using both the NTP and NTP-TL models. Finally, multi-objective optimization is conducted with the NTP-TL model in Subsection 3.3.6 in order to investigate the optimal trade-offs between exergetic efficiency and power input when varying three or more parameters.

3.3.2 Parametric analysis of (R_f, L_f)

Previous studies [8, 9] have shown that the performance of NIFTE is highly sensitive to the feedback resistance R_f and the feedback inductance L_f . These parameters can be varied in a physical device by modifying the geometry of the feedback tube (e.g., l_f , d_f or A_f ; see Equation 2.3.1) or the position of a feedback valve thereof. This subsection presents a joint parametric analysis of the feedback parameters R_f and L_f using both models, while all other parameters are held constant at their nominal values (Table 2.1). The superscript $+$ is used to denote the ratio between a parameter and its nominal value.

3.3.2.1 NTP model

Both the exergetic performance η_{ex} and useful power output $\dot{W}_l$ predicted by the NTP model under CSS operation are shown in Figure 3.2, for a wide parameter range spanning several orders of magnitude around the nominal parameter values. Each point in the plots represents a CSS found for that particular configuration; and the black regions correspond to configurations with no CSS detected. By repeating the solution of the problem (3.2.1) over this range, up to two limit cycles could be identified, denoted by CSS1 and CSS2 in agreement with previous notation. The stability of each of these CSS is further determined by the eigenvalues of their monodromy matrices (Equation 3.2.2), and the separation between stable and unstable CSS is reported on each plot in Figure 3.2. Notice that multiple CSS only co-exist in the subdomain $(R_f^+, L_f^+) \in [0.001, 10] \times [0.044, 0.145]$, with either one stable and one unstable CSS, or even two stable CSS.

As far as the effect of R_f is concerned, a reduction of the feedback resistance with all other parameters fixed leads to a larger useful power output and a larger exergetic efficiency, for both CSS1 (Figure 3.2a) and CSS2 (Figure 3.2b). This behavior is consistent with previous parametric analyses conducted with the NTP model [24]. The parameter L_f is also found to have a high impact on the exergetic performance η_{ex} and useful power output $\dot{W}_l$, but analyzing these variations is more intricate. In the case of CSS1, an overall trend is for the useful power output to increase as L_f is lowered, whereas the exergetic efficiency is maximal around the nominal L_f value ($L_f^+ = 1$). These different trends, alongside the wavy pattern on Figure 3.2a, is attributed to a shift in phase between that oscillations in the load and displacer cylinders, as

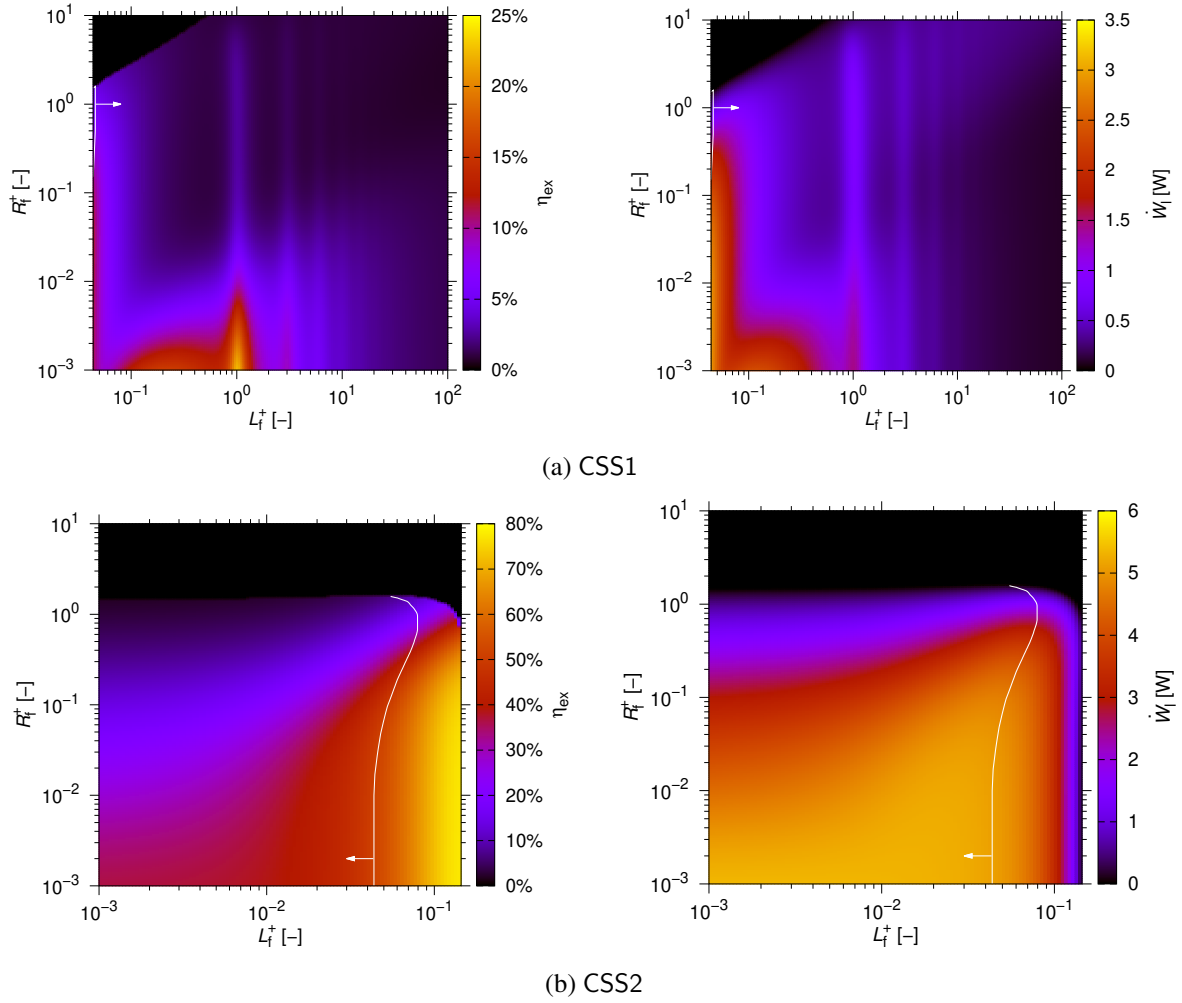


Figure 3.2: Predicted effect of R_f and L_f on NIFTE performance using the NTP model with n-pentane as the working fluid. Two distinct cyclic steady states are detected, denoted by CSS1 and CSS2, which are overlapping for certain (R_f, L_f) combinations. The effect on both the exergetic efficiency η_{ex} (left plots) and the average power output $\dot{W}_l$ (right plots) is shown for each cyclic steady-state. R_f^+ and L_f^+ refer to R_f and L_f normalized by their respective nominal values in Table 2.1. The remaining parameters are set to their nominal values. The separatrix between stable and unstable limit cycles are shown in white, with arrows pointing in the direction of stable limit cycles.

a result of the variation in gas-liquid interface and hydraulic pressure due to the phase change. Notice also that nearly all of CSS1 points are stable, apart from those corresponding to the smallest L_f values that turn into unstable before vanishing. In the case of CSS2 by contrast, a trend is for the exergetic efficiency to decrease as L_f is lowered, while the useful power output passes through a maximum that is shifting to lower L_f values as R_f itself is lowered.

The results in Figure 3.2 thus reveal a trade-off between η_{ex} and $\dot{W}_l$ in terms of maximizing NIFTE performance. The maximal efficiency η_{ex} is predicted to be as high as 77% at $(R_f^+, L_f^+) = (0.001, 0.145)$, yet with a very small power output of 0.1 W; whereas the

maximum power output is 5.4 W at $(R_f^+, L_f^+) = (0.001, 0.001)$, but with a lower exergetic efficiency of about 36%. These values should be considered relative to the nominal NIFTE configuration ($R_f^+ = L_f^+ = 1$), which has a predicted exergetic efficiency and average power output of about 2.6% and 0.86 W, respectively. Whether the primary target is to maximize the efficiency or to produce more power should normally drive the design of a NIFTE in practice. Here, a good compromise could aim for a design near the separatrix between stable and unstable CSS2 in Figure 3.2b; for instance at $L_f^+ = 0.03$ and $R_f^+ < 0.01$, where the predicted power output and exergetic efficiency are greater than 5.2 W and 45%, respectively. Because these CSS are at the verge of instability, a control strategy [79] might become necessary in order to stabilize the NIFTE nonetheless. As per their definitions in Equations (2.3.1), R_f and L_f either increase or decrease in the same direction when the dimensions of the feedback tube are modified, including the tube's length and diameter. Yet, R_f is more sensitive to changes in the tube diameter. Careful choices of the feedback tube's geometry need to be made to approach the recommended configuration.

Although the NTP model is capable of accurate predictions of several key operational characteristics of the NIFTE, such as its oscillation frequency, we shall see in the remainder of the chapter that such improvements in efficiency and power output are unfortunately an artifact of this model, as it fails to describe irreversible thermal losses.

3.3.2.2 NTP-TL model

The same parametric analysis of the NIFTE is conducted, now accounting for thermal losses with the NTL-TL model. The predicted exergetic efficiency η_{ex} and average power output $\dot{W}_1$ under CSS operation are presented in Figure 3.3, for a wide parameter range spanning several orders of magnitude around the nominal parameter values. A first notable difference with the NTP-TL model is that a unique limit cycle is detected via the repeated solution of the problem (3.2.1) for $(R_f^+, L_f^+) \in [0.001, 10]^2$, which is referred to as CSS1' in agreement with Table 3.1. This CSS1' is furthermore stable over the explored parameter range.

The variations in exergetic efficiency and power output of CSS1' do not display the same wavy pattern that could be observed for CSS1 in Figure 3.2a either, although CSS1' presents several minima and maxima across the parameter range. The maximal efficiency η_{ex} is

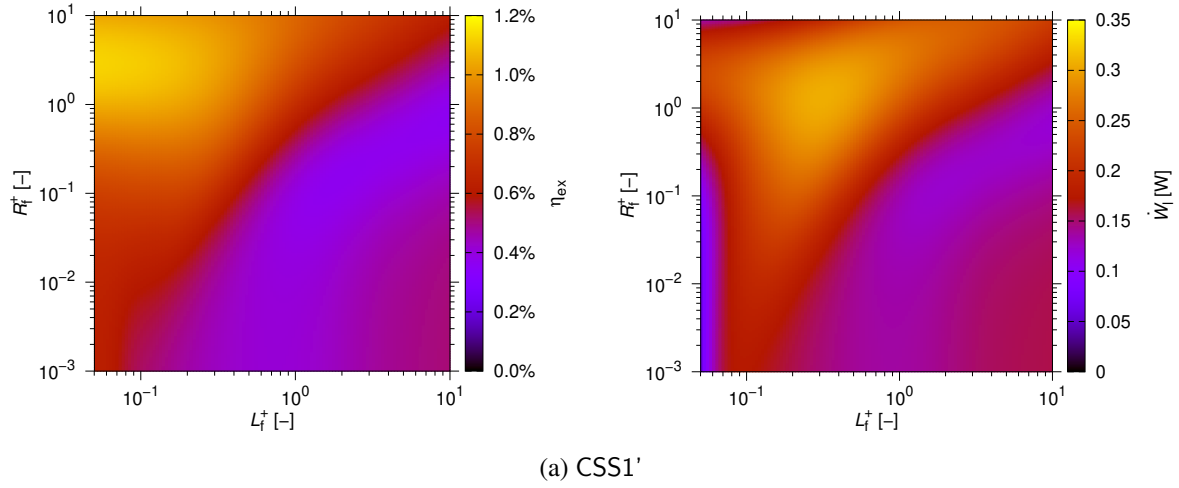


Figure 3.3: Predicted effect of R_f and L_f on NIFTE performance using the NTP-TL model with n-pentane as the working fluid. A unique, stable cyclic steady state is detected here. The effect on both the exergetic efficiency η_{ex} (left plots) and the average power output $\dot{W}_l$ (right plots) is shown for each cyclic steady state. R_f^+ and L_f^+ refer to R_f and L_f normalized by their respective nominal values in Table 2.1. The remaining parameters are set to their nominal values. All of the limit cycles are stable on both plots.

predicted to be around 1.1% at $(R_f^+, L_f^+) = (3.16, 0.001)$, while a maximal average power output $\dot{W}_l$ of 0.31 W is predicted at $(R_f^+, L_f^+) = (1.38, 0.316)$. The values are now to be compared with the predicted exergetic efficiency of 0.69% and predicted average power output of 0.24 W for the nominal NIFTE configuration ($R_f^+ = L_f^+ = 1$). A general trend on Figure 3.3 is for both the efficiency and power output to increase as L_f is reduced from its nominal value. In particular, these results suggest that enhanced performance of the NIFTE prototype by 30–50% could be achieved via a significant reduction in L_f , while maintaining R_f close to nominal. However, this optimal configuration is hard to achieve in practice given the two parameters' dependency on the feedback tube's geometry. A certain degree of compromise on the NIFTE's performance is inevitable if this pair of (R_f, L_f) alone is modified.

These predictions by the NTP-TL model are a major departure from those of the NTP model. The comparison between Figures 3.2a and 3.3a, where the feasible ranges of (R_f, L_f) largely overlap, shows that accounting for thermal losses may cause a 10- to 20-fold drop in the predicted exergetic efficiency and average power output, especially for the most promising NIFTE configurations determined by the NTP model. Despite the fact that the NTP and NTP-TL model predictions are close for certain operational characteristics of the NIFTE, such as its oscillation frequency, these extreme differences in efficiency and power output predictions shed doubt on whether the NTP model is capable of accurate, quantitative predictions of the NIFTE

performance over wide parameter ranges. By contrast, the level of performance predicted by the NTP-TL model is in good agreement with direct experimental evidence [8, 9], where the exergetic efficiency of the NIFTE prototype was measured between 0.4–1.6%. It was also observed experimentally that the exergetic efficiency could improve by increasing the feedback resistance R_f , a behavior that is not described by the NTP model.

3.3.3 Parametric analysis of (C_p, L_p)

Apart from the feedback cylinder, the design of the power cylinder where the working fluid is in direct contact with the pumped fluid in the load (Figure 2.1), is also known to impact the NIFTE performance greatly. This subsection carries out a joint parametric analysis of the capacitance C_p and inductance L_p based on the NTP-TL model only, with all the other parameters held constant at their nominal values (Table 2.1). Both C_p and L_p could be varied in a physical device by modifying either the length l_p or the cross-section area A_p of the power cylinder (Equation 2.3.1).

The predicted exergetic efficiency η_{ex} and average power output $\dot{W}_1$ under CSS operation are presented in Figure 3.4, for $(L_p^+, C_p^+) \in [0.01, 100]^2$. By repeating the solution of the problem (3.2.1) over this parameter range, three distinct limit cycles could be identified, denoted by CSS1", CSS2" and CSS3". Notice that CSS1" is the unique limit cycle in a wide domain around the nominal NIFTE design ($L_p^+ = C_p^+ = 1$). The domains of existence of CSS2" and CSS3" overlap with CSS1" for values of $C_p^+ \leq 0.025$ and $L_p^+ \geq 17$, respectively, yet these domains are much narrower.

By far the most promising limit cycle is CSS1" in Figure 3.4a, both in terms of exergetic efficiency and power output. The entire range of CSS1" is furthermore stable. The maximal efficiency η_{ex} is predicted to be slightly below 1% at $(C_p^+, L_p^+) = (24.5, 0.105)$, with efficiencies above 0.9% achieved in a wide subdomain of CSS1" with $C_p^+ \geq 3$ and $L_p^+ \leq 1$. A maximal average power output $\dot{W}_1$ of 0.31 W is predicted at $(C_p^+, L_p^+) = (4.90, 0.01)$, with improved power outputs again observed in a wide band with $3 \leq C_p^+ \leq 10$ and L_p^+ around or below nominal. Similar to the sensitivity analysis for (R_f, L_f) earlier, these results suggest that the performance of the NIFTE prototype could be enhanced by 30–50% by increasing C_p and reducing L_p from their nominal values in the NIFTE prototype at hand. Unlike the pair of $(R_f,$

L_f), C_p and L_p change in opposite directions when the cross-sectional area of the power cylinder is modified, and C_p is independent of the cylinder's length. Hence, the recommended operating condition can possibly be achieved by increasing the power cylinder's diameter and reducing its length.

3.3.4 Parametric analysis of (C_d, L_d)

A third sensitivity analysis is conducted using the NTP-TL model in terms of the capacitance C_d and inductance L_d of the displacer cylinder, with all the other parameters held constant at their nominal values (Table 2.1). Similar to the parameter pair (C_p, L_p) earlier, both C_d and L_d could be varied in a physical device by modifying either the length l_d or the cross-section area A_d of the displacer cylinder (Equation 2.3.1).

The predicted exergetic efficiency η_{ex} and average power output $\dot{W}_I$ under CSS operation are shown in Figure 3.5, for $(L_d^+, C_d^+) \in [0.01, 100]^2$. By repeating the solution of the problem (3.2.1) over this parameter range, two distinct limit cycles emerged, denoted by CSS1''' and CSS2'''. Like previously with (C_p, L_p) , it is found that CSS1''' is the unique limit cycle in a wide subdomain around the nominal NIFTE design ($L_d^+ = C_d^+ = 1$). The domain of existence of CSS2''' overlaps with that of CSS1''' over a narrow band with $C_d^+ \in [0.046, 0.063]$. A majority of the limit cycles are furthermore stable, under either CSS1''' or CSS2'''.

The most promising limit cycle is CSS1''' in Figure 3.5a, with predicted exergetic efficiencies and power outputs three- to four-times higher than CSS2''' in Figure 3.5b. A maximal efficiency η_{ex} of about 0.93% is predicted at $(C_d^+, L_d^+) = (0.229, 0.010)$; and the maximum power output $\dot{W}_I$ is predicted to be 0.32 W at $(C_d^+, L_d^+) = (0.055, 95.5)$. The gain in power output is about identical to the gains predicted by adjusting the feedback parameters (R_f, L_f) or the displacer parameters (C_p, L_p) ; whereas the gain in efficiency, in the order of 35% compared to the nominal NIFTE, is lower than the gains predicted by adjusting by (R_f, L_f) or (C_p, L_p) . A general trend on Figure 3.5a is for both the efficiency and power output to increase as C_d is reduced to within to the range $C_d^+ \in [0.15, 0.3]$, while the parameter L_d exhibits a relatively lower sensitivity in the NIFTE prototype. Resembling the pair of (C_p, L_p) , modifying the cross-section area of the displacer cylinder leads to changes in opposite directions for C_d and L_d ; and modifying the cylinder's length impacts L_d only. Therefore, the

recommended settings of (C_d, L_d) can be achieved by reducing both the displacer cylinder's diameter and length.

3.3.5 Parametric analysis of K_{th}

To complete the assessment, a final parametric analysis of the nonlinear gain K_{th} is conducted using the NTP-TL model, with all the other parameters held constant at their nominal values (Table 2.1). The gain represents the maximum rate of heat transfer across the heat-exchanger, which may be varied in a physical device by adjusting either the distance between the HHX and CHX elements (Figure 2.1), or the temperature difference thereof.

The top-left plot in Figure 3.6 shows the predicted exergetic efficiency η_{ex} versus average power output $\dot{W}_l$ under CSS operation, for $K_{th}^+ \in [0.09, 10]$. The embedded table reports the values of K_{th}^+ at four selected points (red circles), in addition to the nominal point (green triangle). The general trend is for η_{ex} to increase and $\dot{W}_l$ to decrease as K_{th} is reduced, i.e. as the HHX and CHX elements are drawn further apart or the temperature difference is reduced. The sensitivity of K_{th}^+ appears to be very low around the nominal value ($K_{th}^+ = 1$), where increasing K_{th}^+ only leads to marginal variations in average power output and exergetic efficiency. The engine performance becomes notably more sensitive to values of $K_{th}^+ < 0.5$, but a CSS ceases to exist below $K_{th}^+ \approx 0.09$ due to the HHX and CHX elements no longer allowing for a sufficient heat transfer rate. The exergetic efficiency there is predicted to increase by over 10% compared to the nominal efficiency, but the NIFTE may no longer be practical because of the dramatic reduction in power output that ensues.

A physical interpretation of this parametric analysis is that the performance of the NIFTE would be seldom affected if either the temperature difference or the distance between heat-exchangers were to vary by a factor of 2 or even larger. In particular, these results suggest that a NIFTE might indeed remain feasible for temperature differences lower than 30 °C, without reducing the power output or exergetic efficiency significantly.

3.3.6 Multi-objective optimization by varying multiple parameters

Results of the parametric analyses conducted in the previous subsections revealed that adjusting the pairs (R_f, L_f) , (C_p, L_p) and (C_d, L_d) of design parameters in the different NIFTE cylinders could enhance the work output by up to 30% or the exergetic efficiency by over 50% in the NIFTE prototype. For instance, the top-right plot in Figure 3.6 shows the Pareto frontier of maximal exergetic efficiency versus maximal power output for the pair (R_f, L_f) . This Pareto frontier was constructed using the same parametric sensitivity data as in Figure 3.3. Values of (R_f^+, L_f^+) corresponding to the labeled points A–E on the frontier are also reported in the embedded table for comparison. Moving along the Pareto frontier from A to E in this case entails increasing the value of R_f^+ by 2-fold, while simultaneously decreasing the value of L_f^+ toward zero. Such Pareto frontiers are helpful to visualize the trade-offs between objectives, identify an acceptable compromise based on engineering priorities, and assess the sensitivity of Pareto-optimal decisions toward the design parameters.

Next, multi-objective optimization is applied to characterize the Pareto frontier for combinations of three or more design parameters, by following the scalarization and bi-level decomposition approach described earlier. The Pareto frontier obtained by adding the design parameter C_p to the pair (R_f, L_f) is displayed on the bottom-left plot in Figure 3.6. Also shown on this plot are the results of the Sobol' sampling used to cross-check the Pareto points determined by the Nelder-Mead simplex algorithm.

A first observation is that the performance of the NIFTE is clearly enhanced by the addition of an extra design parameter to the optimization problem, especially in terms of the maximal power output which improves by a further 15%. Values of the triplet (R_f^+, L_f^+, C_p^+) at points A–E along the frontier (embedded table in bottom-left plot) suggest a radically different design strategy compared with the (R_f, L_f) case on the top-right plot. Here, reaching the Pareto frontier entails decreasing the value of R_f^+ toward zero and moving along the frontier from A to E by reducing L_f^+ from 0.34 to 0.28 and C_p^+ from 4.7 to 3.9 simultaneously. Notice, in particular, the high sensitivity to changes in L_f and C_p , whereby the full frontier from A to E is described by a mere 20% variation in their values. A practical implication of this high sensitivity is that mathematical models capable of highly accurate predictions will likely be paramount to support the detailed design of NIFTE, also calling for dedicated experimental campaigns to

further validate these models.

It is important to note, at this point, that the selection of alternative triplets as decision variables in the multi-objective optimization problem, including (R_f, L_f, L_p) and (R_f, L_f, C_d) , produces similar improvements of the NIFTE performance. This, in turn, motivates the consideration of larger parameter subsets in the multi-objective optimization. The bottom-right plot in Figure 3.6, for instance, shows the Pareto frontier corresponding to the quadruplet (R_f, L_f, C_p, C_d) . Comparison with the bottom-left plot shows marginal improvement in terms of exergetic efficiency or power output after adding the parameter C_d . The consequence of C_d being redundant with the other parameters (R_f, L_f, C_p) also manifests through the lack of consistent trends in the design parameter values at points A–E along the frontier (embedded table in bottom-right plot). The consideration of alternative quadruplets to optimize does not produce further improvement for this NIFTE prototype. And even the optimization of $(R_f, L_f, C_p, L_p, C_d, L_d, K_{th})$ all together leads to either a maximal efficiency close to 1.2% or a maximal power output of 0.35 W. Overall, these results establish that near-optimal performance of the NIFTE can be achieved through targeted modifications of three parameters only, such as (R_f, L_f, C_p) . In other words, optimizing the length and/or diameter of the feedback and power cylinders would be sufficient in practice for NIFTE to reach its full potential.

3.4 Summary of the contributions

Both the nonlinear dynamic models, NTP and NTP-TL, were used as the basis for the investigations in this chapter. For the first time, the co-existence of multiple limit cycles could be predicted for certain NIFTE configurations, which may be either stable or unstable.

A parametric analysis of the most sensitive design parameters of the NIFTE device (feedback, displacer and power cylinders, and heat-exchangers) was conducted, followed by a systematic multi-objective optimization to characterize the Pareto frontier of maximal exergetic efficiency versus maximal power output. The results presented establish that failure to account for irreversible thermal losses can grossly overpredict the exergetic efficiency and power output that may be delivered by a NIFTE. The assessment conducted with the extended model predicts improvement greater than 50% in either the work output or the exergetic

efficiency via targeted design modifications to an early-stage NIFTE prototype. The optimization study furthermore revealed that near-optimal performance of the NIFTE can be realized by adjusting combinations of no more than three parameters, such as the dimensions of the feedback tube and power cylinder.

The existence of multiple limit cycles calls for the use of global dynamic optimization to detect the cyclic steady states in the NIFTE, which is a very challenging task. Therefore, in the forthcoming chapters, the focus will be on the development of improved techniques for bounding the NIFTE dynamics, with a view to conducting deterministic global optimization. All future analyses of the NIFTE will be based on the NTP-TL model or its switched-linear counterpart PWLTP-TL.

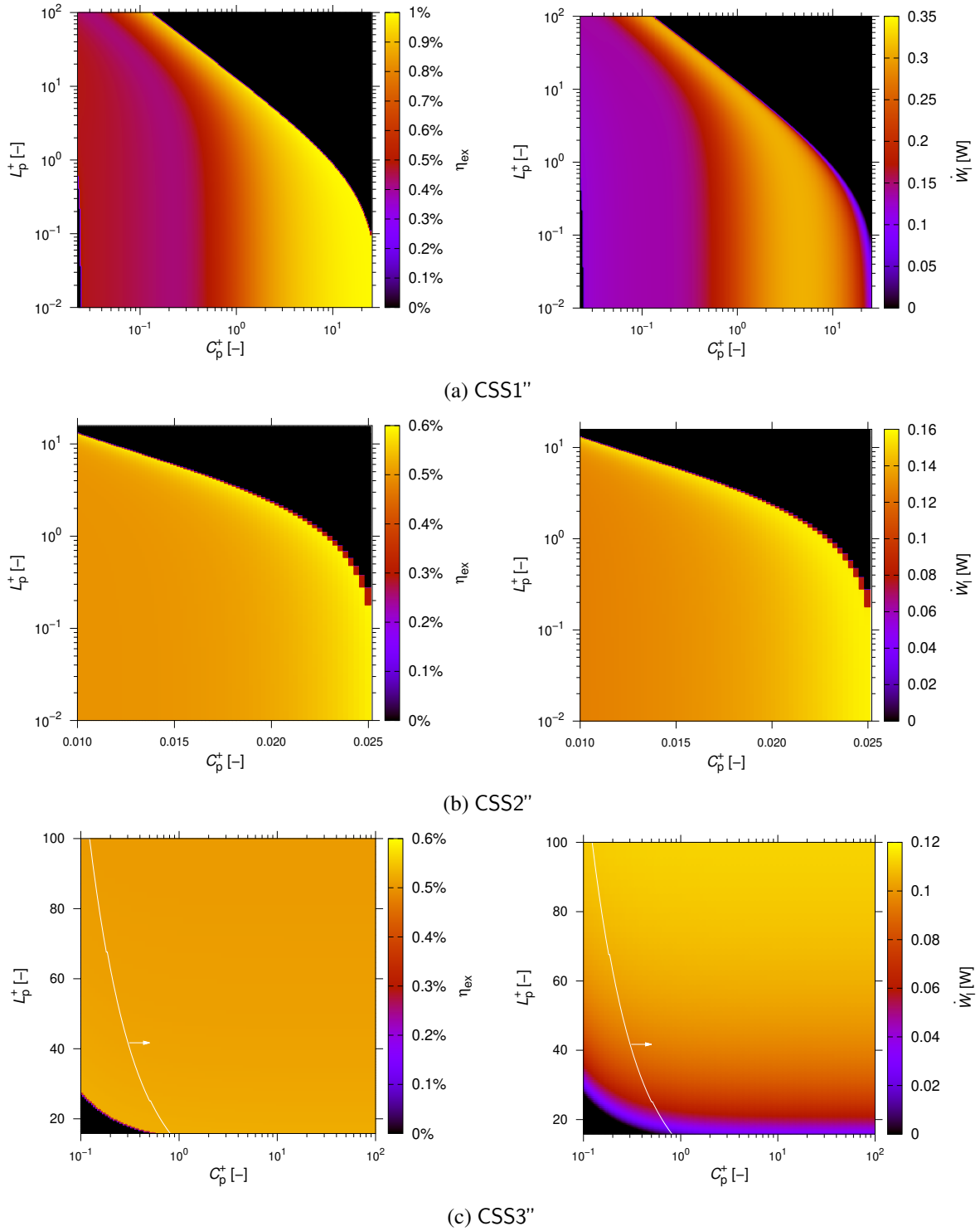


Figure 3.4: Predicted effect of C_p and L_p on NIFTE performance using the NTP-TL model with n-pentane as the working fluid. Three distinct cyclic steady states are detected, denoted by CSS1'', CSS2'', and CSS3'', which are overlapping for certain (C_p, L_p) combinations. The effect on both the exegetic efficiency η_{ex} (left plots) and the average power output $\dot{W}_l$ (right plots) is shown for each cyclic steady-state. C_p^+ and L_p^+ refer to C_p and L_p normalized by their respective nominal values in Table 2.1. The remaining parameters are set to their nominal values. The separatrix between stable and unstable limit cycles are shown in white, with arrows pointing in the direction of stable limit cycles. All of the limit cycles are stable on those plots that do not have a separatrix.

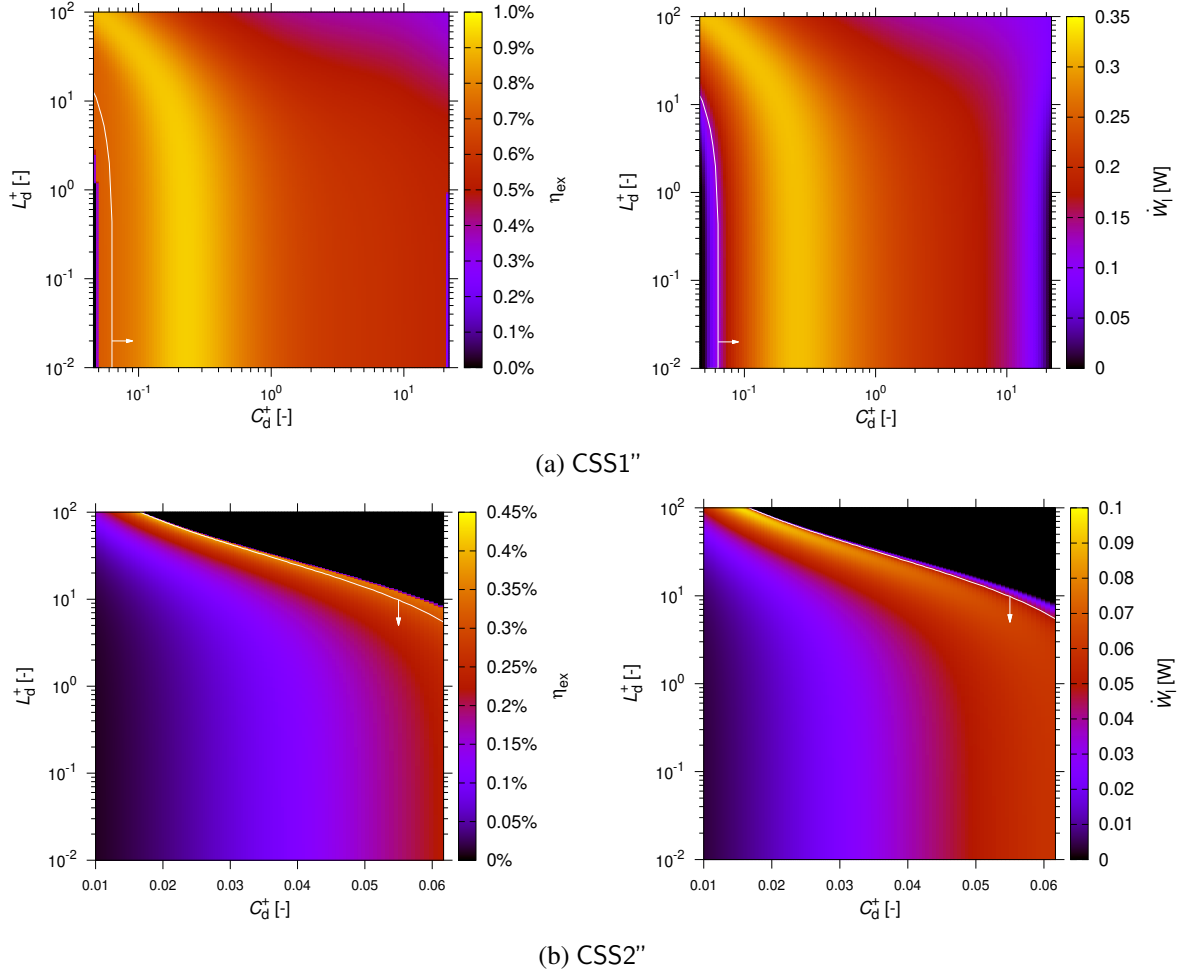


Figure 3.5: Predicted effect of C_d and L_d on NIFTE performance using the NTP-TL model with n-pentane as the working fluid. Two distinct cyclic steady states are detected, denoted by CSS1 and CSS2, which are overlapping for certain (C_d, L_d) combinations. The effect on both the exoegetic efficiency η_{ex} (left plots) and the average power output $\dot{W}_l$ (right plots) is shown for each cyclic steady-state. C_d^+ and L_d^+ refer to C_d and L_d normalized by their respective nominal values in Table 2.1. The remaining parameters are set to their nominal values. The separatrix between stable and unstable limit cycles are shown in white, with arrows pointing in the direction of stable limit cycles.

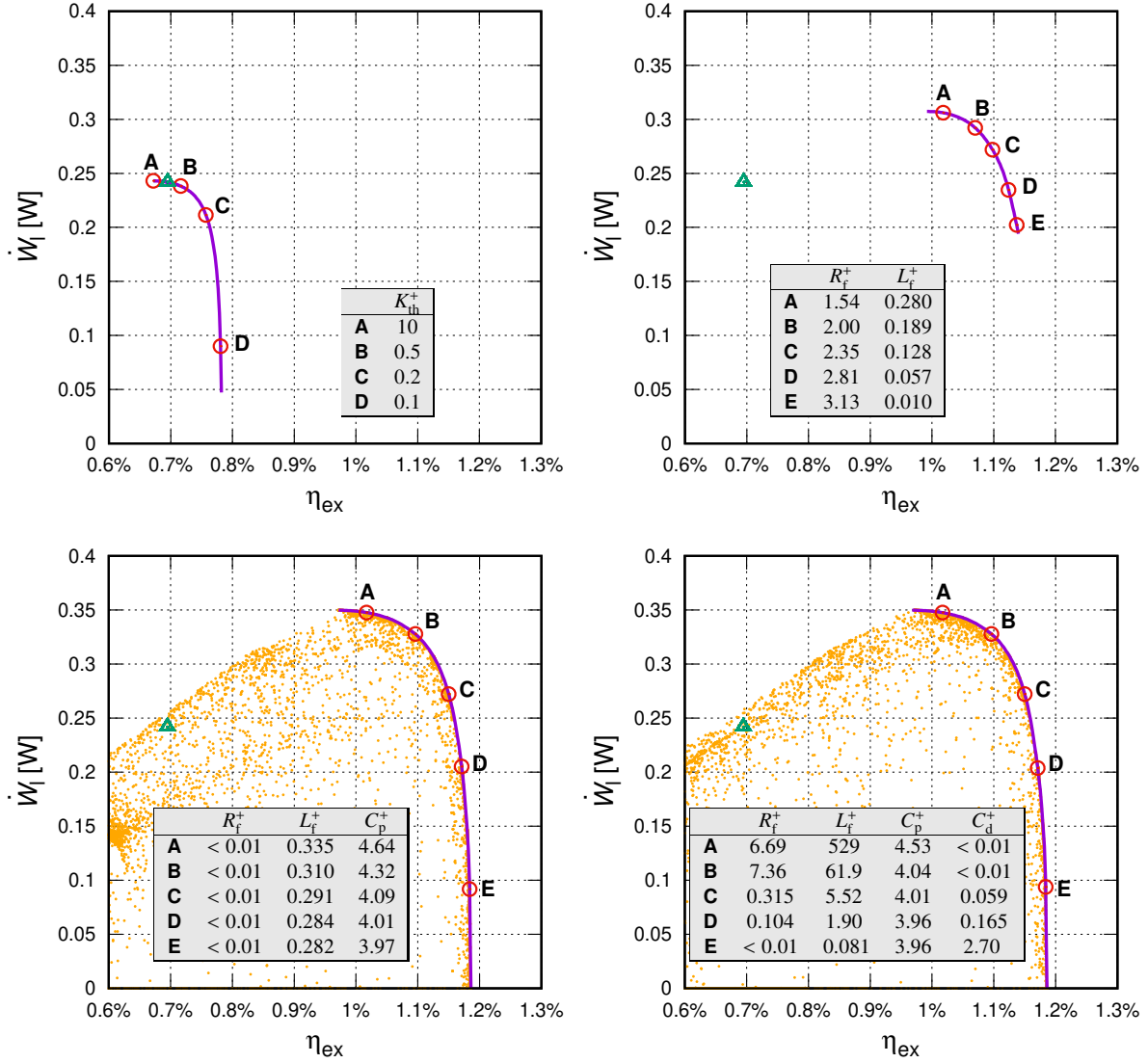


Figure 3.6: Predicted Pareto frontiers for various parameter subsets of the NTP-TL model with n-pentane as the working fluid. The Pareto frontiers are for the exeric efficiency η_{ex} against the average power output $\dot{W}_l$. The subplots correspond to varying the parameters K_{th} (top left), R_f, L_f (top right), R_f, L_f, C_p (bottom left), and R_f, L_f, C_p, C_d (bottom right), with the other parameters kept at their nominal values in Table 2.1. Optimized (normalized) parameter values corresponding to the labeled points on the Pareto frontiers are reported in the tables on each subplot. The reference NIFTE design is marked with a green triangle. The CSS performance at sampled design parameters using Sobol' sequences is displayed with orange dots.

Chapter 4

Background on Set-based Computing

4.1 Introduction

Bounding techniques are all essential components of complete-search methods for global optimization. In dynamic optimization, where a dynamic system is embedded in the constraints, these methods rely on the construction of bounds on the set of all possible dynamic trajectories. Such bounds can be obtained with or without prior discretization of the time horizon depending on whether a sequential or simultaneous approach is used. A key enabler to compute these bounds are set-based computing techniques, which are reviewed in this chapter.

Methods for bounding so-called factorable functions are reviewed in Section 4.2, followed by a summary of methods to compute polyhedral relaxations of factorable programs in Section 4.3. Existing approaches to bounding the solutions of implicit algebraic equations (AEs), which are non-factorable, are discussed in Section 4.4. Lastly, a number of set-propagation methods for bounding the solutions of ordinary differential equations (ODEs) are reviewed in Section 4.5. Existing implementations of all of these methods in the C++ toolkits MC++ [80] and CRONOS [81] are also demonstrated at the end of each section.

Notation In this chapter, uppercase letters are employed to denote intervals for corresponding variables, e.g. X and Y represent sets for variables x and y respectively. $\mathbb{R}$

denote the interval real space. In addition, $[\cdot]$ denotes natural interval extension in interval analysis, and $\llbracket \cdot \rrbracket$ for polynomial model extension in polynomial-based analysis. The midpoint of an interval is denoted by $m(\cdot)$; the width of an interval is denoted by $w(\cdot)$.

4.2 Bounding of factorable functions

Various techniques exist to construct bounds on the image set of functions that are factorable [82, 83]. Functions that belong to this class can be broken down into a finite sequence of binary sums, binary products and univariate compositions. They constitute the majority of the functions that can be represented in a computer and those that are present in optimization problems. For instance, the function $f(\mathbf{x}) = \frac{x-x^2}{\exp(x)}$ is factorable since it can be decomposed via the following sequence (commonly known as *code list* or *computational graph*),

$$\begin{aligned} z_1(x) &= x \\ z_2(x) &= \exp(z_1(x)) \\ z_3(x) &= z_1(x)^2 \\ z_4(x) &= z_1(x) - z_3(x) \\ z_5(x) &= z_4(x)/z_2(x) \\ f(x) &= z_5(x) \end{aligned}$$

A significant advantage of such a representation is that it allows the use of arithmetics for propagating bounds on each elementary operation in the code list whenever bounds are given for the participating variables, thereby leading to an enclosure of the range of the function itself. Three key bounding approaches are interval analysis, convex and concave estimators, and polynomial model estimators, which will be presented in more details in the rest of this section.

4.2.1 Interval Analysis

A closed real interval X is defined as $X = [x^L, x^U] \in \mathbb{IR}$, representing the set $\{x \in \mathbb{R} : x^L \leq x \leq x^U\}$. The midpoint of the interval is given by $m(X) = \frac{x^L + x^U}{2}$, and the width by $w(X) = x^U - x^L$.

The binary operations of interval arithmetic are defined as

$$X \odot Y = \{x \odot y : x \in X, y \in Y\}, \quad (4.2.1)$$

where $\odot$ stands for any of the four binary operations (including $+$, $-$, $\times$ and $\div$). Unary operations are defined in a similar manner,

$$f(X) = \{f(x) : x \in X\}, \quad (4.2.2)$$

where $f(x)$ is a univariate function [84]. For example,

$$X + Y = [x^L + y^L, x^U + y^U] \quad (4.2.3)$$

$$X \cdot Y = [\min(x^L y^L, x^L y^U, x^U y^L, x^U y^U), \max(x^L y^L, x^L y^U, x^U y^L, x^U y^U)]$$

$$\exp(X) = [\exp(x^L), \exp(x^U)]$$

The natural interval extension of a factorable function $f(x)$, denoted by $[f](X)$, can be computed if the bounds of the variable x are provided. A simple example is

$$f(x) = \frac{x - x^2}{\exp(x)}, \quad (4.2.4a)$$

$$x \in [0, 2]. \quad (4.2.4b)$$

The natural interval extension of $f(x)$ is computed and shown in Table 4.1.

Table 4.1: Code list and the natural interval extension of $f(x) = \frac{x-x^2}{\exp(x)}$ with $x \in [0, 2]$

factors	$z_k(x)$	$[z_k]$
$z_1(x)$	x	$[0, 2]$
$z_2(x)$	$\exp(z_1(x))$	$[1, 7.389]$
$z_3(x)$	$z_1(x)^2$	$[0, 4]$
$z_4(x)$	$z_1(x) - z_3(x)$	$[-4, 2]$
$z_5(x)$	$z_4(x)/z_2(x)$	$[-4, 2]$

However, the actual bounds for this function are given by $[-0.271, 0.161]$. A significant overestimation can be easily noticed, and the situation usually gets worse as the factorable functions become more complex and require more interval propagation. Both the *dependency problem* and the *wrapping effect* give rise to the overestimation by interval analysis. The former is caused by the inability of the arithmetic to recognize multiple occurrences of the same variable in the given function, thus the loss of dependency on the original variable [84, 85]. Take $f(x) = x - x^2$ with $x \in [0, 1]$ as an example, the interval arithmetic yields a natural interval extension of $f(x)$, that is $[f(x)] \in [-1, 1]$. On the other hand, the bounds generated using the equivalent form $f(x) = x(1 - x)$ is $[-2, 2]$; while the actual enclosure is obviously $[0, 0]$. The phenomenon of the wrapping effect arises when dealing with interval vector computations. While the solution to an interval vector computation can take any shape, enclosing the solution by interval analysis results in an interval vector, which wraps and overestimates the actual enclosure [84].

4.2.2 Polynomial Model Estimators

A popular alternative to interval analysis entails approximating the function $f(x)$ on $X = [x^L, x^U]$ by a q -th order polynomial $\mathcal{P}_f^q(x)$, along with an interval remainder term $\mathcal{R}_f^q := [r_f^L, r_f^U]$ such that

$$f(x) \in \llbracket f \rrbracket(x) = \mathcal{P}_f^q(x) \oplus \mathcal{R}_f^q. \quad (4.2.5)$$

Note that the complete polynomial model is denoted using $\llbracket \cdot \rrbracket$. By introducing a remainder, the high order terms in the polynomial functions can be truncated in the polynomial approximation of the original function. This construction is depicted in Figure 4.1.

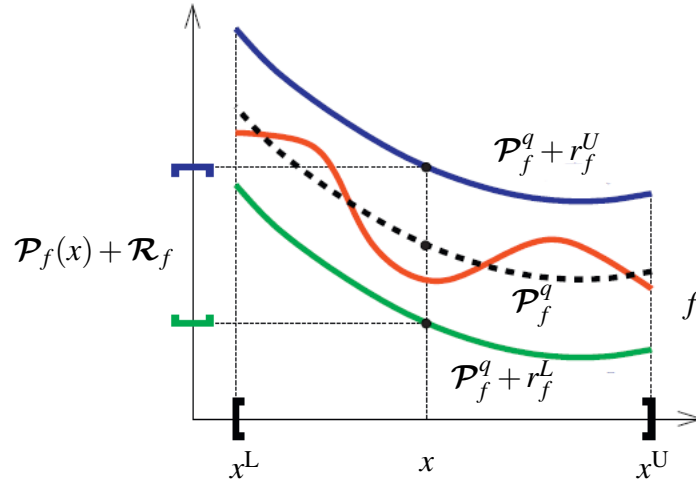


Figure 4.1: Illustration of a polynomial model estimator for a function $f(x)$ on $X = [x^L, x^U]$ [86]

Taylor model [87, 88, 89, 90] and Chebyshev model [91, 92, 93] offer two possible polynomial bases to represent the polynomial term $\mathcal{P}_f^q(\mathbf{x})$. Similar to interval analysis, both Taylor and Chebyshev models define an arithmetic for the binary sum, binary product and univariate composition in factorable functions. In computations where Taylor and Chebyshev models are involved, the polynomial parts that retain parametric dependency are propagated by symbolic manipulations where possible, and the interval remainder terms are handled according to the rules of the interval arithmetic.

In practice, bounds on the resulting polynomial approximant can be computed using interval analysis and added to the remainder term. 4th-order Chebyshev model is applied to approximate (4.2.4). The resulting remainder term $\mathcal{R}_{z_k}(x)$ at each step k is shown in Table 4.2, and compared to the corresponding natural interval extension $[z_k]$. It is clear that using the polynomial model successfully mitigates the wrapping effect and generates significantly tighter dependent bounds than using interval analysis.

Table 4.2: Code list and dependent variable enclosures of $f(x) = \frac{x-x^2}{\exp(x)}$ with $x \in [0, 2]$. Interval analysis is used to generate the natural interval extension, $[z_k]$; and 4th-order Chebyshev model is used to generate the remainder term, $\mathcal{R}_{z_k}(x)$.

factors	$z_k(x)$	$[z_k]$	$\mathcal{R}_{z_k}(x)$
$z_1(x)$	x	$[0, 2]$	$[0, 0]$
$z_2(x)$	$\exp(z_1(x))$	$[1, 7.389]$	$[-1.739\text{E-}3, 1.739\text{E-}3]$
$z_3(x)$	$z_1(x)^2$	$[0, 4]$	$[0, 0]$
$z_4(x)$	$z_1(x) - z_3(x)$	$[-4, 2]$	$[0, 0]$
$z_5(x)$	$z_4(x)/z_2(x)$	$[-4, 2]$	$[-1.552\text{E-}1, 1.552\text{E-}1]$

Other bounding techniques such as convex relaxation or polyhedral relaxations (see below) could also be used to bound the polynomial function.

4.2.3 Other Relaxation Techniques

Multiple techniques exist for the construction of convex or concave relaxations to nonlinear factorable functions. The tightest possible convex underestimator (or concave overestimator) of a function is known as the convex (or concave) envelope [94].

McCormick relaxation technique [95] constructs convex underestimators and concave overestimators for factorable functions by following a set of rules defined for binary operations as well as unary functions. The standard McCormick relaxation technique was formalized in Reference [96] for the bounding of solutions algorithm comprising a finite number of steps. A generalized form was later defined in Reference [97], where the applications were extended to the bounding of solutions to implicit algebraic equations with respect to some parametric dependency. However, the resulting convex relaxations may be nonsmooth, so care must be taken in solving the McCormick relaxation of an optimization problem as conventional gradient-based techniques may no longer be applied [95, 98].

αBB is another commonly used relaxation technique for twice continuously differentiable functions [99, 100, 101, 102]. Convex underestimators are constructed by adding a

non-positive convex quadratic term to the original function. This quadratic term needs to be sufficiently large so as to overpower the non-convexity of the original function. It can be visualized as the quadratic term "pulling down" the function to form a convex underestimator. Concave overestimators can be constructed in a similar manner by adding a non-negative concave quadratic term. This technique may be automated in practice by computing an interval Hessian matrix of the function and bounding its eigenvalues [100, 99].

4.2.4 Implementations in CRONOS

Various relaxation techniques for factorable functions are implemented in the open-source C++ library MC++ [80]. In this section, the use of MC++ to compute an interval enclosure and a polynomial model of the factorable function (4.2.4) is showcased.

4.2.4.1 Interval arithmetic-based relaxations

Basic, non-verified interval bounding of factorable functions is implemented in the header file `interval.hpp`. Third-party verified libraries for interval arithmetics, including PROFIL [103] and FILIB++ [104], are also available in MC++. Listing 4.1 is an example code for generating bounds to problem (4.2.4) using the non-verified interval arithmetic. It is also possible to modify lines 5 and 6 to `#include "mcprofil.hpp"` and `typedef INTERVAL I` respectively so that PROFIL is used instead; and similarly `#include "mcfilib.hpp"` and `typedef filib::interval<double> I` for FILIB++. The output of this script in Listing 4.2 is the interval bound for $f(x)$ with the given domain $x \in [0, 2]$; the measured CPU time is also reported.

```

1 #include <fstream>
2 #include <iomanip>
3 #include <cmath>
4
5 #include "interval.hpp"
6 typedef mc::Interval I;
7

```

```

8  int main()
9  {
10     // Define problem
11     mc::FFGraph DAG;
12     mc::FFVar X( &DAG );
13     mc::FFVar F = (X - pow(X,2)) / exp(X);
14
15     I Xint( 0., 2.);    // domain
16     I Fint;
17
18     // Output bounds
19     DAG.eval( 1, &F, &Fint, 1, &X, &Xint );
20     std::cout << "Domain: " << Xint << std::endl;
21     std::cout << "Bounds: " << Fint << std::endl;
22
23     return 0;
24 }

```

Listing 4.1: Example code for generating interval arithmetic-based bounds using MC++

```

1 Domain: [  0.0000000000000000e+00 :  2.0000000000000000e+00 ]
2 Bounds: [ -4.0000000000000000e+00 :  2.0000000000000000e+00 ]
3
4 CPU TIME (ms)    0.183
5

```

Listing 4.2: Program output for Listing 4.1 in MC++

4.2.4.2 Polynomial model-based relaxation

Taylor and Chebyshev polynomial models are also implemented in MC++, in the header files `tmodel.hpp` and `cmodel.hpp`, respectively. Listing 4.3 shows an example code that constructs a Chebyshev model for the same problem (4.2.4). A Taylor model can be used instead by modifying lines 8 to 10 to `#include "tmodel.hpp"`, `typedef mc::TModel<I> TM`, and `typedef mc::TVar<I> TV` respectively. In Line 17, a polynomial model is initialized by

specifying the number of independent variables (1 here) and order of polynomial expansion (4 here). The output of this program in Listing 4.4 is a set of coefficients for the Chebyshev basis functions, and the remainder bounds, with the given domain $x \in [0, 2]$; the measured CPU time is also reported .

```

1  #include <fstream>
2  #include <iomanip>
3  #include <cmath>
4
5  #include "interval.hpp"
6  typedef mc::Interval I;
7
8  #include "cmodel.hpp"
9  typedef mc::CModel<I> CM;
10 typedef mc::CVar<I> CV;
11
12
13 int main()
14 {
15     // Define Chebyshev model environment
16     const int NX = 1;    // number of independent variables
17     CM mod( NX, 4 );    // initialize 4-th order Chebyshev model
18
19     // Define the problem
20     I IX = I( 0., 2. ); // domain
21     CV X( &mod, 0, IX );
22     CV F = (X - pow(X,2)) / exp(X);
23
24     // Output Chebyshev model
25     std::cout << "Chebyshev model: " << F << std::endl;
26
27     return 0;
28 }
29

```

Listing 4.3: Example code for generating polynomial model-based bounds using MC++

```

1 Chebyshev model of f(x):
2   a0      = -4.8904384398e-02      0
3   a1      = -2.0007930573e-01      1
4   a2      = -6.5825881697e-02      2
5   a3      =  6.7650880588e-02      3
6   a4      = -1.2734715625e-02      4
7   R       = [ -1.55220e-01 :  1.55220e-01 ]
8   B       = [ -5.50415e-01 :  2.50000e-01 ]
9
10  CPU TIME (ms)    0.718
11

```

Listing 4.4: Program output for Listing 4.3 in MC++

The enclosures generated using interval and Chebyshev polynomial model arithmetics are shown in Figure 4.2, along with the exact function. Interval bounds derived from the Chebyshev models are also shown in the same figure. Note that for the Chebyshev model, the blue dashed line denotes the polynomial approximant of order 4, and the solid blue lines represent the complete Chebyshev model with the remainder bounds. The interval enclosure is computed to be $[-4, 2]$ as demonstrated in Table 4.1; and the Chebyshev model gives a remainder term $\mathcal{R}_f \in [-0.16, 0.16]$, as demonstrated in Table 4.2. It is clear that polynomial model yields a much tighter enclosure than interval arithmetic, yet at the cost of extra computational effort as reflected in the measured CPU times – 0.718 ms for the former and 0.183 ms for the latter. This difference, despite being small for such a simple function, is attributed to the greater number of operations required for computing polynomial models.

4.3 Polyhedral relaxations of factorable programs

4.3.1 Polyhedral relaxations

A factorable program is defined as an optimization problem where the cost and constraint functions are all factorable. A nonlinear program (NLP) can be relaxed as by a linear program (LP) by applying the polyhedral relaxation technique, also known as the McCormick

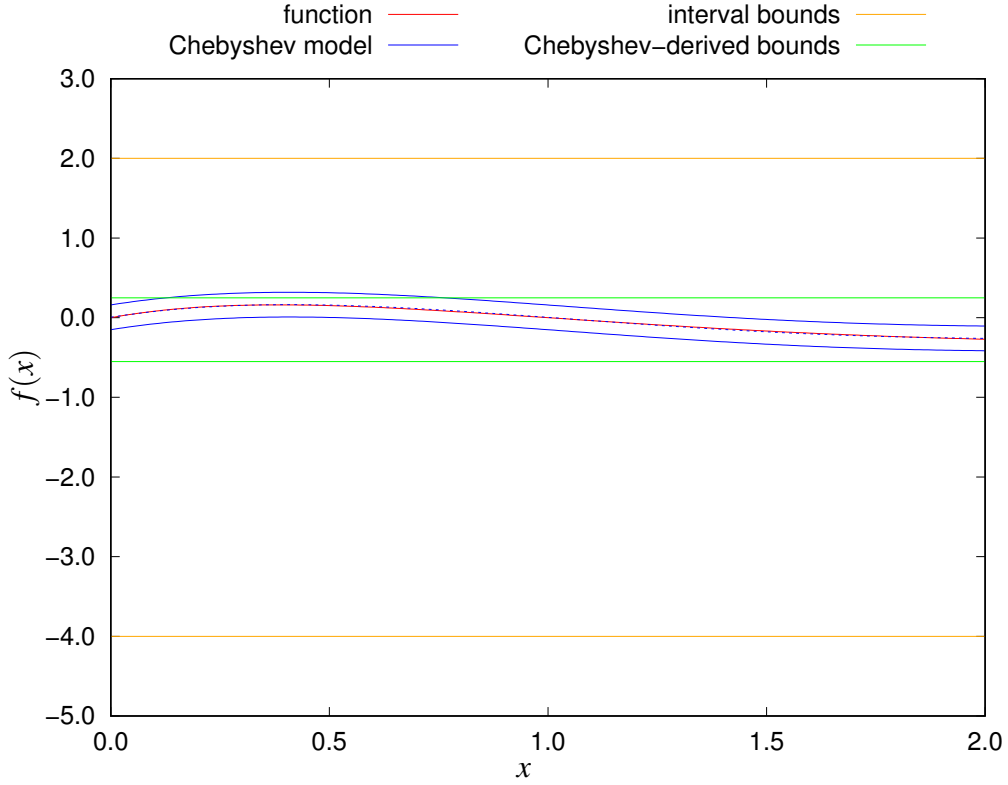


Figure 4.2: Relaxations for $f(x) = \frac{x-x^2}{\exp(x)}$ with $x \in [0, 2]$ using interval arithmetic and Chebyshev polynomial model.

relaxation approach with auxiliary variables. A three-step technique is used to construct polyhedral outer relaxations of a general NLP [105, 106, 107], as summarized below:

Step 1: Decomposition The factorable program undergoes symbolic reformulation to an equivalent form. Auxiliary variables and constraints are introduced to decompose the nonlinear terms. The aim is to create a problem comprising only unary and binary operations in the following form,

$$\begin{aligned} z_k &= z_i \cdot z_j, & i, j < k \\ z_k &= U_k(z_i), & i < k \end{aligned}$$

where $\mathbf{z}$ are the new variables introduced to replace the nonlinear terms and the original variables; and U_k is a univariate function from a given library. The bounds of the new auxiliary variables are computed using interval arithmetic (see Section 4.2.1).

Step 2: Relaxation The nonlinear terms in the new formulation are convexified. The bilinear terms are enclosed using their polyhedral envelopes [108]. The fractional terms can be first converted into the bilinear terms using the inverse function [106] and then convexified. The univariate terms are relaxed based on whether the function U_k is convex, concave or convexo-concave in the variable domain. For example, for a convex function, an affine upper bound is constructed by connecting the two end points of the variable domain; while the convex function itself remains as the nonlinear lower bound. Similar steps are followed for concave functions; and convexo-concave functions require a combination of the two scenarios.

Step 3: Polyhedral outer-approximation Lastly, the nonlinearity that remains in the univariate terms in Step 2 are handled in this step. Linear cuts are created at particular points within the variable domain. These cuts, together with other linear cuts generated in Step 2, form a polyhedral outer-approximation of the nonlinear univariate function. The points at which the cuts are constructed may be automatically selected using the sandwich algorithm; and more cuts are added until the approximation error becomes smaller than a given tolerance [106].

A large number of auxiliary variables and equations may be introduced during the relaxation process. Nevertheless, a variety of robust solvers are available to solve the resulting linear optimization problem, rapidly and reliably; e.g. IBM ILOG CPLEX Optimization Studio (version 12.5.1) [109] and Gurobi Optimizer (version 8.1.1) [110].

4.3.2 Implementation in MC++

An implementation of the polyhedral relaxation technique is available in MC++ for general factorable programs (header file `polimage.hpp`). Listing 4.5 shows an example code for constructing polyhedral relaxations for $\frac{x-x^2}{\exp(x)}$ with $x \in [0, 2]$. The output of this script in Listing 4.6 is the set of cuts generated using polyhedral relaxation for the problem defined. Note that linear operations involved in the decomposed program yield equality constraints. For example, the first equality constraint under the section CUTS represents the definition of the intermediate variable Z2 (using $Z2 := X0 - Z1$) under the section DAG INTERMEDIATES. The relaxations are depicted in Figure 4.3, along with the exact function. The polyhedral cuts

relax (4.2.4) to the bounds denoted by the yellow shaded area.

```

1  #include "interval.hpp"
2  typedef mc::Interval I;
3
4  #include "polimage.hpp"
5  typedef mc::PolImg<I> PI;
6  typedef mc::PolVar<I> PV;
7
8  int main()
9  {
10 // Define the problem
11 mc::FFGraph DAG; // DAG describing the problem
12 const int NX = 1, NF = 1;
13 mc::FFVar X[NX]; // independents
14 X[0].set( &DAG );
15 mc::FFVar F[NF]; // dependents
16 F[0] = ( X[0] - pow(X[0],2) ) / exp(X[0]);
17 I IX[1] = { I(-2,1) }; // domain of independents
18
19 // Create the polyhedral relaxation environment and main variables
20 mc::PolImg<I> Env;
21 mc::PolVar<I> PX[NX];
22 PX[0].set( &Env, X[0], IX[0] );
23
24 // Initialize polyhedral relaxation of the image set
25 mc::PolVar<I> PF[NF];
26 DAG.eval( NF, F, PF, NX, X, PX );
27
28 // Generate and display cuts
29 Env.generate_cuts( NF, PF );
30 std::cout << DAG;
31 std::cout << Env;
32
33 return 0;
34 }

```

35

Listing 4.5: Example code for generating polyhedral relaxations for factorable programs using MC++

```

1  DAG VARIABLES :
2    X0      => { Z0 Z1 Z2 }
3
4  DAG INTERMEDIATES :
5    Z0      <=  EXP( X0 )          => { Z3 }
6    Z1      <=  SQR( X0 )          => { Z2 }
7    Z2      <=  X0 - Z1            => { Z3 }
8    Z3      <=  Z2 / Z0            => { }
9
10 VARIABLES :
11  X0      in [ 0.00000e+00 : 2.00000e+00 ]      (DAG: X0)
12  X3      in [ 1.00000e+00 : 7.38906e+00 ]      (DAG: Z0)
13  X1      in [ 0.00000e+00 : 4.00000e+00 ]      (DAG: Z1)
14  X2      in [ -4.00000e+00 : 2.00000e+00 ]      (DAG: Z2)
15  X4      in [ -4.00000e+00 : 2.00000e+00 ]      (DAG: Z3)
16
17 NO AUXILIARY
18
19 BILINEAR AND FRACTIONAL TERMS:
20  X4
21
22 CUTS :
23  + 1.00000e+00X0 - 1.00000e+00X1 - 1.00000e+00X2 = -0.00000e+00
24  + 1.00000e+00X2 - 1.00000e+00X4 - 2.00000e+00X3 <= -2.00000e+00
25  + 1.00000e+00X2 - 7.38906e+00X4 + 4.00000e+00X3 <= 2.95562e+01
26  + 2.00000e+00X1 - 4.00000e+00X0 <= 0.00000e+00
27  + 2.00000e+00X3 - 6.38906e+00X0 <= 2.00000e+00
28  + 1.00000e+00X2 - 7.38906e+00X4 - 2.00000e+00X3 >= -1.47781e+01
29  + 1.00000e+00X2 - 1.00000e+00X4 + 4.00000e+00X3 >= 4.00000e+00
30  + 1.00000e+00X1 + 0.00000e+00X0 >= 0.00000e+00
31  + 1.00000e+00X1 - 4.00000e+00X0 >= -4.00000e+00

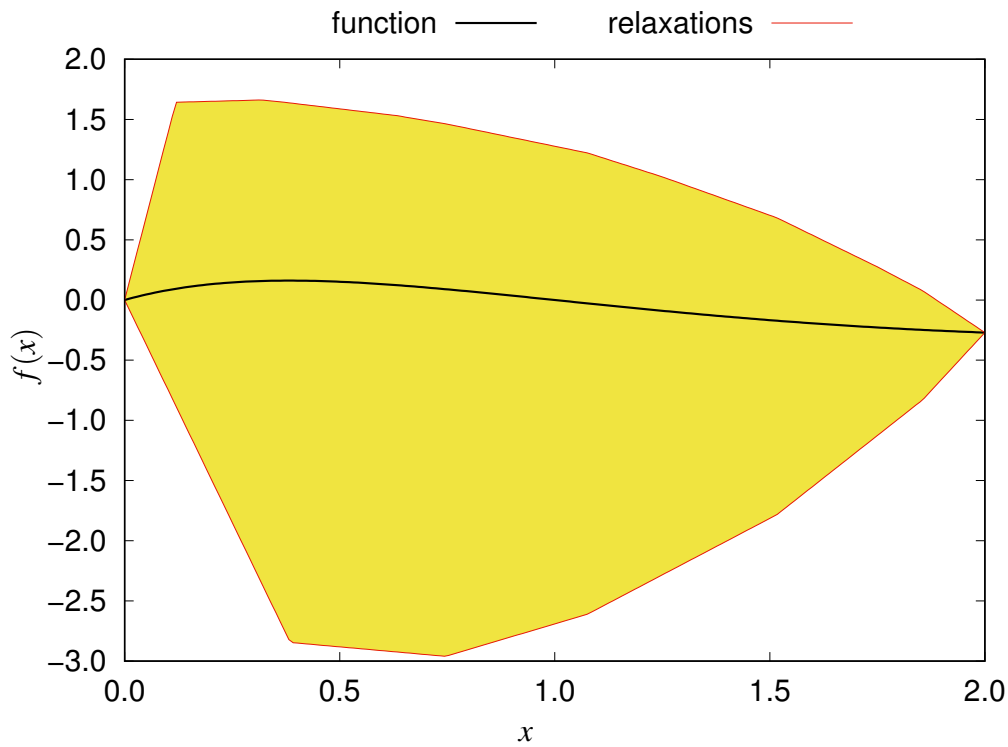
```

```

32 + 1.00000e+00X1 - 2.00000e+00X0 >= -1.00000e+00
33 + 1.00000e+00X1 - 1.00000e+00X0 >= -2.50000e-01
34 + 1.00000e+00X1 - 3.00000e+00X0 >= -2.25000e+00
35 + 1.00000e+00X3 - 1.00000e+00X0 >= 1.00000e+00
36 + 1.00000e+00X3 - 7.38906e+00X0 >= -7.38906e+00
37 + 1.00000e+00X3 - 3.71744e+00X0 >= -1.16369e+00
38 + 1.00000e+00X3 - 2.21715e+00X0 >= 4.51803e-01
39 + 1.00000e+00X3 - 5.44958e+00X0 >= -3.79039e+00
40

```

Listing 4.6: Program output for Listing 4.5 in MC++

Figure 4.3: Polyhedral relaxations for $f(x) = \frac{x-x^2}{\exp(x)}$ for $x \in [0, 2]$

4.4 Bounding the Solutions of Implicit Algebraic Equations

Besides the various relaxation techniques that have been developed for factorable functions, a number of methods are also available for bounding non-factorable functions, including the solutions of differential equations (ODEs) and implicit algebraic equations. Bounding methods

for implicit algebraic equations are first reviewed in this section.

Consider a system of implicit algebraic equations in the form:

$$\mathbf{f}(\mathbf{x}, \mathbf{p}) = 0 \quad (4.4.1)$$

where $\mathbf{p} \in \mathbf{P}$ is a vector of independent variables, with $\mathbf{P} \in \mathbb{IR}^{n_p}$; $\mathbf{x} \in \mathbb{R}^{n_x}$ are the dependent variables; and $\mathbf{f} : \mathbb{R}^{n_x} \times \mathbb{R}^{n_p} \rightarrow \mathbb{R}^{n_x}$ is a vector-valued function. $\mathbf{f}$ is assumed to be continuously differentiable (with $q \geq 1$) in $\mathbf{x}$ and $\mathbf{p}$. Given some interval bounds $\mathbf{P}$ for $\mathbf{p}$, the aim is to describe an interval enclosure of the solution set χ for $\mathbf{x}$, subject to some *a priori* bounds $\mathbf{x} \in \mathbf{X}^0$. This is represented as

$$\chi(\mathbf{p}) := \left\{ \mathbf{x} \in \mathbf{X}^0 \mid \exists \mathbf{p} \in \mathbf{P} : \mathbf{f}(\mathbf{x}, \mathbf{p}) = 0 \right\}. \quad (4.4.2)$$

Notice that the set $\chi(\mathbf{p})$ may only be described by an implicit function if a unique $\mathbf{x} \in \mathbf{X}^0$ solves $\mathbf{f}(\mathbf{x}, \mathbf{p}) = \mathbf{0}$ for each $\mathbf{p} \in \mathbf{P}$. Otherwise the set $\chi(\mathbf{p})$ may contain multiple solution branches, or even no solution.

In principle, iterative Newton-based methods are widely used to calculate an interval enclosure $\mathbf{X}$ such that

$$\mathbf{X} \supseteq \chi(\mathbf{p}), \quad \forall \mathbf{p} \in \mathbf{P};$$

or a polynomial model enclosure:

$$\mathcal{P}_\chi^q(\mathbf{p}) \oplus \mathcal{R}_\chi^q(\mathbf{p}) \supseteq \chi(\mathbf{p}), \quad \forall \mathbf{p} \in \mathbf{P}.$$

Gauss-Seidel Method and Krawczyk Method are two popular variations, for solving both linear and nonlinear equation systems. Similar to bounding of factorable functions, such iterative techniques can propagate both intervals [111, 112, 113] and polynomial model estimators [114, 115]. In addition, a variant of Gaussian elimination can be used to bound the solution of linear algebraic equations in the form $\mathbf{A}(\mathbf{p})\mathbf{x} = \mathbf{b}(\mathbf{p})$.

General iterative methods for both nonlinear and linear systems are first reviewed in Section 4.4.1, followed by the Gaussian elimination method for linear systems in Section 4.4.2.

4.4.1 Iterative methods

Component-wise Gauss-Seidel and Krawczyk methods were initially developed to compute interval bounds on the solutions sets of an implicit algebraic equations, whether it is linear or nonlinear [111, 112, 113]. Given an *a priori* interval vector $\mathbf{X}^0 \in \mathbb{IR}^{n_x}$, both methods construct a nested sequence of interval vectors $\{\mathbf{X}^k\}_{k \in \mathbb{N}}$ containing the solution set (4.4.2), where N is the number of iterations. The key idea of such iterative methods is to apply mean-value theorem at a chosen reference point $\hat{\mathbf{x}}^k \in \mathbf{X}^k$ for each iteration k :

$$\forall \mathbf{p} \in \mathbf{P}, \exists \eta_i \in [0, 1] : f_i(\mathbf{p}, \hat{\mathbf{x}}^k) + \frac{\partial f_i}{\partial \mathbf{x}}(\mathbf{p}, \hat{\mathbf{x}}^k + \eta_i(\mathbf{x} - \hat{\mathbf{x}}^k)) \cdot (\mathbf{x} - \hat{\mathbf{x}}^k) = 0$$

for each $i = 1, \dots, n_x$. In practice, $\hat{\mathbf{x}}^k$ is often picked as the midpoint of $\mathbf{X}^k$ [112]. This way an update can be performed such that:

$$X_i^{k+1} \leftarrow \hat{x}_i^k - \left[\frac{\partial f_i}{\partial x_i} \right]^{-1}(\mathbf{P}, \mathbf{X}^k) \left[[f_i](\mathbf{p}, \hat{\mathbf{x}}^k) + \sum_{j \neq i} \left[\frac{\partial f_i}{\partial x_j} \right](\mathbf{P}, \mathbf{X}^k) \cdot (X_j^k - \hat{x}_j^k) \right]$$

The interval arithmetic-based Gauss-Seidel method shown in Algorithm 2 is a refinement of this idea.

A good preconditioning matrix $\mathbf{Y} \in \mathbb{R}^{n_x \times n_x}$ is furthermore critical. It is typically chosen as the inverse of the midpoint of the Jacobian matrix:

$$\mathbf{Y}^k := \left[m \left(\left[\frac{\partial \mathbf{f}}{\partial \mathbf{x}} \right](\mathbf{P}, \mathbf{X}^k) \right) \right]^{-1} \quad (4.4.3)$$

Alternative choices of preconditioners for interval Newton methods have been developed for different scenarios, for instance preconditioners based on the solutions of linear programming problems [113, 116].

Note that in the Gauss-Seidel method, the preconditioned Jacobian diagonal elements $[G]_{i,i}^k$ may contain zero in range. Otherwise, an alternative is to use the parametric Krawczyk method which does not require division by an interval, but nevertheless generates looser bounds on the solution sets [112, 117]. To represent the component-wise Krawczyk method, Algorithm 2 is

Algorithm 2 Interval-Based Gauss-Seidel Method

Requires: Implicit function $\mathbf{f} : \mathbb{R}^{n_x} \times \mathbb{R}^{n_p} \rightarrow \mathbb{R}^{n_x}$; Initial bounds $(\mathbf{X}^0, \mathbf{P}) \in \mathbb{IR}^{n_x} \times \mathbb{IR}^{n_p}$; Maximum number of iterations $\text{MAXIT} \in \mathbb{N}_{++}$; Termination tolerance $\text{TOL} > 0$

Main Loop: For each $k = 1, \dots, \text{MAXIT}$,

1. $\hat{\mathbf{x}}^k := \text{mid}(\mathbf{X}^k); \mathbf{Y}^k := \left(\text{mid} \left(\left[\frac{\partial \mathbf{f}}{\partial \mathbf{x}} \right] (\mathbf{P}, \mathbf{X}^k) \right) \right)^{-1}$
2. $[c]^k := \mathbf{Y}^k \cdot [\mathbf{f}] (\mathbf{P}, \hat{\mathbf{x}}^k);$
3. $[\mathbf{G}]^k := \mathbf{Y}^k \cdot \left[\frac{\partial \mathbf{f}}{\partial \mathbf{x}} \right] (\mathbf{P}, \mathbf{X}^k);$

For each $i \in 1, \dots, n_x$:

4. $[\Gamma]_i^{k+1} := \hat{x}_i^k - \frac{1}{[\mathbf{G}]_{i,i}^k} \left[[c]_i^k + \sum_{j=1}^{i-1} [\mathbf{G}]_{i,j}^k \cdot (X_j^{k+1} - \hat{x}_j^{k+1}) + \sum_{j=i+1}^{n_x} [\mathbf{G}]_{i,j}^k \cdot (X_j^k - \hat{x}_j^k) \right],$
5. $X_i^{k+1} := X_i^k \cap [\Gamma]_i^{k+1},$
6. **If** $w(\mathbf{X}^k) - w(\mathbf{X}^{k+1}) \leq \text{TOL}$, **break iteration.**

Returns: An interval vector $\mathbf{X} \in \mathbb{IR}^{n_x}$ such that

$$\mathbf{X}^k \supseteq \{\mathbf{x} \in \mathbf{X}^0 \mid \exists \mathbf{p} \in \mathbf{P} : \mathbf{f}(\mathbf{p}, \mathbf{x}) = 0\}$$

slightly modified. Specifically, Step 4 in the **Main Loop** is replaced by the following steps:

$$[\mathbf{H}]^k := \mathbf{I} - [\mathbf{G}]^k$$

$$[K]_i^{k+1} := \hat{x}_i^k - \left[[c]_i^k + \sum_{j=1}^{i-1} [\mathbf{H}]_{i,j}^k \cdot (X_j^{k+1} - \hat{x}_j^{k+1}) + \sum_{j=i+1}^{n_x} [\mathbf{H}]_{i,j}^k \cdot (X_j^k - \hat{x}_j^k) \right],$$

for each $i = 1, \dots, n_x$; where $\mathbf{I}$ is the identity matrix of size n_x .

Instead of propagating intervals, one can use polynomial model as the arithmetic inside these iterative techniques, with the advantage of better mitigating the wrapping effect than interval arithmetic [114, 115]. The objective is to construct a sequence of q -th order polynomial model enclosures $\{\llbracket \chi \rrbracket(\cdot)\}_{k \in \mathbb{N}}$ such that:

$$\llbracket \chi \rrbracket^k(\mathbf{p}) =: \mathcal{P}_\chi^k(\mathbf{p}) \oplus \mathcal{R}_\chi^k \supseteq \chi(\mathbf{p}), \quad \forall \mathbf{p} \in \mathbf{P}.$$

It should be noted that to construct a q -th order polynomial model, $\mathbf{f}(\mathbf{x}, \mathbf{p})$ must be at least q -times continuously differentiable in $\mathbf{x}$ and $\mathbf{p}$.

The polynomial-version of the iterative Gauss-Seidel method is presented in Algorithm 3. The reference point in the mean-value theorem is now chosen to be $\mathcal{P}_{\mathbf{x}}^k$, compared to the midpoint of $\mathbf{X}^k$ in Algorithm 2. Then, a typical preconditioning matrix $\mathbf{Y}^k$ is the inverse of the midpoint of an interval enclosure of:

$$\left[\left[\frac{\partial \mathbf{f}}{\partial \mathbf{x}} \right] \left(\mathcal{P}_{\mathbf{x}}^k(\mathbf{p}) + [0, 1] \mathcal{R}_{\mathbf{x}}^k, \mathbf{p} \right) \right].$$

Some conservative initial bounds $[0, 1] \mathcal{R}_{\mathbf{x}}^k$ are introduced to the polynomial representation because the functions $\mathbf{f}$ can be nonlinear. Notice that the last intersection step in the polynomial model-based is different from that in the interval-based Gauss-Seidel methods. Unlike in Step 5 in Algorithm 2 where the intersection of two intervals yields another interval, intersection between polynomial models of the current and the previous iterations do not simply yield another polynomial model. In Step 5 in Algorithm 3, the polynomial part of the current iteration $\mathcal{P}_{\Gamma_i}^{k+1}$ is retained because it better approximates the actual solution than $\mathcal{P}_{\Gamma_i}^k$. The remainder part, however, is computed by subtracting $\mathcal{P}_{\Gamma_i}^{k+1}$ from both $\llbracket \mathcal{X} \rrbracket_{\Gamma_i}^{k+1}$ and $\llbracket \mathcal{X} \rrbracket_{\Gamma_i}^k$, and intersecting the remainders. As a result, the difference between the two polynomial parts $[\mathcal{P}_{\Gamma_i}^k - \mathcal{P}_{\Gamma_i}^{k+1}]$ is an interval extension and hence introduces some overestimation. Therefore, the polynomial model of the current iteration $\llbracket \mathcal{X} \rrbracket_i^{k+1}$ is not guaranteed to be nested in that of the previous iteration $\llbracket \mathcal{X} \rrbracket_i^k$.

Algorithm 3 Polynomial Model-Based Gauss-Seidel Iterative Algorithm

Requires: Implicit function $\mathbf{f} : \mathbb{R}^{n_x} \times \mathbb{R}^{n_p} \rightarrow \mathbb{R}^{n_x}$; Initial bounds $(\mathbf{X}^0, \mathbf{P}) \subset \mathbb{IR}^{n_x} \times \mathbb{IR}^{n_p}$; Maximum number of iterations $\text{MAXIT} \in \mathbb{N}_{++}$; Termination tolerance $\text{TOL} > 0$; Polynomial order $q \in \mathbb{N}_{++}$

Initialization: Set $\mathcal{P}_\chi^0 = m(\mathbf{X}^0)$, $\mathcal{R}_\chi^0 = \frac{1}{2}w(\mathbf{X}^0) \cdot [-1, 1]$

Main Loop: For each $k \in \{1, \dots, \text{MAXIT}\}$, **compute:**

1. $\mathbf{Y}^k = \left[m \left(\left[\frac{\partial \mathbf{f}}{\partial \mathbf{x}} \right] (\mathcal{P}_\chi^k(\mathbf{p}) + [0, 1]\mathcal{R}_\chi^k, \mathbf{p}) \right) \right]^{-1}$
 2. $\llbracket \mathbf{c} \rrbracket^k(\mathbf{p}) := \mathbf{Y}^k \cdot \llbracket \mathbf{f} \rrbracket (\mathcal{P}_\chi^k(\mathbf{p}), \mathbf{p})$
 3. $\llbracket \mathbf{G} \rrbracket^k(\mathbf{p}) := \mathbf{Y}^k \cdot \left[\frac{\partial \mathbf{f}}{\partial \mathbf{x}} \right] (\mathcal{P}_\chi^k(\mathbf{p}) + [0, 1]\mathcal{R}_\chi^k, \mathbf{p})$
- For each $i = 1, \dots, n_x$:
4. $\llbracket \chi \rrbracket_{\Gamma_i}^{k+1}(\mathbf{p}) := \mathcal{P}_{\chi_i}^k(\mathbf{p}) - \frac{1}{\llbracket \mathbf{G} \rrbracket_{i,i}^k(\mathbf{p})} \left[\llbracket \mathbf{c} \rrbracket_i^k(\mathbf{p}) + \sum_{j=1}^{i-1} \llbracket \mathbf{G} \rrbracket_{i,j}^k \cdot \left(\llbracket \chi \rrbracket_j^{k+1} - \mathcal{P}_{\chi_j}^{k+1} \right)(\mathbf{p}) + \sum_{j=i+1}^{n_x} \llbracket \mathbf{G} \rrbracket_{i,j}^k(\mathbf{p}) \cdot \mathcal{R}_{\chi_j}^k \right],$
 5. $\llbracket \chi \rrbracket_i^{k+1}(\mathbf{p}) := \left(\mathcal{P}_{\Gamma_i}^{k+1}(\mathbf{p}), \mathcal{R}_{\Gamma_i}^{k+1} \cap \left[\mathcal{P}_{\chi_i}^k - \mathcal{P}_{\Gamma_i}^{k+1} \right](\mathbf{p}) + \mathcal{R}_{\chi_i}^k \right),$
 6. **If** $w(\mathcal{R}_\chi^k) - w(\mathcal{R}_\chi^{k+1}) \leq \text{TOL}$, **break iteration.**

Returns: A q -th order polynomial model $\llbracket \chi \rrbracket^k = (\mathcal{P}_\chi(\cdot), \mathcal{R}_\chi)$ satisfying

$$\mathcal{P}_\chi(\mathbf{p}) \oplus \mathcal{R}_\chi \supseteq \chi(\mathbf{p}), \quad \forall \mathbf{p} \in \mathbf{P}$$

The interval Krawczyk method extension to polynomial model arithmetic works analogously. To represent this method, Step 4 in **Main Loop** in Algorithm 3 is replaced with the following steps,

$$\llbracket \mathbf{H} \rrbracket^k(\mathbf{p}) := \mathbf{I} - \llbracket \mathbf{G} \rrbracket^k(\mathbf{p}) \tag{4.4.5a}$$

$$\llbracket \Gamma \rrbracket_i^{k+1}(\mathbf{p}) := \mathcal{P}_{\chi_i}^k(\mathbf{p}) - \left[\llbracket \mathbf{c} \rrbracket_i^k(\mathbf{p}) + \sum_{j=1}^{i-1} \llbracket \mathbf{H} \rrbracket_{i,j}^k \cdot \left(\llbracket \chi \rrbracket_j^k - \mathcal{P}_{\chi_j}^{q,k} \right)(\mathbf{p}) + \sum_{j=i+1}^{n_x} \llbracket \mathbf{H} \rrbracket_{i,j}^k \cdot \mathcal{R}_{\chi_j}^k \right], \tag{4.4.5b}$$

for each $i = 1, \dots, n_x$.

Initialization of the polynomial model iterations may be from any interval enclosure; e.g.,

the *a priori* enclosure $\mathbf{X}^0$, or the convergence interval enclosure $\mathbf{X}^\infty$ from the interval-based iterative methods. Using the latter is a two-phase approach that warm-starts the polynomial arithmetic. Using centered polynomial models allows further simplification of the methods.

In the case of linear algebraic equations, notice that Algorithms 2 and 3 may be simplified by moving the construction of the preconditioning matrix $\mathbf{Y}$, the preconditioned matrix $\mathbf{G}$ and preconditioned right-hand-side vector to outside of the main loop (i.e. as part of the initialization).

4.4.2 Gaussian elimination methods

The special case of a linear algebraic system in the parameters $\mathbf{p} \in \mathbf{P}$ is considered, in the form:

$$\mathbf{A}(\mathbf{p})\mathbf{x} = \mathbf{b}(\mathbf{p}), \quad (4.4.6)$$

where $\mathbf{A}(\mathbf{p}) \in \mathbb{R}^{n_x \times n_x}$ is a matrix of coefficients, and $\mathbf{b}(\mathbf{p}) \in \mathbb{R}^{n_x}$ is a vector of constants.

Interval Gaussian elimination [111, 112] can be applied to bound the solution set of such parametric systems. The real-valued operations in classical Gaussian elimination are simply replaced by the corresponding operations in interval arithmetic. Similar to the iteration methods in Section 4.4.1, it is generally advisable to precondition the linear system to reduce the dependency problem that may cause significant overestimation [112]. The most commonly used preconditioning matrix, $\mathbf{Y} \in \mathbb{R}^{n_x \times n_x}$, is the inverse of the midpoint of matrix $\mathbf{A}$, i.e. $\mathbf{Y} = m([\mathbf{A}](\mathbf{P}))^{-1}$ [118, 119]. The solution set to the preconditioned system $\mathbf{Y} \cdot \mathbf{Ax} = \mathbf{Y} \cdot \mathbf{b}$ contains the solution to the original system; yet is generally enlarged [120]. The new interval matrixes $[\mathbf{G}] \in \mathbb{R}^{n_x \times n_x}$ and $[\mathbf{c}] \in \mathbb{R}^{n_x}$ are defined as follows,

$$[\mathbf{G}] := \mathbf{Y} \cdot [\mathbf{A}], \quad (4.4.7a)$$

$$[\mathbf{c}] := \mathbf{Y} \cdot [\mathbf{b}], \quad (4.4.7b)$$

By analogy to Gaussian elimination, an augmented matrix of the preconditioned system is

formed as,

$$\left[\begin{array}{cccc|c} [G]_{1,1} & [G]_{1,2} & \cdots & [G]_{1,n_x} & [c]_1 \\ [G]_{2,1} & [G]_{2,2} & \cdots & [G]_{2,n_x} & [c]_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ [G]_{n_x,1} & [G]_{n_x,2} & \cdots & [G]_{n_x,n_x} & [c]_{n_x} \end{array} \right] \quad (4.4.8)$$

A sequence of elementary row operations, known as *pivoting*, is applied to the rows of the augmented matrix to transform its left hand side into upper triangular form, specifically a row echelon form,

$$\left[\begin{array}{cccc|c} [G']_{1,1} & [G']_{1,2} & \cdots & [G']_{1,n_x} & [c']_1 \\ & [G']_{2,2} & \cdots & [G']_{2,n_x} & [c']_2 \\ & & \ddots & \vdots & \vdots \\ & & & [G']_{n_x,n_x} & [c']_{n_x} \end{array} \right] \quad (4.4.9)$$

where $[G']$ and $[c']$ denote the transformed matrixes of $[G]$ and $[c]$ respectively. In this way, the preconditioned system can be solved via backward substitution using interval arithmetic. In short, problem (4.4.9) is solved from the bottom row up the upper triangular matrix, and interval bounds for subsequent terms are propagated by substituting solved terms into the current subproblem. Generally, the interval bound for state variable x_i , where $i = 1, \dots, n_x$, is calculated by

$$X_i = \frac{[c']_i - \sum_{j=i+1}^{n_x} [G']_{i,j} X_j}{[G']_{i,i}}. \quad (4.4.10)$$

An alternative is to transform problem (4.4.8) into a lower triangular form; and then apply forward substitution to obtain the solution sets.

It is important to note that no iteration is required in Gaussian elimination; unlike in Newton-based methods. Given an interval for $\mathbf{A}$, the preconditioner $\mathbf{Y}$, the interval matrix $[G]$ and vector $[c]$ are calculated as part of the initialization. In general, Gaussian elimination is more efficient than the iterative methods for small-scale systems.

A polynomial model counterpart of interval Gaussian elimination [115] is readily obtained by considering polynomial models $\llbracket \mathbf{A} \rrbracket(\mathbf{p})$ and $\llbracket \mathbf{b} \rrbracket(\mathbf{p})$ of the parametric matrix $\mathbf{A}(\mathbf{p})$ and

vector $\mathbf{b}(\mathbf{p})$. Then the preconditioning, forward elimination and backward substitution proceed in polynomial model arithmetic:

$$\llbracket X \rrbracket_i(\mathbf{p}) = \frac{\llbracket c' \rrbracket_i(\mathbf{p}) - \sum_{j=i+1}^{n_x} \llbracket G' \rrbracket_{i,j}(\mathbf{p}) \cdot \llbracket X \rrbracket_j(\mathbf{p})}{\llbracket G' \rrbracket_{i,i}(\mathbf{p})}. \quad (4.4.11)$$

The advantage of interval or polynomial model Gaussian elimination is that it does not require an *a priori* enclosure of the solution set. However, this approach fails when 0 is in range of the diagonal entries, $[G']_{i,i}$ or $\llbracket G' \rrbracket_{i,i}(\mathbf{p})$. One way of preventing this is by applying partial pivoting, similar in essence to standard Gaussian elimination. But the bounding will fail if a singular matrix is contained in the parametric matrix $\mathbf{A}(\mathbf{p})$ for some values of $\mathbf{p} \in \mathbf{P}$.

4.4.3 Implementations in CRONOS

An implementation of the aforementioned bounding methods for algebraic equation systems is available in the C++ library CRONOS in the class AEBND (defined in the header file `aebnd.hpp`). An example code snippet is shown in Listing 4.7. The goal is to enclose solutions to the scalar implicit equation:

$$\begin{aligned} 0 &= (p+3)x_1 + px_2 - 3p \\ 0 &= px_1 + (p+3)x_2 \\ 0 &= (p+3)x_3 - x_2 \end{aligned} \quad (4.4.12)$$

with $p \in [-1, 1]$, and the *a priori* state bound $x_1 \in [-10, 10]$, $x_2 \in [-10, 10]$ and $x_3 \in [-20, 30]$. The problem is specified in Lines 11 to 24, including the number of parameters in Line 11, the number of dependent state variables in Line 12, the system dynamics in Lines 19 to 21, the parameter bounds in Line 23 and *a priori* state bounds in Line 24. The polynomial environment is set up in Lines 27 to 29, and 5th-order Chebyshev model is used. The bounding method is specified to be the Gauss-Seidel method in Line 38. Alternative options include KRAW for the Krawczyk method; and GE for Gaussian elimination. The preconditioner is generated using the inverse midpoint, as declared in Line 39. The state enclosures are computed using interval-based in Line 42 and polynomial model-based estimators in Line 43. Notice that the same *a priori* state bounds $\mathbf{I}x_0$ are used in both versions. The program outputs in Listing 4.8 are

the final state bounds computed using interval-based and polynomial model-based methods. It is clear that the 5th-order Chebyshev model-based method generates tighter bounds than the interval-based method for all three dependent variables. Moreover, the former method takes slightly more iterations before convergence than the latter method .

```

1  #include "aebnd.hpp"
2  #include "interval.hpp"
3  typedef mc::Interval I;
4  #include "cmodel.hpp"
5  typedef mc::CModel<I> PM;
6  typedef mc::CVar<I> PV;
7
8  int main()
9  {
10     mc::FFGraph NLE;  // DAG describing the problem
11     const unsigned NP = 1;  // Parameter dimension
12     const unsigned NX = 3;  // State dimension
13     mc::FFVar P[NP];  // Parameters p
14     for( unsigned i=0; i<NP; i++ ) P[i].set( &NLE );
15     mc::FFVar X[NX], SOL[NX];  // Dependents x
16     for( unsigned i=0; i<NX; i++ ) X[i].set( &NLE );
17
18     mc::FFVar F[NX];  // Equations f(x,p)=0
19     F[0] = (P[0]+3.)*X[0] + P[0]*X[1] - 3.*P[0];
20     F[1] = P[0]*X[0] + (P[0]+3.)*X[1];
21     F[2] = (P[0]+3.)*X[2] - X[1];
22
23     I Ip[NP] = { I(-1e0,1e0) };  // parameter bounds
24     I Ix0[NX] = { I(-1e1,1e1), I(-1e1,1e1), I(-2e1,2e1) };  // a
        priori dependent bounds
25     I Ix[NX];
26
27     PM PMEnv( NP, 5 );  // Initialize polynomial environment
28     PV PMp[NP], PMx[NX];
29     for( unsigned i=0; i<NP; i++ ) PMp[i].set( &PMEnv, i, Ip[i] );
30
31     mc::AEBND<I,PM,PV> BND;

```

```

32  BND.set_dag( &NLE );
33  BND.set_var( NP, P );
34  BND.set_dep( NX, X, F );
35  BND.options.INTERBND = true;
36  BND.options.MAXIT    = 20;
37  BND.options.RTOL     = BND.options.ATOL     = 1e-8;
38  BND.options.BOUNDER  = mc::AEBND<I,PM,PV>::Options::ALGORITHM::GS
    ;
39  BND.options.PRECOND  = mc::AEBND<I,PM,PV>::Options::
    PRECONDITIONING::INVMD;
40  BND.setup();
41
42  BND.solve( Ip, Ix, Ix0 );    // Compute interval bounds
43  BND.solve( PMp, PMx, Ix0 ); // Compute polynomial bounds
44
45  return 0;
46 }
47

```

Listing 4.7: Example code for computing polynomial model-based bounds for algebraic equations using CRONOS

```

1  Final Bounds using interval-based bounding method:
2  X1 = [ -2.00000e+00 : 2.00000e+00 ]
3  X2 = [ -1.00000e+00 : 1.00000e+00 ]
4  X3 = [ -5.00000e-01 : 5.00000e-01 ]
5
6  No MAX ITERATIONS 16
7  CPU TIME (SEC)    0.00000
8
9  Final Bounds using polynomial model-based bounding method:
10 X1 =
11  a0    = -2.56211e-01    0
12  a1    = 1.26863e+00    1
13  a2    = -2.93478e-01    2
14  a3    = 1.11801e-01    3
15  a4    = -4.19255e-02    4

```

```

16  a5      =  1.39752e-02      5
17  R       =  [ -1.39752e-02 :  1.39752e-02 ]
18  B       =  [ -2.00000e+00 :  9.00621e-01 ]
19
20  X2 =
21  a0      = -2.56211e-01      0
22  a1      =  2.68634e-01      1
23  a2      = -2.93478e-01      2
24  a3      =  1.11801e-01      3
25  a4      = -4.19255e-02      4
26  a5      =  1.39752e-02      5
27  R       =  [ -1.39752e-02 :  1.39752e-02 ]
28  B       =  [ -1.00000e+00 :  2.49681e-01 ]
29
30  X3 =
31  a0      = -1.10147e-01      0
32  a1      =  1.48461e-01      1
33  a2      = -1.33206e-01      2
34  a3      =  6.38204e-02      3
35  a4      = -2.61137e-02      4
36  a5      =  9.01066e-03      5
37  R       =  [ -9.24024e-03 :  9.24024e-03 ]
38  B       =  [ -5.00000e-01 :  1.51927e-01 ]
39
40  No MAX ITERATIONS  17
41  CPU TIME (SEC)     0.00400
42
43

```

Listing 4.8: Program output for Listing 4.7 in CRONOS

4.5 Bounding the solutions of ordinary differential equations

Another important class of non-factorable functions are the solutions of nonlinear differential equation. Consider an ODE system, in the form:

$$\dot{\mathbf{x}}(t, \mathbf{p}) = \mathbf{f}(\mathbf{x}(t, \mathbf{p}), \mathbf{p}), \quad (4.5.1)$$

where $\mathbf{x}(t, \mathbf{p}) \in \mathbb{R}^{n_x}$ is a vector of dependent (state) variables; $\mathbf{p} \in \mathbf{P} \in \mathbb{R}^{n_p}$ is a vector of parameters; and $\mathbf{f}: \mathbb{R}^{n_x} \times \mathbb{R}^{n_p} \rightarrow \mathbb{R}^{n_x}$ is a vector-valued function. The dependent variables have initial conditions $\mathbf{x}(0, \mathbf{p}) = \mathbf{x}_0(\mathbf{p})$, at $t = 0$. The objective is to enclose the reachable set at a given time $T > 0$:

$$\mathbf{X}(T, \mathbf{p}) = \left\{ \xi \in \mathbb{R}^{n_x} \left| \begin{array}{l} \exists \mathbf{x}: [0, T] \rightarrow \mathbb{R}^{n_x}, \exists \mathbf{p} \in \mathbf{P} \\ \dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{p}); \mathbf{x}(0) = \mathbf{x}_0 \\ \xi = \mathbf{x}(T) \end{array} \right. \right\} \quad (4.5.2)$$

4.5.1 Review of set-propagation approaches

Set-propagation is the major approach towards enclosing solutions to ODEs. There are two main variants of this approach: (1) a continuous-time approach; (2) and a discrete-time approach.

4.5.1.1 Continuous-time set propagation approaches

The key idea behind continuous-time set propagation is the construction of an auxiliary set of ODEs the solution of which parameterizes an enclosure of the reachable set (4.5.2). Commonly used enclosures comprise intervals that are constructed using the standard theory of differential inequalities [121], ellipsoids using ellipsoidal calculus [122], and more recently, polynomial models (either Taylor model or Chebyshev model) with convex remainders using a generalized theory of differential inequalities [123, 43].

Polynomial models have been proven to be effective at handling nonlinear systems, although they can incur a significant computational overhead compared with interval- or ellipsoidal-based approaches. This approach will be reviewed in this subsection. The objective is to find a q -th order polynomial enclosure for the reachable set, such that

$$\chi(T, \mathbf{P}) \in \llbracket \mathbf{X} \rrbracket(T, \mathbf{p}) := \mathcal{P}_{\mathbf{x}}^q(T, \mathbf{p}) \oplus \mathcal{R}_{\mathbf{x}}^q(T) \quad \forall \mathbf{p} \in \mathbf{P}$$

The polynomial term $\mathcal{P}_{\mathbf{x}}^q(T, \mathbf{p})$ approximates the parametric state $\mathbf{X}(T, \mathbf{p})$, and is computed by propagating the coefficients of the polynomial function basis through the ODEs. For example, using a Taylor model gives a polynomial part that matches the Taylor series expansion of the ODEs at time T . The remainder term $\mathcal{R}_{\mathbf{x}}^q(T)$ can be defined in the form of an interval vector $\mathcal{R}_{\mathbf{x}}^q(T) := [r_{\mathbf{x}}^L(T), r_{\mathbf{x}}^U(T)]$, where

$$\dot{r}_i^L(T) = \min_{\xi, \rho} \left\{ f_i(\mathcal{P}_{\mathbf{x}}^q(T, \mathbf{p}) + \xi, \rho) - \dot{\mathcal{P}}_{x_i}^q(T, \rho) \left| \begin{array}{l} \xi_i = r_{x_i}^L(T) \\ \xi \in [r_{\mathbf{x}}^L(T), r_{\mathbf{x}}^U(T)] \\ \rho \in \mathbf{P} \end{array} \right. \right\} \quad (4.5.3a)$$

$$\dot{r}_i^U(T) = \max_{\xi, \rho} \left\{ f_i(\mathcal{P}_{\mathbf{x}}^q(T, \mathbf{p}) + \xi, \rho) - \dot{\mathcal{P}}_{x_i}^q(T, \rho) \left| \begin{array}{l} \xi_i = r_{x_i}^U(T) \\ \xi \in [r_{\mathbf{x}}^L(T), r_{\mathbf{x}}^U(T)] \\ \rho \in \mathbf{P} \end{array} \right. \right\} \quad (4.5.3b)$$

for each $i = 1, \dots, n_x$. $\mathcal{R}_{\mathbf{x}}^q(T)$ is then determined by integrating the system of ODEs in Equations (4.5.3a-4.5.3b). As a result, computing a polynomial model $\mathcal{P}_{\mathbf{x}}^q(t, \mathbf{p}) \oplus [r_{\mathbf{x}}^L(t), r_{\mathbf{x}}^U(t)]$ requires solving an auxiliary ODE system of size $\mathcal{O}(n_x n_p^q)$.

The remainder term can alternatively be chosen to be an ellipsoidal remainder,

$$\mathcal{R}_{\mathbf{x}}^q(t) = \mathcal{E}(Q_{\mathbf{x}}^q(t)) := \left\{ Q_{\mathbf{x}}^q(t)^{\frac{1}{2}} \mathbf{v} \mid \exists \mathbf{v} \in \mathbb{R}^{n_x} : \mathbf{v}^T \mathbf{v} \leq 1 \right\},$$

which is less likely to suffer from the wrapping effect. The following ODE describes the ellipsoidal bounder,

$$\dot{Q}_{\mathbf{x}}^q(t) = A(t)Q_{\mathbf{x}}^q(t) + Q_{\mathbf{x}}^q(t)A(t)^T + \sum_{i=1}^{n_x} \kappa_i(t)Q_{\mathbf{x}}^q(t) + \text{diag}(\kappa(t))^{-1} \text{diag rad} \left(\Omega_f^q[Q_{\mathbf{x}}^q(t)] \right)^2, \quad (4.5.4)$$

where

$$A(t) := \left(\frac{\partial f}{\partial x} (\mathcal{P}_x^q(t, \hat{p}), \hat{p}) \right).$$

The nonlinearity bounder $\Omega_f^q[Q] \in \mathbb{R}^{n_x}$ at any given time t must satisfy the following condition,

$$f(\mathcal{P}_x^q(t, \rho) + r, \rho) - \dot{\mathcal{P}}_x^q(t, \rho) - \frac{\partial f}{\partial x}(\mathcal{P}_x^q(t, \rho), \rho) r \in \Omega_f^q[Q] \quad \forall (r, \rho) \in \mathcal{E}(Q) \times P,$$

and it can be constructed automatically via interval analysis. κ is a scaling function that can be chosen e.g. to minimize $\text{tr}(Q_x^q(t))$. Hence, the resulting auxiliary ODE system for $\{\mathcal{P}_x^q(t, \mathbf{p}) \mid \mathbf{p} \in \mathbf{P}\} \oplus \mathcal{E}(Q_x^q(t))$ has a size of $\mathcal{O}(n_x n_p^q + n_x^2)$. The reader is referred to the works of Villeneuve et al. [123, 43] for further details.

4.5.1.2 Discrete-time set-propagation approaches

By contrast, discrete-time set-propagation proceeds by first discretizing the time horizon into a finite number of steps, then propagating a chosen type of enclosure from one step to the next along the time horizon. Certain discrete-time methods keep track of the discretization error in order to guarantee a rigorous enclosure [124], while other methods simply discard the discretization errors [125]. Examples of enclosures include intervals [126], polynomial models with interval remainders [86], polynomial models with ellipsoidal remainders [127, 124], and zonotopes [128, 129].

More recently, the continuous-time theory of differential inequalities was extended to discrete-time set propagation using the explicit Euler scheme for bounding the solutions to ODEs [130].

There have been a number of toolboxes where these continuous-time or discrete-time methods are implemented, including ACADO [131] where a discrete-time approach with polynomial models is implemented; CORA [132, 133, 134] which follows the same approach and uses zonotopes and polynomial zonotopes for set propagation; and the library CRONOS [81] where both continuous-time and discrete-time methods with various types of enclosures are implemented. These tools represent state-of-the-art methods in computing validated solutions for initial value problems (IVPs) in ODEs. Further methods exist to bound the solution of differential algebraic equations [135, 136, 137], but these are beyond the scope of this thesis.

4.5.2 Implementations in CRONOS

An implementation of set-propagation methods for ODEs is available in CRONOS in the class ODEBND (defined in the header file `odebnd_sundials.hpp`). The auxiliary set of ODEs which parameterize a selected form of enclosure are constructed automatically and integrated using state-of-the-art solver CVODES in SUNDIALS [71]. The forms of enclosures available in ODEBND include intervals, ellipsoids, and (Taylor and Chebyshev) polynomial models with interval ellipsoidal remainders.

An example code snippet is shown in Listing 4.9. The objective is to enclose the set of solutions to the parametric ODEs:

$$\dot{x}_1(t) = p_1 x_1(t) (1 - x_2(t)), \quad (4.5.5a)$$

$$\dot{x}_2(t) = p_2 x_2(t) (x_1(t) - 1), \quad (4.5.5b)$$

over $t \in [0, 25]$ with $x_1(0) = 1.2$, $x_2(0) = 1.1$, $p_1 \in [2.99, 3.01]$ and $p_2 \in [0.99, 1.01]$. The problem is defined in Lines 12 to 24, including the number of parameters in Line 12, the number of dependent variables in Line 13, the system dynamics in Lines 20 and 21, and the initial conditions in Lines 23 and 24. The time horizon is defined to be $t = [0, 10]$ in Line 28. The parameter bounds are given in Line 36 and 37. Line 38 computes the state bounds using interval-based estimators, and Line 45 computes the state bounds using polynomial-based estimators. A 5th-order Chebyshev model is used as specified in Line 41. The program outputs in Listing 4.10 are the final state bounds computed using interval-based estimators and polynomial model-based estimators. The required CPU time in seconds is also reported. Figure 4.4 shows the reachable set enclosures for the dependent variables using: inner-approximation constructed (INNER) via computing an extensive number of uncertainty scenarios in the given initial condition range; continuous-time set-propagation with interval-based methods (DINEQI); and continuous-time set-propagation with 5th-order Chebyshev model-based estimators. Notice that in this small-scale case study, DINEQPM is as computationally efficient as DINEQI, while the former generates considerably tighter enclosures than the latter.

```

1 #include <fstream>
2 #include "odebnd_sundials.hpp"
3 #include "interval.hpp"
4 typedef mc::Interval I;
5 #include "cmodel.hpp"
6 typedef mc::CModel<I> PM;
```

```

7  typedef mc::CVar<I>    PV;
8
9  int main()
10 {
11     mc::FFGraph IVP;
12     const unsigned int NP = 2;  // Parameter dimension
13     const unsigned int NX = 2;  // State dimension
14     mc::FFVar P[NP];  // Parameter array
15     for( unsigned int i=0; i<NP; i++ ) P[i].set( &IVP );
16     mc::FFVar X[NX];  // State array
17     for( unsigned int i=0; i<NX; i++ ) X[i].set( &IVP );
18
19     mc::FFVar RHS[NX];  // Right-hand side function
20     RHS[0] = P[0] * X[0] * ( 1. - X[1] );
21     RHS[1] = P[1] * X[1] * ( X[0] - 1. );
22     mc::FFVar IC[NX];  // Initial conditions
23     IC[0] = 1.2;
24     IC[1] = 1.1;
25
26     mc::ODEBND_SUNDIALS<I,PM,PV> LV;
27     LV.set_dag( &IVP );
28     LV.set_time( 0., 10. );
29     LV.set_state( NX, X );
30     LV.set_parameter( NP, P );
31     LV.set_differential( NX, RHS );
32     LV.set_initial( NX, IC );
33
34     // Compute Interval Bounds
35     I Ip[NP];
36     Ip[0] = I(2.99,3.01);
37     Ip[1] = I(0.99,1.01);
38     LV.bounds( Ip );
39
40     // Compute Polynomial Bounds
41     PM PMenv( NP, 5 );  // polynomial model environment
42     PV PMp[NP];
43     PMp[0] = PV( &PMenv, 0, Ip[0] );
44     PMp[1] = PV( &PMenv, 1, Ip[1] );

```

```

45  LV.bounds( PMp );
46
47  return 0;
48 }
49

```

Listing 4.9: Example code for generating polynomial model-based bounds for ODE-IVPs using set-propagation methods implemented in CRONOS

```

1  Using interval-based estimators:
2  @t = 0.0000e+00 :
3  x[0] = [ 1.20000e+00 : 1.20000e+00 ]
4  x[1] = [ 1.10000e+00 : 1.10000e+00 ]
5  @t = 1.0000e+01 :
6  x[0] = [ 8.26407e-01 : 1.45626e+00 ]
7  x[1] = [ 7.39714e-01 : 1.04102e+00 ]
8  No STEPS      623
9  No EVALATIONS  RHS: 988  JAC: 0
10 CPU TIME (SEC)      0.03542
11
12 Using polynomial model-based estimators:
13 @t = 0.0000e+00 :
14 x[0] =
15  a0      = 1.20000e+00
16  R      = [ 0.00000e+00 : 0.00000e+00 ]
17  B      = [ 1.20000e+00 : 1.20000e+00 ]
18
19 x[1] =
20  a0      = 1.10000e+00
21  R      = [ 0.00000e+00 : 0.00000e+00 ]
22  B      = [ 1.10000e+00 : 1.10000e+00 ]
23
24 @t = 1.0000e+01 :
25 x[0] =
26  a0      = 1.14131e+00      0  0
27  a1      = 9.23081e-02      0  1
28  a2      = 3.22281e-02      1  0

```

```

29  a3      = -5.87633e-03      0  2
30  a4      = -5.83567e-03      1  1
31  a5      = -5.99372e-04      2  0
32  a6      = -8.99799e-04      0  3
33  a7      = -2.12942e-03      1  2
34  a8      = -7.42731e-04      2  1
35  a9      = -3.86943e-05      3  0
36  a10     =  7.06545e-06      0  4
37  a11     = -3.82515e-05      1  3
38  a12     = -3.44004e-05      2  2
39  a13     = -6.63558e-06      3  1
40  a14     = -3.99933e-09      4  0
41  a15     =  4.16266e-06      0  5
42  a16     =  1.50221e-05      1  4
43  a17     =  9.95384e-06      2  3
44  a18     =  3.37557e-06      3  2
45  a19     =  6.05756e-07      4  1
46  a20     =  2.14770e-08      5  0
47  R       = [ -2.42565e-05 :  2.42565e-05 ]
48  B       = [  1.00051e+00 :  1.26916e+00 ]
49
50  x[1] =
51  a0      =  8.90325e-01      0  0
52  a1      =  3.00286e-02      0  1
53  a2      =  1.15187e-02      1  0
54  a3      =  5.44265e-03      0  2
55  a4      =  7.82784e-03      1  1
56  a5      =  5.89594e-04      2  0
57  a6      = -1.16199e-04      0  3
58  a7      = -1.00927e-04      1  2
59  a8      = -5.39482e-05      2  1
60  a9      = -7.96477e-06      3  0
61  a10     = -2.03587e-05      0  4
62  a11     = -6.28750e-05      1  3
63  a12     = -3.35859e-05      2  2
64  a13     = -7.69517e-06      3  1
65  a14     = -2.84633e-07      4  0
66  a15     = -7.74972e-07      0  5

```

```

67  a16  = -4.10758e-06      1  4
68  a17  = -3.37074e-06      2  3
69  a18  = -1.15138e-06      3  2
70  a19  = -1.59510e-07      4  1
71  a20  = -2.85733e-09      5  0
72  R    = [ -1.33314e-05 :  1.33314e-05 ]
73  B    = [  8.46555e-01 :  9.46159e-01 ]
74
75  No STEPS      295
76  No EVALATIONS  RHS: 539   JAC: 0
77  CPU TIME (SEC)    0.02962
78

```

Listing 4.10: Program output for Listing 4.9 in CRONOS

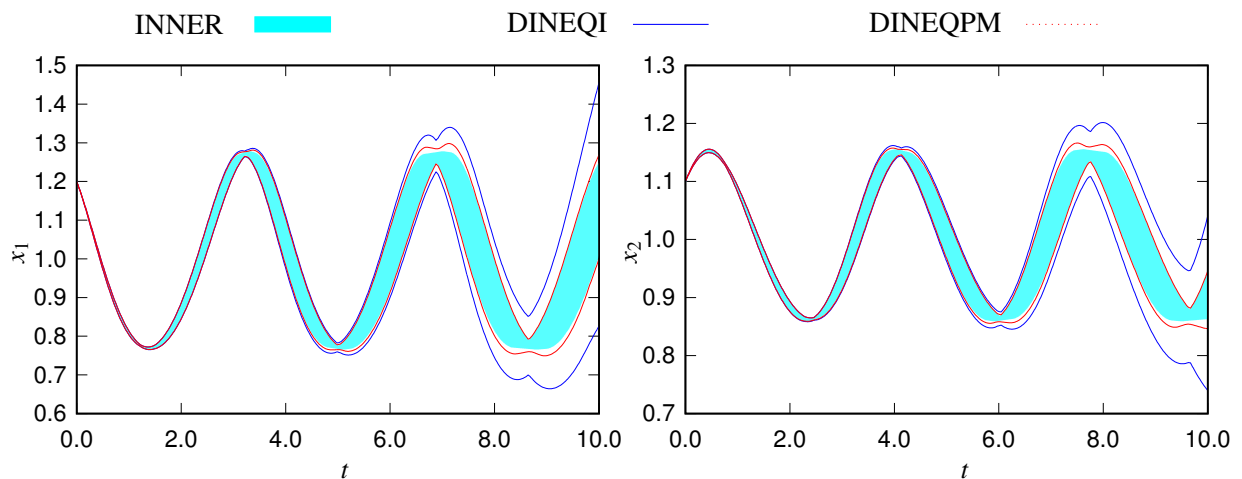


Figure 4.4: Comparison of reachable set enclosures to (4.5.5) with $x_1(0) = 1.2, x_2(0) = 1.2, p_1 \in [2.99, 3.01]$ and $p_2 \in [0.99, 1.01]$ using: inner approximation (INNER); continuous set-valued integration with interval-based estimators (DINEQI) and continuous set-valued integration with 5th-order Chebyshev model-based estimators (DINEQPM).

4.6 Summary

In this chapter, some of the most popular bounding techniques for factorable functions are reviewed, as well as procedures for bounding the solutions of both algebraic and differential equation systems.

These methods form the foundation to the contributions in the upcoming chapters, in which the aim is to develop new bounding approaches for dynamic systems that will enhance their global optimization. Most of research in the field has been focused on set-propagation techniques. By contrast, discretization methods based on ODE discretization have not been studied as extensively. Therefore, the investigation will mainly focus on such discretization methods for ODE bounding in the upcoming chapters.

Chapter 5

Discretization and Block Decomposition Approach to Bounding of Nonlinear ODEs

5.1 Introduction

Computing tight enclosures on the reachable set of nonlinear ordinary equations (ODEs) is important in many applications, including global optimization and robust control as well as uncertainty analysis. Thermofluidic engines, in particular the NIFTE device that is the focus of this work, are highly nonlinear dynamic systems that can exhibit multiple limit cycles. Effective bounding of the state trajectories of such engines is critical to analyzing the effect of modeling uncertainties and design parameters. This sensitivity analysis capability is in turn essential to safety verification, process control and optimization.

Chapter 4 has reviewed existing techniques that are applicable to the bounding of both algebraic and differential equations. The focus in this chapter is on the full discretization of an ODE system using classical schemes such as Runge-Kutta formula.

The result of this discretization is a large system of algebraic equations. Since these systems of equations are highly structured, block decomposition is attractive in order to break the system into smaller subsystems which can be solved sequentially, leading to significant reduction in computational complexity. How to effectively perform such decompositions, in particular regarding the preconditioning of the subsystems, constitute the main novelty of the work in this chapter.

The chapter is outlined as follows. In Section 5.2, existing discretization-based approaches to ODE bounding are reviewed and basic background material on Runge-Kutta methods is summarized. The

methodology section starts by presenting the basic decomposition approach to bounding the discretized system of algebraic equations in Section 5.3, and illustrates its limitations on a simple example. A more sophisticated decomposition approach based on recursive block preconditioning is developed in Section 5.3.2. The implementation of the proposed decomposition strategies in the toolkit CRONOS is described in Section 5.3.3. Two numerical case studies are then presented in Sections 5.4 and 5.5. The modified bounding techniques are first applied to a two-reaction batch reactor model in Section 5.4, and then to the larger-scale NTP-TL model of a NIFTE device in Section 5.5. The enclosures and the computational efficiencies are evaluated and compared against those generated using state-of-the-art set-valued integration tools. Finally, in Section 5.6 the main findings of this chapter are summarized.

5.2 Background

In order to bound the solutions of initial value problems (IVPs) with parametric uncertainty, an alternative to the set-valued integration methods reviewed in Section 4.5 entails the full discretization of the ODEs and computation of bounds. This approach requires an effective and efficient method for bounding the solution of large-scale algebraic equation systems. Existing bounding techniques to both linear and nonlinear algebraic systems were reviewed in Section 4.4 based on either interval arithmetic or polynomial model arithmetic. Convex and concave relaxations of implicit functions can be constructed in an analogous manner by using McCormick relaxation arithmetic [138]. More recently, Wilhelm et al. [125] applied their approach to bound the solutions of discretized stiff ODEs. They considered implicit high-order discretization schemes and exploited the special structure of the ODE residual in a single block decomposition approach in order to reduce the computational time. A similar approach was used by Mitsos et al. [96] to discretize a set of ODEs using an explicit Euler scheme.

The approach developed in this chapter is also similar in essence. The focus is on explicit Runge-Kutta discretization schemes in order to retain linearity in the block subsystems. However, instead of applying a standard block decomposition with a diagonal preconditioning block matrix as in References [125, 96], which can lead to significant overestimation due to wrapping effect, a novel recursive block preconditioning strategy is to be presented with its benefits demonstrated.

5.2.1 Review on discretization methods for differential equations

Consider a system of ordinary differential equations (ODEs)

$$\dot{\mathbf{x}}(t, \mathbf{p}) = \mathbf{f}(\mathbf{x}(t, \mathbf{p}), \mathbf{p}), \quad (5.2.1)$$

where $\mathbf{x}(t, \mathbf{p}) \in \mathbb{R}^{n_x}$ is a vector of dependent (state) variables; $\mathbf{p} \in \mathbf{P} \in \mathbb{R}^{n_p}$ is a vector of parameters; and $\mathbf{f}: \mathbb{R}^{n_x} \times \mathbb{R}^{n_p} \rightarrow \mathbb{R}^{n_x}$ is a vector-valued function. A large number of numerical methods can be used to solve this initial value problem, which rely on the discretization of the time horizon [139, 140, 141]. The focus throughout this chapter is on the popular Runge-Kutta methods.

Given a time grid $t_k := k \cdot h$ for a time step $h > 0$ and $k = 0, \dots, N$ steps, a general Runge-Kutta formula with $S \geq 1$ stages per step approximates the state $\mathbf{x}(t_k, \mathbf{p}) \approx \mathbf{x}_k(\mathbf{p})$ as follows:

$$\begin{aligned} \mathbf{x}_k(\mathbf{p}) &= \mathbf{x}_{k-1}(\mathbf{p}) + h \sum_{i=1}^S b_i \varphi_i(\mathbf{p}) \\ \text{with: } \varphi_i(\mathbf{p}) &= f\left(\mathbf{x}_{k-1}(\mathbf{p}) + h \sum_{j=1}^S a_{ij} \varphi_j(\mathbf{p}), \mathbf{p}\right) \end{aligned} \quad (5.2.2)$$

where the coefficients a_{ij} and b_i make a so-called Butcher's tableau as shown in Table 5.1.

Table 5.1: The general Butcher's tableau for explicit Runge-Kutta methods

c_1	$a_{1,1}$	$a_{1,2}$	$\cdots$	$a_{1,N}$
c_2	$a_{2,1}$	$a_{2,2}$	$\cdots$	$a_{2,N}$
$\vdots$	$\vdots$	$\vdots$		$\vdots$
c_N	$a_{N,1}$	$a_{N,2}$	$\cdots$	$a_{N,N}$
	b_1	b_2	$\cdots$	b_N

In an explicit Runge-Kutta scheme, the matrix $\mathbf{a}$ has a lower triangular form with a zero diagonal, i.e. $a_{i,j} = 0$ for $i \leq j$. This implies that discretized states $x_k(\mathbf{p})$ are explicit functions of the state $x_{k-1}(\mathbf{p})$ at the previous step. By contrast, in an implicit Runge-Kutta method the matrix $\mathbf{a}$ may be full and the formula (5.2.2) becomes an implicit function of $x_k(\mathbf{p})$. Implicit methods are thus more difficult to implement and to solve than explicit methods. Nevertheless, they are able to handle stiffer ODEs and ensure better numerical stability in real-valued numerical integration.

For a p -th order Runge-Kutta method, the local truncation error is in the order of $\mathcal{O}(h^{p+1})$, and the global accumulated error is in the order of $\mathcal{O}(h^p)$. The most famous members of the Runge-Kutta family are the forward and backward Euler methods, and the classical fourth-order Runge-Kutta method. The Butcher's tableaus for these three methods are shown in Table 5.2(a) – (c) respectively. The corresponding local truncation errors are of orders $\mathcal{O}(h^2)$ and $\mathcal{O}(h^5)$; and the global errors are $\mathcal{O}(h)$ and $\mathcal{O}(h^4)$.

Adopting smaller steps reduces approximation errors effectively, and is beneficial to numerical stability. On the other hand, it enlarges the size of the discretized system, hence requires more computational effort.

Table 5.2: The Butcher's tableau for: (a) the forward Euler method; (b) the backward Euler method; and (c) the classical 4th-order Runge-Kutta method

(a)		(b)		(c)				
0	0	1	1	0	0	0	0	0
1		1		1/2	1/2	0	0	0
				1/2	0	1/2	0	0
				1	0	0	1	0
				1/6	1/3	1/3	1/6	

5.3 Methodology: Block decomposition strategy

The discretization of ODEs, e.g. using Runge-Kutta schemes, leads to large-scale systems of algebraic equations. Existing bounding techniques, such as the one reviewed earlier in Section 4.4, may be applied to enclose the range of discretized state. However, the computational effort may be significant given the large number of variables and equations. Instead, the focus herein is on exploiting the special block structure of the algebraic equation systems.

Building on the development of the various bounding techniques for algebraic equations shown in Algorithms 2 and 3, the focus of this chapter is on block decomposition which is essential to reduce the computational burden.

Block decomposition has been applied in many engineering applications. For example, it is known as tearing or sequential-modular approach in process flowsheet simulation and optimization [142]. Efficient algorithms doing triangular block decomposition are available, e.g. Tarjans' algorithm [143]; bordered block triangular decomposition, including MC13, MC33 and MC66 in the Harwell Scientific Library [144]. Recently, Baharev and co-workers [145, 146] presented a block decomposition-based approach that is capable of approximating all well-separated solutions in a large-scale nonlinear system of equation systems. By breaking a large system of equations into smaller blocks and introducing a so-called backsolve step, the computational burden may be reduced considerably while maintaining

numerical stability. Applying such block decomposition in combination with set-arithmetic to bound the solution set therefore appears promising.

The key idea is to re-write a large-scale algebraic system into lower-triangular form via permutation of its variables and equations, and apply existing bounding technique recursively to each block. Specifically, consider an algebraic system $\mathbf{f}(\mathbf{p}, \mathbf{x}) = \mathbf{0}$ that can be decomposed into B blocks given by

$$\mathbf{F}_1(\mathbf{p}, \mathbf{x}_1) = 0, \quad (5.3.1)$$

$$\mathbf{F}_2(\mathbf{p}, \mathbf{x}_1, \mathbf{x}_2) = 0, \quad (5.3.2)$$

$$\vdots$$

$$\mathbf{F}_B(\mathbf{p}, \mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_B) = 0; \quad (5.3.3)$$

where the dependent variables $\mathbf{x}$ are reordered into subvectors as $\mathbf{x} = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_B\}$ and $\mathbf{x}_s \in \mathbb{R}^{n_{x_s}}$ are the dependent variables in each block $s = 1, \dots, B$. The bounding technique, Algorithm 2 or Algorithm 3 of Section 4.4, can then be readily applied to solve each subsystem, $\mathbf{F}_1 = 0, \dots, \mathbf{F}_B = 0$ sequentially.

5.3.1 Block-diagonal preconditioning

A critical aspect of bounding methods for algebraic equation systems is preconditioning in order to mitigate the wrapping effect and compute tighter enclosure. In a block decomposition strategy, the simplest choice for the preconditioning matrix is a block-diagonal matrix:

$$\mathbf{Y}^k = \begin{pmatrix} \mathbf{Y}_1^k & & \\ & \ddots & \\ & & \mathbf{Y}_B^k \end{pmatrix}. \quad (5.3.4)$$

Each preconditioning block $\mathbf{Y}_s$ can be chosen, e.g. as the inverse of the midpoint of the interval Jacobian matrix, in an analogous manner to Equation 4.4.3:

$$\mathbf{Y}_s^k = \left[m \left(\left[\frac{\partial \mathbf{f}_s}{\partial \mathbf{x}_s} \right] (\mathbf{P}, \mathbf{X}_1^\infty, \dots, \mathbf{X}_{s-1}^\infty, \mathbf{X}_s^k) \right) \right]^{-1}, \quad (5.3.5)$$

where $\mathbf{X}_1^\infty, \dots, \mathbf{X}_{s-1}^\infty$ are the bounds on the solution set of the previous blocks; and $\mathbf{X}_s^k$ is the bound on the solution set of the current block at iteration k .

Algorithm 4 presents an adaption of the interval Gauss-Seidel method (Algorithm 2) for the lower-block-triangular system (5.3.1) - (5.3.3) using a block diagonal preconditioning matrix as in Equation (5.3.5).

Algorithm 4 Interval-Based Gauss-Seidel Method with Block-Diagonal Preconditioning

Requires: Implicit function $\mathbf{F} : \mathbb{R}^{n_x} \times \mathbb{R}^{n_p} \rightarrow \mathbb{R}^{n_x}$; Initial bounds $(\mathbf{X}^0, \mathbf{P}) \in \mathbb{IR}^{n_x} \times \mathbb{IR}^{n_p}$; Maximum number of iterations $\text{MAXIT} \in \mathbb{N}_{++}$; Termination tolerance $\text{TOL} > 0$.

Initialization: Decompose $\mathbf{F}$ into B blocks.

Main Loop: For each $s = 1, \dots, B$:

For each $k = 0, \dots, \text{MAXIT}$, compute:

$$1. \hat{\mathbf{x}}_s^k := \text{mid}(\mathbf{X}_s^k), \quad \mathbf{Y}_s^k = \left[m \left(\left[\frac{\partial \mathbf{f}_s}{\partial \mathbf{x}_s} \right] (\mathbf{P}, \mathbf{X}_1^\infty, \dots, \mathbf{X}_{s-1}^\infty, \mathbf{X}_s^k) \right) \right]^{-1};$$

$$2. [\mathbf{c}_s]^k := \mathbf{Y}_s^k \cdot [\mathbf{F}_s] (\mathbf{P}, \mathbf{X}_1^\infty, \dots, \mathbf{X}_{s-1}^\infty, \hat{\mathbf{x}}_s^k);$$

$$3. [\mathbf{G}_s]^k := \mathbf{Y}_s^k \cdot \left[\frac{\partial \mathbf{F}_s}{\partial \mathbf{x}_s} \right] (\mathbf{P}, \mathbf{X}_1^\infty, \dots, \mathbf{X}_{s-1}^\infty, \mathbf{X}_s^k);$$

For each $i = 1, \dots, n_{x_s}$:

$$4. [\Gamma_s]_i^{k+1} := \hat{x}_{s:i}^k - \frac{1}{[\mathbf{G}_s]_{i,i}^k} \left[[c_s]_i^k + \sum_{j=1}^{i-1} [\mathbf{G}_s]_{i,j}^k \cdot (X_{s:j}^{k+1} - \hat{x}_{s:j}^{k+1}) + \sum_{j=i+1}^{n_{x_s}} [\mathbf{G}_s]_{i,j}^k \cdot (X_{s:j}^k - \hat{x}_{s:j}^k) \right];$$

$$5. X_{s:i}^{k+1} := X_{s:i}^k \cap [\Gamma_s]_i^{k+1};$$

6. If $w(\mathbf{X}_s^{k+1}) - w(\mathbf{X}_s^k) \leq \text{TOL}$, set $\mathbf{X}_s^\infty := \mathbf{X}_s^k$ and break iteration.

Returns: An interval vector $\mathbf{X} := \{\mathbf{X}_1^\infty, \dots, \mathbf{X}_B^\infty\}$ such that

$$\mathbf{X} \supseteq \{ \mathbf{x} \in \mathbf{X}^0 \mid \exists \mathbf{p} \in \mathbf{P} : \mathbf{F}(\mathbf{p}, \mathbf{x}) = 0 \}$$

Albeit simple, information about the interdependency between subsystems is lost with this simple preconditioning, which can lead to significant overestimation of the solution set due to the wrapping effect. This caveat is illustrated in the following example .

Illustrating Example Consider a Hamiltonian system described by the two linear ODEs

$$\dot{u}(t) = v(t), \quad (5.3.6)$$

$$\dot{v}(t) = -u(t)$$

where $(u_0, v_0) \in \mathbb{R}^2$ are the initial conditions for $u(t)$ and $v(t)$ at $t = 0$. Discretizing the ODEs using an explicit Euler scheme with a constant step size h yields the following linear algebraic system,

$$\underbrace{\begin{pmatrix} 1 & & & & & & & \\ & 1 & & & & & & \\ -1 & -h & 1 & & & & & \\ h & -1 & & 1 & & & & \\ & & -1 & -h & \ddots & & & \\ & & h & -1 & \ddots & 1 & & \\ & & & & \ddots & & 1 & \\ & & & & \ddots & -1 & -h & 1 \\ & & & & & h & -1 & 1 \end{pmatrix}}_{\mathbf{A}} \underbrace{\begin{pmatrix} u_1 \\ v_1 \\ u_2 \\ v_2 \\ \vdots \\ u_{N-1} \\ v_{N-1} \\ u_N \\ v_N \end{pmatrix}}_{\mathbf{X}} = \underbrace{\begin{pmatrix} u_0 + h v_0 \\ -h u_0 + v_0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}}_{\mathbf{b}(u_0, v_0)}, \quad (5.3.7)$$

where $N > 0$ is the number of steps. The goal is to describe an interval enclosure $\mathbf{X}$ of the solution set subject to uncertainties in the initial conditions, $u_0 \in [-0.05, 0.05]$ and $v_0 \in [0.95, 1.05]$, and assuming an initial enclosure $u(t), v(t) \in [-10, 10]$ for the discretized states. The following two approaches are compared:

Approach 1: Bounds on the solution set are computed by applying Algorithm 2 to the $(2N \times 2N)$ system in (5.3.7) and using the inverse of the coefficient matrix $\mathbf{A}$ for preconditioning, i.e. $\mathbf{Y} = \mathbf{A}^{-1}$.

Approach 2: Bounds are computed by applying Algorithm 4, where $2N$ (1×1) blocks are solved recursively - preconditioning is not needed for the $2N$ blocks since they all comprise a single equation

and variable.

The resulting enclosure set $\mathbf{X}$ using both approaches are shown and compared in Figure 5.1. It is clear that solving the full system without block decomposition (Approach 1) generates tighter bounds than with block decomposition and block-diagonal preconditioning (Approach 2). This demonstrates the need for adequate preconditioning, when block decomposition is used. In Approach 1, preconditioning is applied to the full $2N \times 2N$ system in (5.3.7) and negates the wrapping effect as the equation system is linear. By contrast, the block-diagonal preconditioning in Approach 2 ignores the interdependency between the blocks and the bounds are affected by the wrapping effect - these bounds would eventually diverge to infinity.

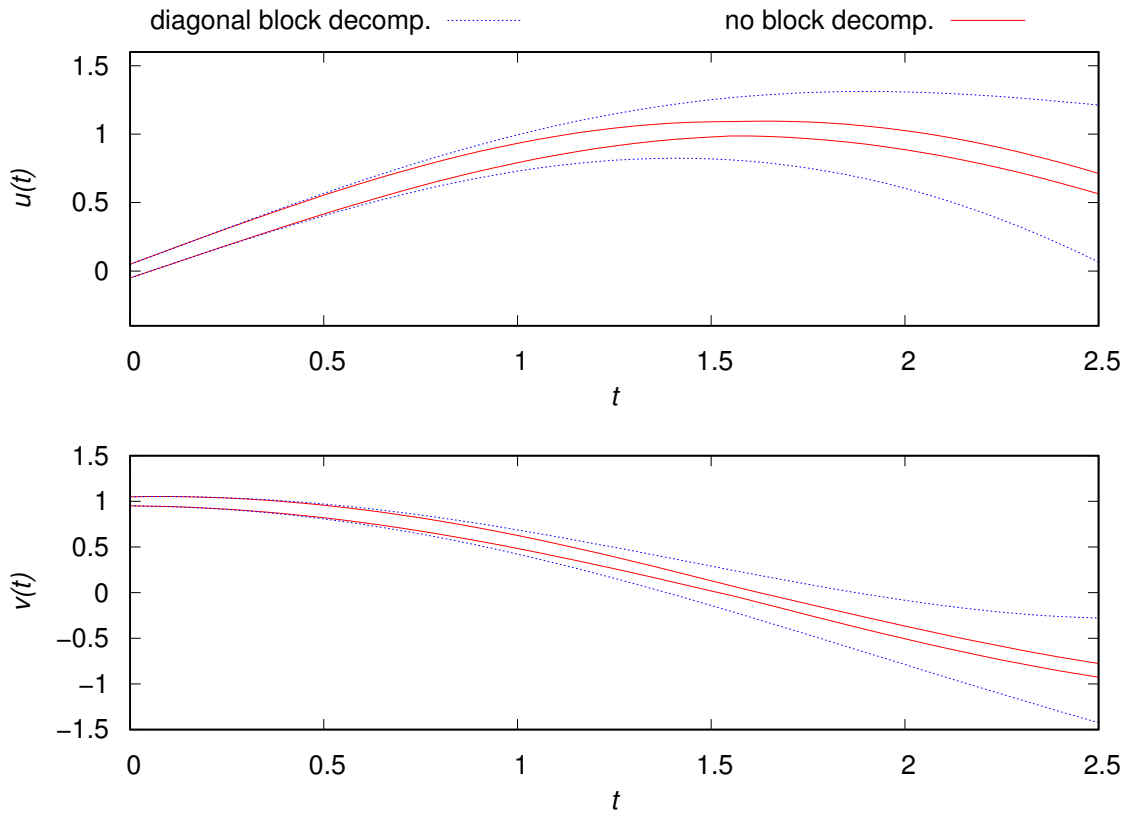


Figure 5.1: Comparison of bounds on the solution set of (5.3.7) for $u_0 \in [-0.05, 0.05]$, $v_0 \in [0.95, 1.05]$, $h = 0.05$ and $N = 50$.

Nevertheless, using an explicit scheme has the important advantage that all the resulted blocks after decomposition are of size (1×1) and therefore Algorithm 4 is much faster than Algorithm 2. Also, for a nonlinear system of ODEs, the full algebraic system would be nonlinear in the dependents, whereas the block decomposition results in elementary (1×1) blocks all linear in the dependents. Imposing the preconditioning in the block-decomposition approach so that the computed bounds are the same as those obtained with the full (non-decomposed) system is the focus of the following section.

5.3.2 Recursive block preconditioning

Preconditioning of the full algebraic system (5.3.1) - (5.3.3) using the classical mid-point inverse strategy [147] entails computing the following matrix:

$$\mathbf{Y} := \left[m \left(\frac{\partial \mathbf{F}}{\partial \mathbf{x}}(\mathbf{P}, \mathbf{X}) \right) \right]^{-1} = \left[m \begin{pmatrix} \frac{\partial \mathbf{F}_1}{\partial \mathbf{x}_1}(\mathbf{P}, \mathbf{X}_1) & & 0 \\ \vdots & \ddots & \\ \frac{\partial \mathbf{F}_B}{\partial \mathbf{x}_1}(\mathbf{P}, \mathbf{X}_1, \dots, \mathbf{X}_B) & \cdots & \frac{\partial \mathbf{F}_B}{\partial \mathbf{x}_B}(\mathbf{P}, \mathbf{X}_1, \dots, \mathbf{X}_B) \end{pmatrix} \right]^{-1}.$$

Since the inverse of a lower-triangular block matrix is also a lower-triangular block matrix, $\mathbf{Y}$ may be denoted as

$$\mathbf{Y} =: \begin{pmatrix} \mathbf{Y}_{11} & & 0 \\ \vdots & \ddots & \\ \mathbf{Y}_{B1} & \cdots & \mathbf{Y}_{BB} \end{pmatrix}.$$

Let $\mathbf{M}$ denote the matrix $m \left(\frac{\partial \mathbf{F}}{\partial \mathbf{x}}(\mathbf{P}, \mathbf{X}) \right)$. Hence, $\mathbf{Y}$ is the inverse of $\mathbf{M}$,

$$\begin{pmatrix} \mathbf{M}_{11} & & 0 \\ \vdots & \ddots & \\ \mathbf{M}_{B1} & \cdots & \mathbf{M}_{BB} \end{pmatrix} \cdot \begin{pmatrix} \mathbf{Y}_{11} & & 0 \\ \vdots & \ddots & \\ \mathbf{Y}_{B1} & \cdots & \mathbf{Y}_{BB} \end{pmatrix} = \mathbf{I}$$

where $\mathbf{I}$ is the identity matrix. Expanding the left hand side gives the following equations,

$$\begin{aligned} \mathbf{M}_{ii} \mathbf{Y}_{ii} &= \mathbf{I}, \\ \sum_{l=j}^i \mathbf{M}_{il} \mathbf{Y}_{lj} &= 0, \end{aligned}$$

for each $j = 1, \dots, i-1$. The preconditioning matrix can be constructed effectively in a block recursive manner, for each subsystem $i = 1, \dots, B$. The diagonal blocks $\mathbf{Y}_{ii}$ are given by

$$\begin{aligned} \mathbf{Y}_{ii} &= \mathbf{M}_{ii}^{-1} \\ &= \left[m \left(\frac{\partial \mathbf{F}_i}{\partial \mathbf{x}_i}(\mathbf{P}, \mathbf{X}_1, \dots, \mathbf{X}_i) \right) \right]^{-1}, \end{aligned}$$

and the off-diagonal blocks $\mathbf{Y}_{ij}$ can be computed as

$$\begin{aligned}\mathbf{Y}_{ij} &= -\mathbf{M}_{ii}^{-1} \cdot \sum_{l=j}^{i-1} \mathbf{M}_{il} \mathbf{Y}_{lj} \\ &= -\mathbf{Y}_{ii} \cdot \sum_{l=j}^{i-1} m \left(\frac{\partial \mathbf{F}_i}{\partial \mathbf{x}_l}(\mathbf{P}, \mathbf{X}_1, \dots, \mathbf{X}_i) \right) \cdot \mathbf{Y}_{lj},\end{aligned}$$

for each $j = 1, \dots, i-1$. Such recursive construction of the preconditioning matrix can be integrated with existing iterations or Gaussian elimination bounding methods (Section 4.4) in order to enable proper preconditioning on the equation systems while retaining the computational benefit of a block decomposition strategy.

5.3.2.1 Recursive iterative methods

Applying the mean-value theorem in each subsystem s requires a reference $\hat{\mathbf{x}}_s^k$ and gives:

$$\forall \mathbf{p} \in \mathbf{P}, \exists \eta_s \in [0, 1] : \mathbf{F}_s(\mathbf{p}, \hat{\mathbf{x}}) + \sum_{s'=1}^s \frac{\partial \mathbf{F}_s}{\partial \mathbf{x}_{s'}}(\mathbf{p}, \hat{\mathbf{x}}_{s'} + \eta_{s'}(\mathbf{x}_{s'} - \hat{\mathbf{x}}_{s'}^k)) \cdot (\mathbf{x}_{s'} - \hat{\mathbf{x}}_{s'}^k) = 0.$$

$\hat{\mathbf{x}}_s^k$ is chosen to be the midpoint of the interval vector $\mathbf{X}_s^k$, resembling Step 2 in Algorithm 4 with block-diagonal preconditioning. This way, an update can be performed such that for each $i = 1, \dots, n_{x_s}$:

$$\begin{aligned}X_{s:i}^{k+1} &\leftarrow \hat{x}_{s:i}^k - \left[\frac{\partial F_s}{\partial x_s} \right]_{i,i}^{-1} (\mathbf{P}, \mathbf{X}_1^\infty, \dots, \mathbf{X}_s^k) \cdot \left[F_s \right]_i(\mathbf{p}, \hat{\mathbf{x}}_1^\infty, \dots, \hat{\mathbf{x}}_s^k) \\ &\quad + \sum_{s'=1}^{s-1} \left[\frac{\partial F_s}{\partial x_{s'}} \right]_{i,s'} (\mathbf{P}, \mathbf{X}_1^\infty, \dots, \mathbf{X}_s^k) \cdot (\mathbf{X}_{s'}^\infty - \hat{\mathbf{x}}_{s'}^\infty) \\ &\quad + \sum_{j=1}^{i-1} \left[\frac{\partial F_s}{\partial x_s} \right]_{i,j} (\mathbf{P}, \mathbf{X}_1^\infty, \dots, \mathbf{X}_s^k) \cdot (X_{s:j}^{k+1} - \hat{x}_{s:j}^{k+1}) \\ &\quad + \sum_{j=i+1}^{n_{x_s}} \left[\frac{\partial F_s}{\partial x_s} \right]_{i,j} (\mathbf{P}, \mathbf{X}_1^\infty, \dots, \mathbf{X}_s^k) \cdot (X_{s:j}^k - \hat{x}_{s:j}^k)\end{aligned}$$

Similar to Algorithm 4 above, the superscript $^\infty$ refers to quantities from previous converged blocks; and n_{x_s} is the number of dependent variables in block s . Preconditioning is then applied together with the recursive block decomposition strategy to give rise to the recursive interval-based Gauss-Seidel method as presented in Algorithm 5.

Compared to Algorithm 4, the interval vector $[\mathbf{c}_s]^k$ in Step 2 is defined using not only the function matrix in the current block $[\mathbf{F}_s]$, but also those in the previous solved blocks $[\mathbf{F}_\beta]$, for each $\beta = 1, \dots, s$. Then, in addition to the diagonal block $[\mathbf{G}_{s,s}]^k$, $s-1$ blocks $[\mathbf{G}_{s,s'}]^k$ are now taken into account in Step 3.

Notice that the preconditioning blocks $\mathbf{Y}_{s,s'}$ for $s' = 1, \dots, s$ are also computed per the recursive formula derived previously. All of these blocks participate in the update formula for $[\Gamma_s]^{k+1}$ in Step 4, so the interdependency between the blocks is preserved and equivalent to the full-system preconditioning in Step 4 of Algorithm 2. Finally, the intersection in Step 5 and the termination criterion in Step 6 remain the same as in Algorithm 4.

Algorithm 5 Interval-Based Gauss-Seidel Method with Recursive Block Preconditioning

Requires: Implicit function $\mathbf{F} : \mathbb{R}^{n_x} \times \mathbb{R}^{n_p} \rightarrow \mathbb{R}^{n_x}$; Initial bounds $(\mathbf{X}^0, \mathbf{P}) \in \mathbb{IR}^{n_x} \times \mathbb{IR}^{n_p}$; Maximum number of iterations $\text{MAXIT} \in \mathbb{N}_{++}$; Termination tolerance $\text{TOL} > 0$

Initialization: Decompose $\mathbf{F}$ into B blocks.

Main Loop: **For** each $s = 1, \dots, B$:
 For each $k = 0, \dots, \text{MAXIT}$:

1. $\hat{\mathbf{x}}_s^k := m(\mathbf{X}_s^k)$.
 $\mathbf{Y}_{s,s}^k = \left[m \left(\frac{\partial \mathbf{F}_s}{\partial \mathbf{x}_s}(\mathbf{P}, \mathbf{X}_1^\infty, \dots, \mathbf{X}_s^k) \right) \right]^{-1}$,
 $\mathbf{Y}_{s,\beta}^k = -\mathbf{Y}_{s,s}^k \cdot \sum_{j=\beta}^{s-1} m \left(\frac{\partial \mathbf{F}_s}{\partial \mathbf{x}_j}(\mathbf{P}, \mathbf{X}_1^\infty, \dots, \mathbf{X}_s^k) \right) \cdot \mathbf{Y}_{j,\beta}^k, \beta = 1, \dots, s-1$.
 2. $[\mathbf{c}_s]^k := \sum_{\beta=1}^s \mathbf{Y}_{s,\beta}^k \cdot [\mathbf{F}_\beta](\mathbf{P}, \hat{\mathbf{x}}_1^\infty, \dots, \hat{\mathbf{x}}_{s-1}^\infty, \hat{\mathbf{x}}_s^k)$;
 3. $[\mathbf{G}_{s,s'}]^k := [\mathbf{Y}_{s,1} \cdots \mathbf{Y}_{s,s} \ 0 \cdots 0] \begin{bmatrix} 0 \cdots 0 & \frac{\partial \mathbf{F}_{s'}}{\partial \mathbf{x}_{s'}} \cdots \frac{\partial \mathbf{F}_B}{\partial \mathbf{x}_{s'}} \end{bmatrix}^\top$
 $= \sum_{\beta=s'}^s \mathbf{Y}_{s,\beta}^k \cdot \left[\frac{\partial \mathbf{F}_\beta}{\partial \mathbf{x}_{s'}} \right] (\mathbf{P}, \mathbf{X}_1^\infty, \dots, \mathbf{X}_{s-1}^\infty, \mathbf{X}_s^k), s' = 1, \dots, s$;
- For** each $i = 1, \dots, n_{x_s}$:
4. $[\Gamma_s]_i^{k+1} := \hat{x}_{s:i}^k - \frac{1}{[\mathbf{G}_{s,s}]_{i,i}^k} \left[[\mathbf{c}_s]_i^k + \sum_{s'=1}^{s-1} [\mathbf{G}_{s,s'}]_{i,\cdot}^k \cdot (\mathbf{X}_{s'}^\infty - \hat{\mathbf{x}}_{s'}^\infty) \right.$
 $\quad + \sum_{j=1}^{i-1} [\mathbf{G}_{s,s}]_{i,j}^k \cdot (X_{s:j}^{k+1} - \hat{x}_{s:j}^{k+1})$
 $\quad \left. + \sum_{j=i+1}^{n_{x_s}} [\mathbf{G}_{s,s}]_{i,j}^k \cdot (X_{s:j}^k - \hat{x}_{s:j}^k) \right]$;
 5. $X_{s:i}^{k+1} := X_{s:i}^k \cap [\Gamma_s]_i^{k+1}$;
 6. **If** $w(\mathbf{X}_s^{k+1}) - w(\mathbf{X}_s^k) \leq \text{TOL}$, **set** $\mathbf{X}_s^\infty := \mathbf{X}_s^k$ **and break iteration**.

Returns: An interval vector $\mathbf{X} := \{\mathbf{X}_1^\infty, \dots, \mathbf{X}_B^\infty\}$ such that

$$\mathbf{X} \supseteq \{ \mathbf{x} \in \mathbf{X}^0 \mid \exists \mathbf{p} \in \mathbf{P} : \mathbf{F}(\mathbf{p}, \mathbf{x}) = 0 \}$$

A recursive-block preconditioning algorithm based on the Krawczyk method proceeds along the same lines, by simply substituting Steps 3 - 5 in Algorithm 5 as follows:

$$\begin{aligned}
& 3. [\mathbf{H}_{s,s}]^k := \mathbf{I} - [\mathbf{G}_{s,s}]^k, \\
& \text{For each } i = 1, \dots, n_{x_s} : \\
& 4. [K_s]_i^{k+1} := \hat{x}_{s:i}^k - \left[[c_s]_i^k - \sum_{\beta=1}^{s-1} [G_{s,\beta}]_{i,\cdot}^k \cdot (\mathbf{X}_\beta^\infty - \hat{\mathbf{x}}_\beta^\infty) \right. \\
& \quad \left. + \sum_{j=1}^{i-1} [H_{s,s}]_{i,j}^k \cdot (\mathbf{X}_{s:j}^{k+1} - \hat{x}_{s:j}^{k+1}) + \sum_{j=i+1}^{n_{x_s}} [H_{s,s}]_{i,j}^k \cdot (\mathbf{X}_{s:j}^k - \hat{x}_{s:j}^k) \right]; \\
& 5. X_{s:i}^{k+1} := X_{s:i}^k \cap [K_s]_i^{k+1}.
\end{aligned}$$

Resembling the original methods in Section 4.4.1, the modified Krawczyk method does not entail division by the $[G_{s,s}]_{i,i}^k$. Therefore it may be used in place of the adapted Gauss-Seidel method whenever $[G_{s,s}]_{i,i}^k$ contains zero in range.

The counterpart of Algorithm 5 using polynomial model arithmetic is presented in Algorithm 6. Similar to Algorithm 3 in Section 4.4.1, the objective is now to determine an enclosure on the solution set $\chi(\mathbf{p})$ of $\mathbf{F}(\mathbf{x}, \mathbf{p}) = \mathbf{0}$ in the form of:

$$[\chi]^k(\mathbf{p}) =: \mathcal{P}_\chi^k(\mathbf{p}) + \mathcal{R}_\chi^k \supseteq \chi(\mathbf{p}), \quad \forall \mathbf{p} \in \mathbf{P}.$$

The key difference with interval-based methods are in the choice of the reference point in each block as the polynomial part $\mathcal{P}_\chi^k$. Resembling the interval-based methods, an update can be performed such that for each $i = 1, \dots, n_{x_s}$:

$$\begin{aligned}
[\chi]_{s:i}^{k+1}(\mathbf{p}) \leftarrow \mathcal{P}_{\chi_{s:i}}^k(\mathbf{p}) - \left[\frac{\partial \mathbf{F}_s}{\partial \mathbf{x}_s} \right]_{i,i}^{-1} & \left(\mathbf{p}, [\chi]_1^\infty, \dots, [\chi]_{s-1}^\infty, \mathcal{P}_{\chi_s}^k(\mathbf{p}) + [0, 1] \mathcal{R}_{\chi_s}^k \right) \\
& \left[\mathbf{F}_s \right]_i \left(\mathbf{p}, \mathcal{P}_{\chi_1}^\infty(\mathbf{p}), \dots, \mathcal{P}_{\chi_{s-1}}^\infty(\mathbf{p}), \mathcal{P}_{\chi_s}^k(\mathbf{p}) \right) \\
& + \sum_{s'=1}^{s-1} \left[\frac{\partial \mathbf{F}_s}{\partial \mathbf{x}_s} \right]_{i,s'} \cdot \left([\chi]_{s'}^\infty - \mathcal{P}_{\chi_{s'}}^\infty \right)(\mathbf{p}) \\
& + \sum_{j=i}^{i-1} \left[\frac{\partial \mathbf{F}_s}{\partial \mathbf{x}_s} \right]_{i,j} \left(\mathbf{p}, [\chi]_1^\infty, \dots, [\chi]_{s-1}^\infty, \mathcal{P}_{\chi_s}^k(\mathbf{p}) + [0, 1] \mathcal{R}_{\chi_s}^k \right) \cdot \left([\chi]_{s:j}^{k+1} - \mathcal{P}_{\chi_{s:j}}^{k+1} \right)(\mathbf{p}) \\
& + \sum_{j=i+1}^{n_{x_s}} \left[\frac{\partial \mathbf{F}_s}{\partial \mathbf{x}_s} \right]_{i,j}(\mathbf{p}) \cdot \mathcal{R}_{\chi_{s:j}}^k \Big]
\end{aligned}$$

Using a centered polynomial model, $\mathcal{P}_{\chi_s}^k(\mathbf{p})$ is simply the midpoint of the state enclosure $\text{mid}(\mathbf{X}_s^k)$, as

shown in the **Initializaiton** step. Same as in Algorithm 5, the preconditioning blocks $\mathbf{Y}_{s,s'}$ for $s' = 1, \dots, s$ are computed recursively. Similar to Algorithm 5, the interval vector $\llbracket \mathbf{c}_s^k \rrbracket$ in Step 2 is computed not only the function matrix in the current block $\llbracket \mathbf{F} \rrbracket_s$, but also those in the previous solved blocks $\llbracket \mathbf{F} \rrbracket_\beta$, for each $\beta = 1, \dots, s$. Furthermore, in addition to the diagonal block $\llbracket \mathbf{G}_{s,s} \rrbracket^k$, $s - 1$ blocks $\llbracket \mathbf{G}_{s,s'} \rrbracket^k$ are now taken into account in Step 3. All of these blocks participate in the update formula for $\llbracket \chi \rrbracket_{\Gamma_{s,i}}^{k+1}(\mathbf{p})$ in Step 4, so the interdependency between the blocks is preserved and equivalent to the full-system preconditioning in Step 4 of Algorithm 3. The intersection step, i.e. Step 5 in Algorithm 6, follows the same logic as Step 5 in Algorithm 2. The polynomial part of the current iteration $\mathcal{P}_{\Gamma_{s,i}}^{k+1}$ is retained because it improves with every iteration. The remainder part is computed by first subtracting the improved polynomial term from the polynomial models of the previous and the current iterations, i.e. $\llbracket \chi \rrbracket_{s,i}^{k+1}$ and $\llbracket \chi \rrbracket_{s,i}^k$, respectively, and then intersecting the remainders. As a result, the subtraction step gives rise to $\mathcal{P}_{\Gamma_{s,i}}^k - \mathcal{P}_{\Gamma_{s,i}}^{k+1}$ and introduces some overestimation. Therefore, the resulting polynomial model of the current iteration, $\llbracket \chi \rrbracket_{s,i}^{k+1}$, is not necessarily nested in that of the previous iteration, $\llbracket \chi \rrbracket_{s,i}^k$, in contrast to the interval-based version in Algorithm 5. Finally, the iteration terminates when the difference in width between the updated and the current remainder terms becomes trivial, as shown in in Step 6.

Lastly, in the Krawczyk method variant with recursive block preconditioning, Steps 2-4 of Algorithm 6 may be substituted with:

$$2. \llbracket \mathbf{H}_{s,s} \rrbracket^k(\mathbf{p}) := \mathbf{I} - \llbracket \mathbf{G}_{s,s} \rrbracket^k(\mathbf{p}),$$

For each $i = 1, \dots, n_{x_s}$:

$$3. \llbracket K_s \rrbracket_i^{k+1}(\mathbf{p}) := \mathcal{P}_{x_{s,i}}^k(\mathbf{p}) - \left[\llbracket \mathbf{c}_s \rrbracket_i^k(\mathbf{p}) + \sum_{\beta=1}^{s-1} \llbracket \mathbf{G}_{s,\beta} \rrbracket_{i,\cdot}^k \cdot \left(\llbracket \chi \rrbracket_\beta^\infty - \mathcal{P}_{\chi_\beta}^{k,\infty} \right)(\mathbf{p}) \right. \quad (5.3.9a)$$

$$+ \sum_{j=1}^{i-1} \llbracket \mathbf{H}_{s,s} \rrbracket_{i,j}^k \cdot \left(\llbracket \chi \rrbracket_{s,j}^{k+1} - \mathcal{P}_{\chi_{s,j}}^{k+1} \right)(\mathbf{p}) \quad (5.3.9b)$$

$$\left. + \sum_{j=i+1}^{n_{x_s}} \llbracket \mathbf{H}_{s,s} \rrbracket_{i,j}^k(\mathbf{p}) \cdot \mathcal{R}_{\chi_{s,j}}^k \right],$$

$$4. \llbracket \chi \rrbracket_i^{k+1}(\mathbf{p}) := \left(\mathcal{P}_{K_i}^{k+1}(\mathbf{p}), \mathcal{R}_{K_i}^{k+1} \cap \left[\mathcal{P}_{\chi_i}^k - \mathcal{P}_{K_i}^{k+1} \right](\mathbf{p}) + \mathcal{R}_{\chi_i}^k \right)$$

In the case of linear parametric equation, i.e. when a subsystem $\mathbf{F}_s(\mathbf{p}, \mathbf{x}_1, \dots, \mathbf{x}_s) = 0$ in Equations (5.3.1) - (5.3.3) is partially linear in the dependent $\mathbf{x}_s$, the vector $\mathbf{c}_s$ and matrix $\mathbf{G}_{s,s'}$ become independent of the iteration counter k and need not be recomputed at each iteration. Another recursive approach tailored to linear algebraic equation is presented in the following subsection.

Algorithm 6 Polynomial Model-Based Gauss-Seidel Method with Recursive Block Preconditioning

Requires: Implicit function $\mathbf{F} : \mathbb{R}^{n_x} \times \mathbb{R}^{n_p} \rightarrow \mathbb{R}^{n_x}$; Initial bounds $(\mathbf{X}^0, \mathbf{P}) \subset \mathbb{IR}^{n_x} \times \mathbb{IR}^{n_p}$; Maximum number of iterations $\text{MAXIT} \in \mathbb{N}_{++}$; Termination tolerance $\text{TOL} > 0$; Polynomial order $q \in \mathbb{N}_{++}$

Initialization: Decompose $\mathbf{F}$ into B blocks.
 Set $\mathcal{P}_\chi^0(\mathbf{p}) = m(\mathbf{X}^0)$, $\mathcal{R}_\chi^0 = \frac{1}{2}w(\mathbf{X}^0) \cdot [-1, 1]$.

Main Loop: For each $s = 1, \dots, B$:
 For each $k = 1, \dots, \text{MAXIT}$:

1. $\mathbf{Y}_{s,s}^k = \left[m \left(\left[\frac{\partial \mathbf{F}_s}{\partial \mathbf{x}_s} \right] (\mathbf{p}, [\chi]_1^\infty, \dots, [\chi]_{s-1}^\infty, \mathcal{P}_{\chi_s}^k(\mathbf{p}) + [0, 1]\mathcal{R}_{\chi_s}^k) \right) \right]^{-1}$
 $\mathbf{Y}_{s,\beta}^k = -\mathbf{Y}_{s,s}^k \cdot \sum_{j=\beta}^{s-1} m \left(\left[\frac{\partial \mathbf{F}_s}{\partial \mathbf{x}_s} \right] (\mathbf{p}, [\chi]_1^\infty, \dots, [\chi]_{s-1}^\infty, \mathcal{P}_{\chi_s}^k(\mathbf{p}) + [0, 1]\mathcal{R}_{\chi_s}^k) \right)$
 $\cdot \mathbf{Y}_{j,\beta}^k, \quad \beta = 1, \dots, s-1.$
 2. $[\mathbf{c}_s]^k(\mathbf{p}) := \sum_{\beta=1}^s \mathbf{Y}_{s,\beta}^k \cdot [\mathbf{F}_\beta] \left(\mathbf{p}, \mathcal{P}_{\chi_1}^\infty(\mathbf{p}), \dots, \mathcal{P}_{\chi_{s-1}}^\infty(\mathbf{p}), \mathcal{P}_{\chi_s}^k(\mathbf{p}) \right);$
 3. $[\mathbf{G}_{s,s'}]^k(\mathbf{p}) := \sum_{\beta=s'}^s \mathbf{Y}_{s,\beta}^k \cdot \left[\frac{\partial \mathbf{F}_\beta}{\partial \mathbf{x}_{s'}} \right] (\mathbf{p}, [\chi]_1^\infty, \dots, [\chi]_{s-1}^\infty,$
 $\mathcal{P}_{\chi_s}^k(\mathbf{p}) + [0, 1]\mathcal{R}_{\chi_s}^k), s' = 1, \dots, s;$
- For each $i \in 1, \dots, n_{x_s}$:
4. $[\chi]_{\Gamma_{s:i}}^{k+1}(\mathbf{p}) := \mathcal{P}_{\chi_{s:i}}^k(\mathbf{p}) - \frac{1}{[\mathbf{G}_{s,s}]_{i,i}^k(\mathbf{p})} \left[[\mathbf{c}_s]_i^k(\mathbf{p}) \right.$
 $+ \sum_{\beta=1}^{s-1} [\mathbf{G}_{s,\beta}]_{i,\cdot}^k \cdot ([\chi]_\beta^\infty - \mathcal{P}_{\chi_\beta}^\infty)(\mathbf{p})$
 $+ \sum_{j=1}^{i-1} [\mathbf{G}_{s,s}]_{i,j}^k \cdot ([\chi]_{s:j}^{k+1} - \mathcal{P}_{\chi_{s:j}}^{k+1})(\mathbf{p})$
 $\left. + \sum_{j=i+1}^{n_{x_s}} [\mathbf{G}_{s,s}]_{i,j}^k(\mathbf{p}) \cdot \mathcal{R}_{\chi_{s:j}}^k \right];$
 5. $[\chi]_{s:i}^{k+1}(\mathbf{p}) := \left(\mathcal{P}_{\Gamma_{s:i}}^{k+1}(\mathbf{p}), \mathcal{R}_{\Gamma_{s:i}}^{k+1} \cap [\mathcal{P}_{\chi_{s:i}}^k - \mathcal{P}_{\Gamma_{s:i}}^{k+1}](\mathbf{p}) + \mathcal{R}_{\chi_{s:i}}^k \right);$
 6. If $w(\mathcal{R}_{\chi_s}^k) - w(\mathcal{R}_{\chi_s}^{k+1}) \leq \text{TOL}$, set $[\chi]_s^\infty := \mathcal{P}_{\chi_s}^k(\mathbf{p}) + \mathcal{R}_{\chi_s}^k$ and break iteration.

Returns: A q -th order polynomial model $[\chi] := \{[\chi_1]^\infty, \dots, [\chi_B]^\infty\}$ satisfying

$$[\chi] =: \mathcal{P}_\chi(\mathbf{p}) + \mathcal{R}_\chi \rightarrow \chi(\mathbf{p}), \quad \forall \mathbf{p} \in \mathbf{P}.$$

5.3.2.2 Recursive Gaussian elimination

Consider the special case of a subsystem $\mathbf{F}_s(\mathbf{p}, \mathbf{x}_1, \dots, \mathbf{x}_s) = 0$ in Equations (5.3.1) - (5.3.3) that is partially linear in the dependent $\mathbf{x}_s$; that is, this subsystem can be re-written in the form:

$$\mathbf{A}_s(\mathbf{p}, \mathbf{x}_1, \dots, \mathbf{x}_{s-1})\mathbf{x}_s = \mathbf{b}_s(\mathbf{p}, \mathbf{x}_1, \dots, \mathbf{x}_{s-1}) \quad (5.3.10)$$

where $\mathbf{A}_s := \frac{\partial \mathbf{F}_s}{\partial \mathbf{x}_s}$, $\mathbf{b}_s := -\mathbf{F}_s|_{\mathbf{x}_s=0}$. Such subsystems are found frequently in practical applications, for instance when an explicit discretization scheme is applied to a nonlinear ODE system as in Equation (5.2.2) in Section 5.2.1.

Instead of applying the Gauss-Seidel approach to bound the solutions of this subsystem as in Algorithms 5 and 6 above, one can apply Gaussian elimination while still taking advantage of the recursive block preconditioning strategy. Specifically, applying the mean-value theorem, and preconditioning to Equation (5.3.10) gives:

$$\begin{aligned} \mathbf{G}_{s,s}\mathbf{x}_s &= -\mathbf{c}_s - \sum_{s'=1}^{s-1} \mathbf{G}_{s,s'}(\mathbf{x}_{s'} - \hat{\mathbf{x}}_{s'}) \\ \text{with: } \mathbf{c}_s &:= \mathbf{Y}_{s,s} \cdot \mathbf{F}_s(\mathbf{p}, \hat{\mathbf{x}}_1, \dots, \hat{\mathbf{x}}_{s-1}, 0) + \sum_{\beta=1}^{s-1} \mathbf{Y}_{s,\beta} \cdot \mathbf{F}_\beta(\mathbf{p}, \hat{\mathbf{x}}_1, \dots, \hat{\mathbf{x}}_\beta) \\ \mathbf{G}_{s,s'} &:= \mathbf{Y}_{s,s} \cdot \frac{\partial \mathbf{F}_s}{\partial \mathbf{x}_s}(\mathbf{p}, \hat{\mathbf{x}}_1 + \eta_1(\mathbf{x}_1 - \hat{\mathbf{x}}_1), \dots, \hat{\mathbf{x}}_{s-1} + \eta_{s-1}(\mathbf{x}_{s-1} - \hat{\mathbf{x}}_{s-1}), 0) \\ &\quad + \sum_{\beta=s'}^{s-1} \mathbf{Y}_{s,\beta} \cdot \frac{\partial \mathbf{F}_\beta}{\partial \mathbf{x}_{s'}}(\mathbf{p}, \hat{\mathbf{x}}_1 + \eta_1(\mathbf{x}_1 - \hat{\mathbf{x}}_1), \dots, \hat{\mathbf{x}}_\beta + \eta_\beta(\mathbf{x}_\beta - \hat{\mathbf{x}}_\beta)) \end{aligned}$$

where $\hat{\mathbf{x}}_1, \dots, \hat{\mathbf{x}}_{s-1}$ are reference points from previous blocks, and $\eta_1, \dots, \eta_{s-1} \in [0, 1]$ are scalars in the mean-value theorem.

For interval bounding, the interval vector $[\mathbf{c}_s]$ and interval matrix $[\mathbf{G}_{s,s'}]$ are therefore computed per Steps 2 and 3 in Algorithm 5, where the reference point in Step 1 is now chosen as $\hat{\mathbf{x}}_s = 0$. Then, the resulting interval linear system:

$$[\mathbf{G}_{s,s}] \cdot \mathbf{X}_s = -[\mathbf{c}_s] - \sum_{s'=1}^{s-1} [\mathbf{G}_{s,s'}] \cdot [\mathbf{X}_{s'}^\infty - \hat{\mathbf{x}}_{s'}^\infty]$$

is solved using interval-based Gaussian elimination (Section 4.4.2).

For polynomial models likewise, the vector $\llbracket c_s \rrbracket(\mathbf{p})$ and matrix $\llbracket G_{s,s'} \rrbracket(\mathbf{p})$ are computed per Steps 1 and 2 in Algorithm 6 with 0 as the reference value instead of $\mathcal{P}_{\chi_s}(\mathbf{p})$. Then the resulting linear system is solved with Gaussian elimination in polynomial model arithmetic (Section 4.4.2).

Whenever a subsystem is partially linear in the dependent variables, the steps in the main loops of Algorithms 5 and 6 can therefore be easily modified to use Gaussian elimination instead of a Gauss-Seidel (or Krawczyk) iteration. The advantage is that Gaussian elimination does not require initial bounds for the dependent. For low-dimensional subsystems, Gaussian elimination often entail fewer operations than iterative methods in addition.

5.3.3 Implementations in CRONOS

The full discretization approach for bounding solution sets for ODEs is implemented in CRONOS using a new header file `odebnd_expand.hpp`. Various discretization methods have been implemented, including explicit Runge-Kutta schemes of orders 1 – 6 and 8 and Taylor series expansion. Coefficients of high-order Runge-Kutta schemes are adopted from References [148, 149, 150]. The order and step size of discretization can be specified by the user during the problem setup. The problem defined using the class `ODEBND_EXPAND` then calls another class `AEBND`, which is implemented in the header file `aebnd.hpp`, in CRONOS to bound solution sets to the resulting algebraic equations.

The size of the algebraic system to be solved with `AEBND` can be controlled through the option `LBLK`. Explicit discretization naturally yields a block-structured algebraic system, with the same number of blocks as the number of discretization stages. The time horizon can be split to create subsystems which comprise a number of blocks and are solved sequentially. The value of `LBLK` indicates the number of blocks in each subsystem. Notice that if the `LBLK` value is equal to the number of discretization stages, the time horizon is not split and the entire algebraic system is solved simultaneously. The number of block shifts between subsystems is controlled through another option `DBLK`. Naturally, `DBLK` cannot have a value that is greater than `LBLK`. By specifying the same value for both `DBLK` and `LBLK`, the resulting subsystems do not have overlapping blocks or dependent variables. The subsystem then undergoes preconditioning with the selected block decomposition strategy which is specified using the option `AEBND.BLKDEC`. The option `AEBND.BLKDEC` can take any of the three values, `NONE` for no decomposition, `DIAG` for diagonal decomposition, or `RECUR` for recursive decomposition. While the first subsystem is initialized using the given *a priori* state bounds, the subsequent subsystems are re-initialized and preconditioned based on computed bounds from previous subsystems. It is important to note that because each subsystem is preconditioned independently, splitting the time horizon leads to loss in interdependency between the blocks in later subsystems and the blocks in earlier subsystems, and thus wrapping.

The use of `AEBND` has been demonstrated in Section 4.4.3. More specifically, some existing features

of AEBND are realized with the support of the following third-party libraries. The interval and polynomial model arithmetics are performed using the MC++ library [80]. The HSL libraries [144] are employed for block decomposition. In particular, Algorithm MC21 [151] is used to permute the rows of an algebraic system so that the matrix has a zero-free diagonal; and Algorithm MC13 [152] is subsequently applied to permute the matrix into a block lower triangular form. The Jacobian matrix is generated via automatic differentiation using the package FADBAD++ [153]. Matrix inversion involved in calculating preconditioning matrix is achieved using the package CPPLAPACK [154]. In addition to the existing features of AEBND as demonstrated in Section 4.4.3, the proposed diagonal and recursive block decomposition strategies are added to the header file as a new option.

5.4 Case study: Series reaction in batch reactor

Consider a series reaction $A \rightarrow B \rightarrow C$ in a batch reactor, whose dynamics are described by

$$\begin{aligned}\dot{x}_A(t) &= -k_1(T)x_A(t)^2, \\ \dot{x}_B(t) &= k_1(T)x_A(t)^2 - k_2(T)x_B(t), \\ x_A(0) &= 1, \quad x_B(0) = 0,\end{aligned}\tag{5.4.1}$$

where x_A and x_B denote concentrations of the reactants A and B; and $k_1(T)$ and $k_2(T)$ are temperature-dependent reaction rate constants given by,

$$\begin{aligned}k_1(T) &:= 4.0 \times 10^3 \exp\left(\frac{-2.5 \times 10^3}{T}\right), \\ k_2(T) &:= 6.2 \times 10^5 \exp\left(\frac{-5.0 \times 10^3}{T}\right).\end{aligned}$$

The goal is to describe an interval enclosure of the solution set for $t \in [0, 1]$ subject to a temperature range $T \in [298\text{K}, 398\text{K}]$ and *a priori* enclosure $\mathbf{X}^0 = [0, 1]^2$. The results provided by a full discretization of the ODEs are compared to those provided by state-of-the-art, continuous-time set-valued integration.

In particular, the following two strategies are compared:

- (1) DINEQ: Continuous-time set-propagation is applied to bound the ODEs using Chebyshev models with ellipsoidal remainders or the set arithmetic (Section 4.5.1).
- (2) DISCR: Full discretization of the ODE system is conducted using explicit Runge-Kutta method of order 4. The resulting algebraic equations are then bounded using either the Gaussian elimination,

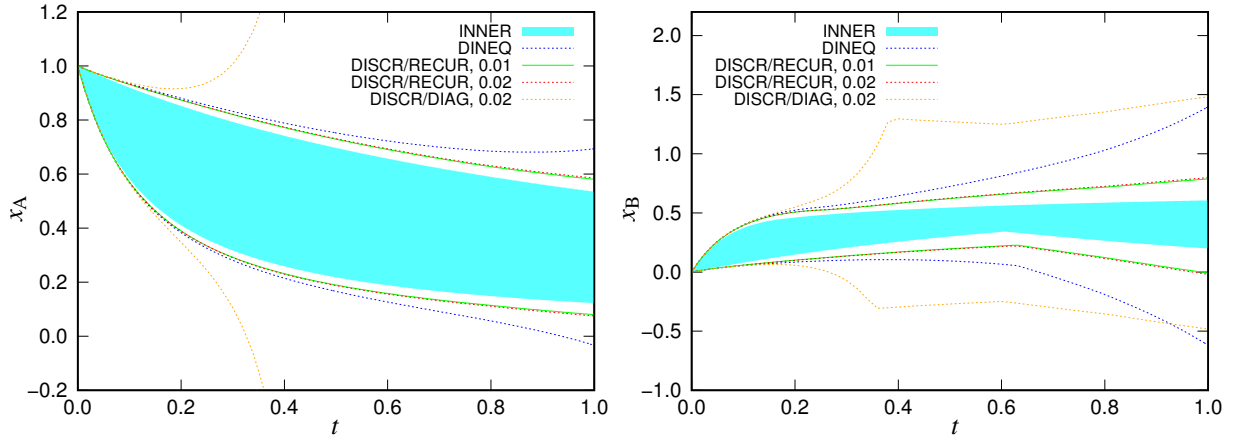
or the Gauss-Seidel method. Chebyshev models of the same order as in the DINEQ strategy are used to enable fair comparisons. Several preconditioning and block-decomposition strategies are compared: (a) No block-decomposition and midpoint-inverse preconditioning of algebraic equation systems and solution using Gauss-Seidel (DISCR/FULL); (b) lower-triangular block-decomposition and block-diagonal preconditioning using midpoint inverse and solution using either Gauss-Seidel or Gaussian elimination (DISCR/DIAG); (c) same as DISCR/DIAG with recursive block preconditioning (DISCR/RECUR).

5.4.1 Implementation in CRONOS

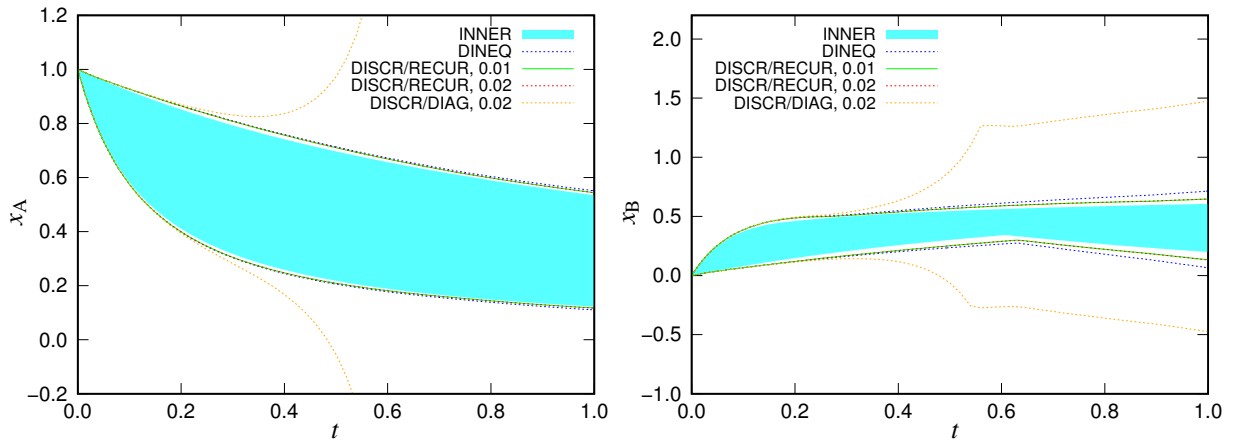
Both the DINEQ and the DISCR approaches are implemented in CRONOS. The DINEQ approach is implemented using the class ODEBND with `odebnd_sundials.hpp`, in the same manner as illustrated in Section 4.5.2. The DISCR approaches are implemented using the new class ODEBND_EXPAND, and can be found in Appendix B. Explicit Runge-Kutta method is selected for discretization by specifying `options.INTMETH`; and the order and step size are specified through `options.TORD` and `options.H0` respectively. The options `options.LBLK` and `options.DBLK` are set to be equal in this case study, indicating no block overlap between adjacent subsystems created from splitting the time horizon. The bounding method and the choice of preconditioner for the resulting algebraic system are specified through the options of the class AEBND, as illustrated in Listing 4.7. The block decomposition method is specified through `options.AEBND.BLKDEC`, which is assigned a value `DIAG` for the DISCR/DIAG approach; a value `RECUR` for the DISCR/RECUR approach; and `NONE` for the DISCR/FULL approach.

5.4.2 Reachable set enclosures

All DINEQ and DISCR use Chebyshev models (order 3 and order 4) for constructing polynomial enclosures to the reachable sets. In all approaches in the DISCR family, the classical 4-th order Runge-Kutta scheme is used. Two step sizes, 0.02 and 0.01 are attempted to investigate the effects of discretization errors. Furthermore, the modified Gauss-Seidel method in Algorithm 6 is used for bounding solution sets to the resulting algebraic equation system in the DISCR approaches. The reachable set enclosures computed using the three approaches, DINEQ, DISCR/RECUR, and DISCR/DIAG are compared in Figure 5.2. In fact, the two approaches DISCR/RECUR and DISCR/FULL generate exactly the same enclosures; since the recursive block decomposition strategy is designed to retain the interdependency between blocks. The reachable set enclosures are also



(a) Using 3rd-order Chebyshev model-based estimator



(b) Using 4th-order Chebyshev model-based estimator

Figure 5.2: Comparison of reachable set enclosures to (5.4.1) with *a priori* state bounds $\mathbf{X}^0 = [0, 1]^2$ using: inner approximation (INNER); continuous set-valued integration (DINEQ) with Chebyshev model-based estimators; discretized set-valued integration with recursive block decomposition (DISCR/RECUR); and discretized set-valued integration with diagonal block decomposition (DISCR/DIAG). A step size of either 0.02 or 0.01 is used in the discretized set-valued integration approaches.

compared with an inner-approximation constructed (INNER) via computing an extensive number of uncertainty scenarios in the given initial condition range.

It can easily be observed that for all DINEQ and DISCR approaches, with all other parameters held constant, using a higher-order Chebyshev model results in a tighter enclosure. In general, DISCR/RECUR is more effective than DINEQ. Notice that the DISCR approaches are not validated, where discretization errors are simply discarded. Conversely, the DINEQ approach is validated and guaranteed to contain the true solution. To investigate the effects of discretization errors in the

DISCR/RECUR approach, two step sizes, 0.02 and 0.01, are considered separately. It is evident that halving the step size has negligible benefit in producing tighter bounds. Hence, it can be deduced that the discretization errors herein are trivial; and the advantage of DISCR/RECUR over DINEQ is not an artefact because of the failure to account for discretization errors. Furthermore, it is clear that the recursive block decomposition strategy (DISCR/RECUR) successfully mitigates the wrapping effect which is significant in the diagonal strategy (DISCR/DIAG).

The computational efficiencies of these DINEQ and DISCR approaches are compared in Table 5.3. The table reports the CPU times (t_{CPU}) required by using DINEQ, DISCR/FULL or DISCR/RECUR using 3rd- and 4th-order Chebyshev models. A PC with a Intel Core i7-2600 CPU @ 3.60 GHz x 4 processor and running GCC version 7.4.0 was used. For all DISCR approaches reported in this table, the step size of 0.02 is used.

In Table 5.3, the polynomial model remainder bounds $\mathcal{R}_{x_A}$ and $\mathcal{R}_{x_B}$ are indicators of bound tightness. Comparing 3rd- and 4th- order Chebyshev models, it is clear that using a higher order yields tighter enclosures, but is more computationally expensive. Using the Chebyshev model of the same order, DISCR/FULL generates on average 60-70% tighter bounds of both x_A and x_B than DINEQ. However, the computational effort demanded by the former was more than 7- to 25-fold that required by the latter, as reflected in t_{CPU} .

Without splitting the time horizon, using recursive block decomposition (DISCR/RECUR with $\text{LBLK} = 50$) cuts the computational demand by approximately 80% compared to DISCR/FULL. Recall that in the implementation in Listing B.1, the header file `odebnd_expand.hpp` allows specifying the value for the option `LBLK`, which denotes the length of blocks in each algebraic subsystem. Blocks within the same subsystem are preconditioned and solved simultaneously. It is clear that having smaller subsystems (i.e. a smaller `LBLK` value) allows further CPU time reduction, because there are fewer algebraic equations in each subsystem to be solved. However, smaller subsystems also lead to slightly looser reachable set enclosures computed. This is because each subsequent subsystem has to be reinitialized and preconditioned independently, and some wrapping is introduced. The DISCR/RECUR approach may outperform the DINEQ approach in terms of computational efficiency in this way.

Lastly, with the DISCR approach, since all the algebraic equation blocks herein are of dimension 1 and linear in the dependents, the recursive Gaussian elimination (GE) method proposed in Section 5.3.2.2 is also applicable in addition to the Gauss-Seidel (GS) method. In fact, the reachable set enclosures computed using GE and GS are virtually indistinguishable because the blocks have dimension of 1. However, GE is about 30 to 50% faster than GS.

Table 5.3: Performance of various techniques towards enclosing solution sets to (5.4.1) at $t = 1$ subject to *a priori* state bounds $\mathbf{X}^0 \in [0, 1]^2$ using continuous set-valued integration (DINEQ) and discretized set-value integration (DISCR) approaches with 3rd- and 4th-order Chebyshev model estimators. In all DISCR approaches, a classical 4th-order Runge-Kutta scheme with a step size of 0.02 is used. Recursive Gaussian elimination (GE) and recursive Gauss-Seidel (GS) methods are applied to solve the resulting algebraic system. The discretized algebraic system undergoes either no block decomposition (DISCR/FULL) or recursive block decomposition (DISCR/RECUR). LBLK stands for the length of blocks in each algebraic subsystem for simultaneous bounding when block decomposition is conducted. A PC with a Intel Core i7-2600 CPU @ 3.60 GHz x 4 processor and running GCC version 7.4.0 was used.

		DINEQ	DISCR/	DISCR / RECUR				
			FULL	LBLK=50	LBLK=25	LBLK=10	LBLK=5	LBLK=2
Order 3	$\mathcal{R}_{x_A}$	0.157	0.048	0.048	0.048	0.048	0.048	0.048
	$\mathcal{R}_{x_B}$	0.790	0.190	0.190	0.196	0.204	0.208	0.210
	t_{CPU} (GE)	0.08 s	0.99 s	0.21 s	0.10s	0.06 s	0.04 s	0.04 s
	t_{CPU} (GS)	0.08 s	1.80 s	0.32 s	0.15s	0.08 s	0.07 s	0.06 s
Order 4	$\mathcal{R}_{x_A}$	0.012	0.005	0.005	0.005	0.005	0.005	0.005
	$\mathcal{R}_{x_B}$	0.099	0.032	0.032	0.034	0.034	0.035	0.035
	t_{CPU} (GE)	0.15 s	1.08 s	0.22 s	0.13 s	0.07 s	0.06 s	0.04 s
	t_{CPU} (GS)	0.15 s	1.75 s	0.37 s	0.17 s	0.12 s	0.08 s	0.07 s

5.5 Case study: Nonlinear thermofluidic engine

Having showcased the effectiveness of the proposed full discretization approach coupled with recursive preconditioning in the previous case study, the problem of propagating uncertainty in nonlinear thermofluidic engines is revisited, in particular the NIFTE device.

In practice, the operation of the NIFTE is inevitably subject to disturbances, which are manifested as uncertainty in the pressures and flowrates at various positions of the device. Herein, the aim is to enclose

the device trajectories subject to uncertainties in the initial conditions. Specifically, the dynamics of the NIFTE are described by a system of ODEs,

$$\frac{d\hat{\mathbf{x}}}{d\hat{t}} = \mathbf{f}(\hat{\mathbf{x}}(\hat{t}), \mathbf{p}), \quad (5.5.1)$$

with: $\hat{\mathbf{x}}(0) \in [\hat{\mathbf{x}}_L, \hat{\mathbf{x}}_U]$

where $\mathbf{f}$ corresponds to the NTP-TL model in (2.3.7a')–(2.3.7f)). The vector $\hat{\mathbf{x}}$ comprises nine independent thermodynamic states, namely pressures and flowrates at various locations of the NIFTE ($\hat{P}_{\text{ad}}, \hat{P}_{\text{d}}, \hat{P}_{\text{p}}, \hat{U}_{\text{f}}, \hat{U}_{\text{p}}, \hat{P}_{\text{tl},1}, \hat{P}_{\text{tl},2}, \hat{P}_{\text{tl},3}, \hat{P}_{\text{tl},4}$); and $\mathbf{p}$ is a vector of model parameters. $\hat{\mathbf{x}}_L$ and $\hat{\mathbf{x}}_U$ denote the lower and the upper bounds in the initial states respectively. The definitions of the state variables and parameters can be found in Section 2.3 in Chapter 2; and the nominal parameter values are listed in Table 2.1. It is important to note that the ODEs in problem (5.5.1) are non-dimensionalized (denoted with $\hat{\cdot}$), by using P° , U° and τ as reference values for the pressures, flowrates and time, respectively. Two scenarios with different levels of uncertainty are investigated in the next subsection.

5.5.1 Case 1: Reachable set enclosure subject to $\hat{\mathbf{x}}_0 \in [0.9, 1.1] \hat{\mathbf{x}}_{\text{ref}}$

A 10% uncertainty is considered in the initial conditions of all nine states, i.e. $\pm 10\%$ deviation from their reference positions $\hat{\mathbf{x}}_{\text{ref}}$, which correspond to a (stable) cyclic steady state of the NIFTE, with a reference cycle period $\hat{T}_{\text{ref}} = 0.584$. Hence, $\hat{\mathbf{x}}_L := 0.9\hat{\mathbf{x}}_{\text{ref}}$, and $\hat{\mathbf{x}}_U := 1.1\hat{\mathbf{x}}_{\text{ref}}$. The aim is to enclose the state trajectories over a period of two reference cyclic steady states, i.e. $\hat{t} \in [0, 2\hat{T}_{\text{ref}}]$.

5.5.1.1 Implementation and comparison in CRONOS

The continuous-time set-valued integration (DINEQ), and the new discretization approach with recursive block decomposition strategy (DISCR/RECUR) as available in CRONOS are first compared. The results for the five key states (i.e. $\hat{P}_{\text{ad}}, \hat{P}_{\text{d}}, \hat{P}_{\text{p}}, \hat{U}_{\text{f}}$ and $\hat{U}_{\text{p}}$) are shown in Figure 5.3. A fourth-order Chebyshev model is used in both approaches. In the DISCR/RECUR approach, the dynamical system is furthermore discretized using the classical fourth-order Runge-Kutta method with a step size $h = 0.005\hat{T}_{\text{ref}}$. Since the scheme is explicit, the resulting system of algebraic equations has a block structure where all the blocks are of dimension 1 and partially linear in the dependent variables. This allows the use of Gaussian elimination which is more computationally efficient than the iterative Gauss-Seidel method. Note that problem (5.5.1) is a stiff system because of the tanh switching function (see Section 2.3.3) and therefore a relative small step size is necessary for numerical stability. The fully discretized system comprises 3609

algebraic equations¹. The full algebraic system is split into subsystems, whose size is controlled through `options.LBLK`. In this case study, the same value is enforced for `options.LBLK` and `options.DBLK`, hence subsystems do not share dependent variables.

Comparing against the inner-approximating samples that are represented in the blue shaded regions in Figure 5.3, both approaches yield tight enclosures of the possible state trajectories even at locations of steep transitions; and these bounds can hardly be distinguished. Moreover, both approaches are able to closely predict the contraction of the trajectories around the stable limit cycle of the NIFTE. The bounds for the nine key states after 1 and 2 reference cycles, i.e. $\hat{t} = \hat{T}_{\text{ref}}$ and $\hat{t} = 2\hat{T}_{\text{ref}}$, are compared to those at $\hat{t} = 0$ in Table 5.4. Those bounds that are greater than their respective initial values are highlighted in red; the others in blue. It can be observed that the states with relatively wider initial ranges (i.e. $\hat{P}_{\text{d}}$, $\hat{P}_{\text{p}}$, $\hat{U}_{\text{p}}$ and $\hat{P}_{\text{tl},3}$) are more likely to experience reductions in uncertainties than those with narrower initial ranges (i.e. $\hat{U}_{\text{f}}$, $\hat{P}_{\text{tl},1}$ and $\hat{P}_{\text{tl},2}$).

Table 5.4: Width of bounds computed for nine thermodynamic state variables in problem 5.5.1 at various time instances subject to $\hat{\mathbf{x}}_0 \in [0.9, 1.1] \hat{\mathbf{x}}_{\text{ref}}$. A step size $h = 0.005\hat{T}_{\text{ref}}$ was used for the DISCR/RECUR approach; and a fourth-order Chebyshev polynomial estimator was adopted for both DINEQ and DISCR/RECUR approaches. State bounds that are greater than the respective initial bounds are highlighted in red; the others in blue.

	$\hat{t}$	$\Delta\hat{P}_{\text{ad}}$	$\Delta\hat{P}_{\text{d}}$	$\Delta\hat{P}_{\text{p}}$	$\Delta\hat{U}_{\text{f}}$	$\Delta\hat{U}_{\text{p}}$
DINEQ	0	8.70E-03	9.38E-03	7.54E-03	0.00E+00	7.61E-02
	$\hat{T}_{\text{ref}}$	1.27E-02	5.14E-03	4.83E-03	3.92E-02	4.46E-02
	$2\hat{T}_{\text{ref}}$	6.13E-03	2.71E-03	3.06E-03	5.47E-02	5.78E-02
DISCR/ RECUR	$\hat{T}_{\text{ref}}$	1.27E-02	5.20E-03	4.87E-03	3.89E-02	4.44E-02
	$2\hat{T}_{\text{ref}}$	6.75E-03	2.98E-03	3.31E-03	5.44E-02	5.76E-02

	$\hat{t}$	$\Delta\hat{P}_{\text{tl},1}$	$\Delta\hat{P}_{\text{tl},2}$	$\Delta\hat{P}_{\text{tl},3}$	$\Delta\hat{P}_{\text{tl},4}$
DINEQ	0	1.77E-04	6.20E-04	4.99E-03	3.96E-03
	$\hat{T}_{\text{ref}}$	6.93E-04	2.42E-03	3.47E-03	5.71E-03
	$2\hat{T}_{\text{ref}}$	2.76E-04	1.08E-03	1.13E-03	3.18E-03
DISCR/ RECUR	$\hat{T}_{\text{ref}}$	6.99E-04	2.44E-03	3.53E-03	5.81E-03
	$2\hat{T}_{\text{ref}}$	3.44E-04	1.31E-03	1.56E-03	3.66E-03

¹9 states $\times$ 401 discretized time instances = 3609 algebraic equations

Similar to the case study of a series reaction in a batch reactor in Section 5.4, the full discretization approach is generally more computationally demanding than the continuous-time set-valued integration approach due to the large numbers of equations generated. To illustrate this, the performance of various techniques towards enclosing the solution sets over a shorter period $\hat{t} \in [0, 0.2] \hat{T}_{\text{ref}}$ are reported in Table 5.5. As shown in Table 5.5a, with recursive preconditioning applied in the DISCR/RECUR approach, splitting the time horizon helps improve the speed. Nevertheless, even the fastest DISCR/RECUR scheme with $\text{LBLK} = 2$ required approximately 10-fold more CPU time than state-of-the-art set-valued integration approach (DINEQ). By profiling the code in CRONOS, it is found that on average over 70% of the run-time was consumed on performing multiplications of Chebyshev polynomial models in both approaches; and the DISCR/RECUR approach entailed approximately 20- to 50-fold more such multiplications than the other. This is because full discretization requires preconditioning of the system of algebraic equations, which does not preserve the sparsity structure. The more blocks comprised in the equation subsystem (i.e. larger LBLK), the larger the number of Chebyshev model multiplications. Another reason is the discretization approach uses a constant step size in the current implementation, whereas set-valued integration uses a time-step adaptation mechanism. On the other hand, as shown in Table 5.5b, the fastest DISCR/RECUR scheme with $\text{LBLK} = 2$ generated remainder bounds in the Chebyshev models that are on average 1 to 2 orders of magnitude tighter than those produced by the DINEQ approach. However, it is important to note that the discretization approach does not include the truncation errors; whereas set propagation methods are verified methods that determine a guaranteed error bound on the exact solution.

5.5.1.2 Comparison with CORA (release 2018)

Algorithms for reachability analysis of continuous nonlinear dynamics are available in the public toolbox CORA [128, 132, 134]. The key idea of CORA is to linearize a nonlinear model using Taylor series expansion, then calculate reachable sets at sample instances, and join the reachable sets using over-approximating zonotopes. The linearization errors are bounded by the zonotopes during flowpipe propagation.

The continuous nonlinear NTP-TL model in Equations (2.3.7a')–(2.3.7f) was implemented in CORA for comparison with CRONOS. A step size $h = 0.002 \hat{T}_{\text{ref}}$ was adopted. Enclosures of the reachable set computed over the time horizons $\hat{t} \in [0, 0.4]$ and $\hat{t} \in [0, 0.5]$ are presented in Figure 5.4; along with 100 randomly simulated trajectories with $\hat{\mathbf{x}}_0 \in [0.9, 1.1] \hat{\mathbf{x}}_{\text{ref}}$. As shown in Figure 5.4a, overestimation of the reachable set is significant where the system is experiencing rapid transitions in the states. Linearization leads to larger approximation errors at such locations. CORA fails to bound the reachable sets soon after

Table 5.5: Performance of various techniques towards enclosing solution sets to problem (5.5.1) at $\hat{t} = 0.2\hat{T}_{\text{ref}}$ with a step size $h = 0.005\hat{T}_{\text{ref}}$ subject to $\hat{\mathbf{x}}_0 \in [0.9, 1.1] \hat{\mathbf{x}}_{\text{ref}}$. A fourth-order Chebyshev polynomial estimator was adopted for both approaches DINEQ and DISCR/RECUR. The modified Gaussian elimination method was applied for bounding algebraic systems. LBLK stands for the number of blocks in each subsystem created during bounding of algebraic equations. A PC with a Intel Core i7-2600 CPU @ 3.60 GHz x 4 processor and running GCC version 7.4.0 was used.

(a) CPU times

	DINEQ	DISCR / RECUR			
		LBLK = 40	LBLK = 10	LBLK = 5	LBLK = 2
t_{CPU} [s]	15.5	720.3	258.4	187.5	146.5
number of Cheb multiplications ¹	2328	105270	56700	48640	43900
fraction of t_{CPU} on Cheb multiplications ²	69.8%	79.3%	77.6%	76.0%	73.7%

¹ This is the number of Chebyshev model multiplications performed.

² This is the percentage of total CPU time that is spent on performing Chebyshev model multiplications.

(b) Remainder bounds in Chebyshev model enclosure on reachable set at $\hat{t} = 0.2\hat{T}_{\text{ref}}$

	$\mathcal{R}_{\hat{P}_{\text{ad}}}$	$\mathcal{R}_{\hat{P}_{\text{d}}}$	$\mathcal{R}_{\hat{P}_{\text{p}}}$	$\mathcal{R}_{\hat{U}_{\text{f}}}$	$\mathcal{R}_{\hat{U}_{\text{p}}}$	$\mathcal{R}_{\hat{P}_{\text{TL},1}}$	$\mathcal{R}_{\hat{P}_{\text{TL},2}}$	$\mathcal{R}_{\hat{P}_{\text{TL},3}}$	$\mathcal{R}_{\hat{P}_{\text{TL},4}}$
DINEQ	1.0E-6	1.0E-7	9.6E-8	1.8E-6	1.9E-6	1.5E-7	6.2E-8	6.0E-8	1.1E-7
DISCR/RECUR $N = 2$	7.8E-7	1.5E-11	2.0E-10	8.6E-9	1.1E-7	3.4E-7	5.5E-8	2.0E-8	1.4E-8

$\hat{t} = 0.4$, as shown in Figure 5.4b. The bound explosion could be attributed to the significant linearization errors, and escalating wrapping effects over time. Reducing the step size to $h = 0.001\hat{T}_{\text{ref}}$, or even smaller, does not help this issue or propagate the reachable set enclosures further. Overall, both the continuous-time and discretized set-valued integration approaches in CRONOS outperform CORA.

5.5.2 Case 2: Reachable set enclosure subject to $\hat{\mathbf{x}}_0 \in [0.5, 2] \hat{\mathbf{x}}_{\text{ref}}$

Having demonstrated the effectiveness of the proposed full discretization scheme in handling relatively small disturbances, the effects of more significant uncertainty on the reachable set enclosures are investigated. The initial states are set to vary from 50% to 200% of their respective reference cyclic steady state positions, i.e. $\hat{\mathbf{x}}_{\text{L}} := 0.5\hat{\mathbf{x}}_{\text{ref}}$, and $\hat{\mathbf{x}}_{\text{U}} := 2\hat{\mathbf{x}}_{\text{ref}}$.

The reachable sets of the five states $\hat{P}_{ad}$, $\hat{P}_d$, $\hat{P}_p$, $\hat{U}_f$, and $\hat{U}_p$ are predicted with the same two approaches (DINEQ and DISCR/RECUR) in CRONOS; and presented in Figure 5.5. The inner-approximating samples are added to the plots in shaded blue regions; and they indicate more complicated system trajectories in this case than in Case 1. Both the approaches DINEQ and DISCR/RECUR fail to enclose the state trajectories tightly after only $\hat{t} = 0.1$. The predicted enclosures explode soon after, near a location of rapid transition. This is particularly notable for the state $\hat{P}_{ad}$, whose trajectories experience more abrupt changes than those of the other states. Using a higher-order Chebyshev polynomial basis or a smaller step size does not alleviate the problem. This bound explosion, which is typical of set-propagation techniques is also affecting the bounds captured with a full-discretization approach.

5.6 Summary

In this chapter, block decomposition and preconditioning for large sparse system of algebraic equations were investigated. After showing that a single block-diagonal preconditioning can lead to significant wrapping effect, a new recursive block preconditioning was proposed. This recursive approach retains the interdependency between blocks and better mitigates the wrapping effect.

Compared against state-of-the-art tools for enclosing the trajectories of a single batch reactor with uncertain temperature, the proposed approach was shown to generate tighter bounds and achieve faster computation at the same time. However, for the more complex NIFTE system with a small uncertainty, the proposed approach costs considerably more computational efforts than state-of-the-art tools for attaining comparable enclosures. Breaking the time horizon into subsystems helps reduce the computational burden, yet generates looser bounds. Overlapping the subsystems may be able to retain some degree of interdependency between blocks, but this needs further investigation. In addition, more efficient discretization based on time-step adaptation is needed. The truncation error in the discretization process also has to be accounted for to make the method rigorous.

Furthermore, both the proposed method and state-of-the-art tool employed fail to produce an effective enclosure of the system behaviour when a large uncertainty is imposed onto the NIFTE. This calls for the development of more effective bounding methods for problems with large uncertainty. One such approach is investigated in the following chapter based on mixed-integer optimization.

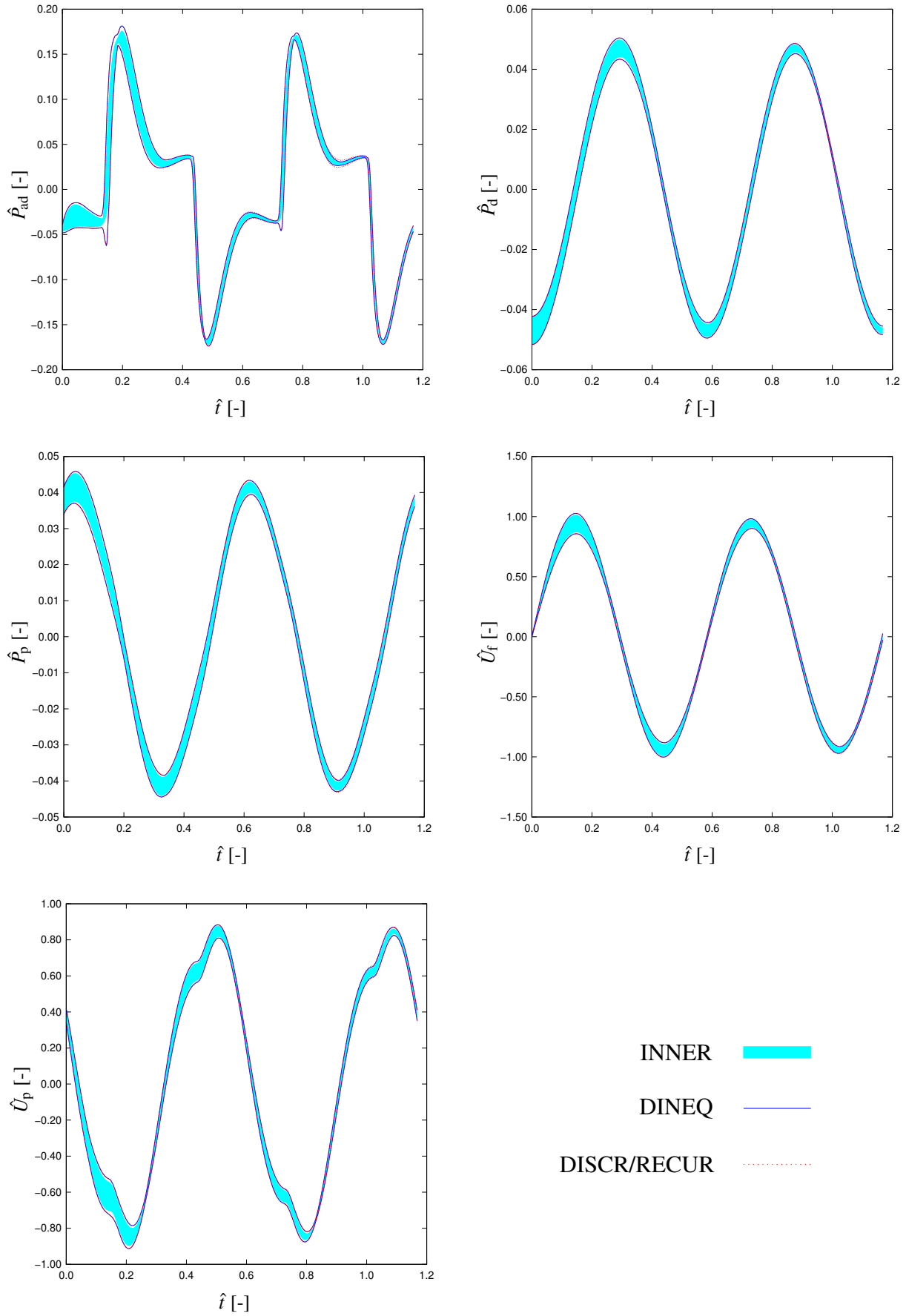


Figure 5.3: Reachable set enclosures for the thermodynamic variables $\hat{P}_{ad}$, $\hat{P}_d$, $\hat{P}_p$, $\hat{U}_f$, and $\hat{U}_p$ for problem (5.5.1) subject to $\hat{\mathbf{x}}_0 \in [0.9, 1.1]$ $\hat{\mathbf{x}}_{ref}$ using: inner approximation (INNER); continuous set-valued integration (DINEQ); and discretized set-valued integration with recursive block decomposition (DISCR/RECUR). 4th-order Chebyshev model-based estimators are used in both DINEQ and DISCR/RECUR; a step size of $0.005\hat{T}_{ref}$ is used in DISCR/RECUR.

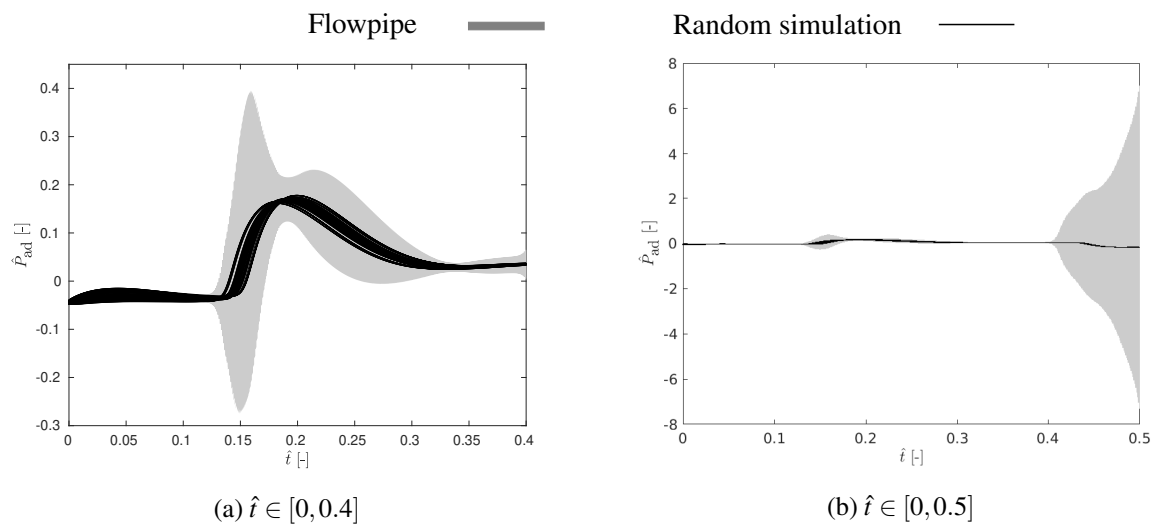


Figure 5.4: Reachable sets for $\hat{P}_{ad}$ using the continuous nonlinear NTP-TL model for the NIFTE device in Equations (2.3.7a'–2.3.7f) with the toolbox CORA (release 2018), subject to $\hat{\mathbf{x}}_0 \in [0.9, 1.1] \hat{\mathbf{x}}_{ref}$.

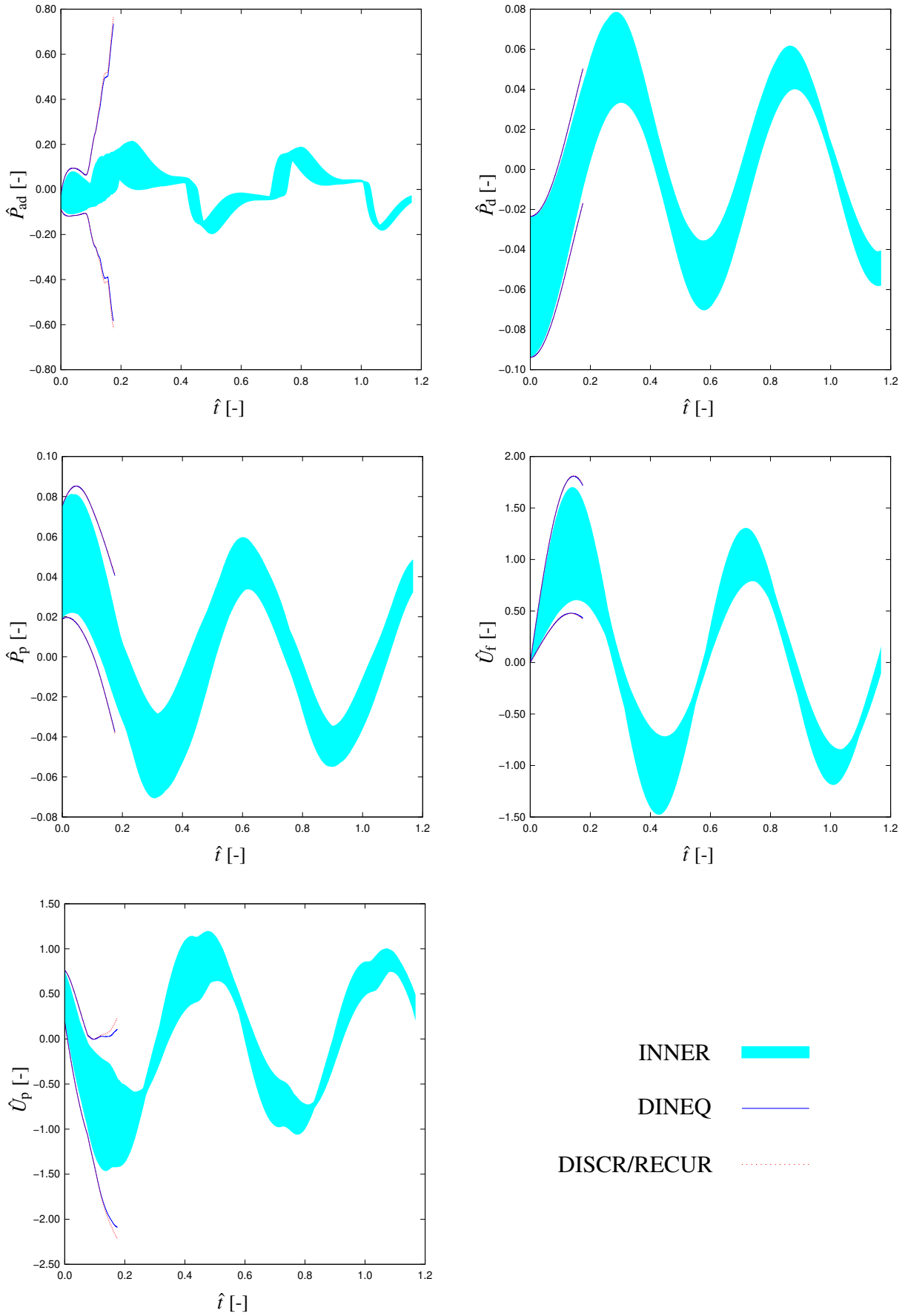


Figure 5.5: Reachable sets for the thermodynamic variables $\hat{P}_{ad}$, $\hat{P}_d$, $\hat{P}_p$, $\hat{U}_f$, and $\hat{U}_p$ for problem (5.5.1) subject to $\hat{\mathbf{x}}_0 \in [0.5, 2]$ $\hat{\mathbf{x}}_{ref}$ using: inner approximation (INNER); continuous set-valued integration (DINEQ); and discretized set-valued integration with recursive block decomposition (DISCR/RECUR). 4th-order Chebyshev model-based estimators are used in both DINEQ and DISCR/RECUR; a step size of $0.005\hat{T}_{ref}$ is used in DISCR/RECUR.

Chapter 6

Mixed-integer Optimization Approach to Bounding of Linear Switched ODEs

6.1 Introduction

Computing bounds on the reachable set of dynamic systems described by ODEs poses formidable challenges in practice. State-of-the-art methods based on set propagation or discretization, such as the ones investigated in Chapter 4, can fail to compute such enclosures for general nonlinear ODEs when the uncertainty is too large or the integration horizon too long. This chapter focuses on a special class of dynamic systems described by switched linear ODEs and develops a new approach based on discretization and mixed-integer linear optimization to compute bounds on their reachable sets. The main motivation behind these developments is that dynamic models of a NIFTE device can be formulated in terms of switched linear ODEs, as explained in Section 2.3 in Chapter 2, thereby providing a new angle of attack to reachability analysis in thermofluidic engines.

This chapter is organized in the following manner. Section 6.2 introduces a basic hybrid automaton model and reviews state-of-the-art methods and tools for the reachability analysis of such hybrid automaton. Section 6.3 presents the new approach based on discretization and mixed-integer optimization, and discusses its implementations in the library CRONOS. The simple case study of a PI-controlled DC motor subject to actuator saturation is investigated in Section 6.4 to demonstrate the capability of this new approach, including comparison with state-of-the-art reachability analysis tools for hybrid systems. The more complex case study of reachability analysis of a NIFTE device based on the PWLTP-TL model is presented next in Section 6.5 to deal with the more complicated, large-scale

linearized NTP-TL model, under scenarios of different degrees of uncertainties. Finally, Section 6.6 summarizes the main findings in this chapter.

6.2 Reachability analysis of hybrid automata (HA)

The PWLTP-TL model of the NIFTE in Equations (2.3.7a')–(2.3.7f) with (2.3.10)) describes a so-called *hybrid automaton*, which involves continuous evolution interspersed by discrete changes. Formally, a hybrid automaton [60, 61] can be represented as $\mathcal{H} = (loc, \mathbf{x}, \mathbf{x}_0, \mathbf{u}, \mathbf{p}, inv, flow, guard, jump)$, where:

- loc that is the set of operational modes, or *locations*, of the hybrid system;
- $\mathbf{x} \in \mathbf{X} \in \mathbb{R}^{n_x}$ is the continuous state vector in the dynamical system, with a set of initial conditions $\mathbf{x}_0 \in \mathbf{X}$;
- $\mathbf{u} \in \mathbf{U} \in \mathbb{R}^{n_u}$ is the continuous input vector;
- $\mathbf{p} \in \mathbf{P} \in \mathbb{R}^{n_p}$ is the parameter vector;
- inv are the set of constraints, or *invariants*, attached to the locations and that restrict the system's behavior within each location;
- $flow$ are the functions that define the continuous dynamics within each location;
- $guard$ are the conditions that dictate the transitions between two locations;
- $jump$ are the functions that describe the discrete state changes during a transition.

According to this formalism, the PWLTP-TL model is a hybrid automaton with linear dynamics, three locations, nine state variables and no input variables. The guards correspond to conditions on the pressure state $\hat{P}_d$ and the transitions are continuous.

The rest of this section reviews recent advances in the reachability analysis of hybrid automata, which is important for understanding the impact of uncertainty, and ensuring safe operation of a system by preventing it from reaching unsafe states across many engineering fields.

A variety of methods have been developed to construct the outer-approximation of the so-called *flowpipe* or *reachable set*, that is the bundle of state trajectories reachable over the time horizon as illustrated in Figure 6.1.

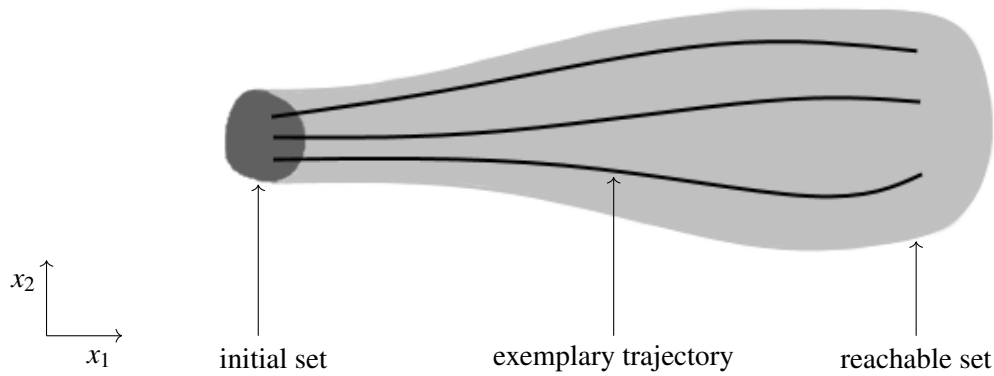


Figure 6.1: An illustration of the reachable set of a 2-dimensional system.

The basic algorithm proceeds by partitioning time horizon into intervals of equal length (see e.g. [155, 156, 128]). Within the continuous parts, reachable sets at the chosen sampling times are constructed stepwise using a given set representation, and then joined together using an over-approximating structure. When a jump is encountered, the reachable sets for the continuous parts intersect the guard sets to give rise to an outer-approximating flowpipe.

Different proven set representations have been considered in this field. Convex set representations have been found useful, particularly for hybrid systems with linear dynamics. Examples of such structures include hyperrectangles [61], convex polytopes [157, 158, 159, 156], ellipsoids [160, 161], support functions [162, 163, 164] and zonotopes [155, 165, 166]. The convex hull for each step is constructed by minimizing the Hausdorff distance between the convex set and the actual reachable set of states. However, the estimation error caused by over-approximation accumulates with flowpipe propagation, resulting in a wrapping effect. Methods that can effectively mitigate this wrapping effect are available for linear dynamical systems [165]. In the basic algorithm for linear systems, the step sizes are fixed and equal. An alternative variable step size method was developed using Bernstein polynomial approximations, a nonconvex representation [167]; that is scalable to high dimensional linear systems.

For hybrid automata with nonlinear dynamics, the use of convex set representations does not produce tight enclosures to the state trajectories in general. Taylor model [90, 168], which comprises a polynomial part and an interval or convex remainder part, is a widely used nonconvex representation. Star sets [169, 170] and polynomial zonotopes [129] are also used in practice. In addition to these direct flowpipe propagation methods, other alternative approaches have been developed for hybrid automata with complex nonlinear dynamics. For example, the original dynamics can be decomposed and reformulated into a set of lower-dimensional ordinary differential equations with no state variables

shared, whose resulting flowpipe is an over-approximation to that of the original system [171]. Hybridization is another approach, where nonlinear dynamics is approximated using linear dynamics, and the linearization errors are evaluated and bounded [172, 128, 173].

A number of computational toolkits have been developed for reachability analysis of different types of continuous and discrete systems, with various methods implemented. For example, SpaceEx [174, 175] is a tool for reachability analysis and safety verification of linear hybrid systems, where the continuous dynamics is enclosed using support functions, and the intersections with guard sets are over-approximated using outer polyhedrons. HyPro [176], another tool that focuses on reachability of linear hybrid systems, implements a unified interface for a selection of convex state set representations, including boxes, polytopes, zonotopes and support functions. HyPro is found to be as computationally competent as SpaceEx using inexact number arithmetic.

As for nonlinear hybrid systems, FLOW* [177] is a useful tool that constructs Taylor model flowpipes. Adaptive choices of step sizes and polynomial orders are supported in FLOW*. Lastly, CORA [132, 133, 134] is a state-of-the-art toolkit that can handle both linear and nonlinear systems. For continuous dynamics, flowpipe sampling and propagation can be performed using either zonotope or polynomial zonotope as set representations. When a jump is encountered, the intersection between the reachable set and the guard set is represented using polytopes, and bounded using over-approximating zonotopes which are propagated further.

It is important to note that for all aforementioned fixed step methods and tools, the step size should be sufficiently small to ensure correct jumps between locations; yet not so small to mitigate the computational burden. Moreover, it is necessary to ensure sufficient overlap between adjacent invariant sets; otherwise, the algorithm may enter an infinite loop by jumping between adjacent locations (chattering).

6.3 Methodology

An alternative approach to set-propagation techniques for bounding the reachable set of hybrid automata entails the discretization of the continuous dynamics into a large set of algebraic equations and subsequently enclosing their solutions, similar in essence to the approach used in Chapter 5 for nonlinear ODEs. However, instead of using bounding techniques for implicit algebraic equations, optimization methods are considered herein. A number of effective and efficient optimization tools are available for solving large-scale linear and nonlinear problems, such as IBM ILOG CPLEX

Optimization Studio [109] and Gurobi Optimizer [110]. The capabilities of such tools can be exploited to handle fully discretized dynamic systems.

Motivated by the uncertainty propagation problem in the NIFTE device, the focus herein is on a special class of linear hybrid automata, whereby the state evolution is continuous yet nonsmooth due to the presence of a saturation function. Formally, the saturation function is defined as:

$$\text{sat}(\xi, \xi^{\min}, \xi^{\max}) = \begin{cases} \xi^{\min}, & \text{if } \xi \leq \xi^{\min} \\ \xi^{\max}, & \text{if } \xi \geq \xi^{\max} \\ \xi, & \text{if otherwise,} \end{cases} \quad (6.3.1a)$$

$$(6.3.1b)$$

$$(6.3.1c)$$

with scalars $\xi^{\min} < \xi^{\max}$ and $\xi \in \mathbb{R}$. Notice that this function can also be expressed in terms of min and max operators, e.g.:

$$\begin{aligned} \text{sat}(\xi, \xi^{\min}, \xi^{\max}) &= \max(\xi^{\min}, \min(\xi^{\max}, \xi)) \\ &= \min(\xi^{\max}, \max(\xi^{\min}, \xi)). \end{aligned}$$

The switched linear dynamic system itself is written in the form:

$$\dot{\mathbf{x}}(t) = \mathbf{A} \cdot \mathbf{x}(t) + \mathbf{b} \cdot \text{sat}(\alpha^T \mathbf{x}(t) + \beta^T \mathbf{p} + \gamma, \xi^{\min}, \xi^{\max}) + \mathbf{c} \cdot \mathbf{p} + \mathbf{d} \quad (6.3.2)$$

where $\mathbf{x} \in \mathbb{R}^{n_x}$ are the state variables; $\mathbf{p} \in \mathbb{R}^{n_p}$, the time-invariant (uncertain) parameters; $\mathbf{A} \in \mathbb{R}^{n_x \times n_x}$, $\mathbf{b} \in \mathbb{R}^{n_x}$, $\mathbf{c} \in \mathbb{R}^{n_x \times n_p}$ and $\mathbf{d} \in \mathbb{R}^{n_x}$ are constant matrixes or vectors; and $\alpha^T \mathbf{x}(t) + \beta^T \mathbf{p} + \gamma$ is the linear saturation function between $\xi^{\min}$ and $\xi^{\max}$, with constant vectors $\alpha \in \mathbb{R}^{n_x}$, $\beta \in \mathbb{R}^{n_p}$ and scalar $\gamma \in \mathbb{R}$.

In the formalism of hybrid automata, this dynamic system (6.3.2) has 3 locations, with linear flow functions in each location and no time-varying inputs. All of the guards are furthermore linear conditions in $\mathbf{x}(t)$ and $\mathbf{p}$, and the state remains continuous across a transition. A graphical representation is given in Figure 6.2.

Notice that this framework can be readily extended to switched linear dynamic systems with multiple saturation functions, thereby increasing the number of locations in the hybrid automaton. More generally, switched linear systems with any number of min or max terms may also be handled in the same way.

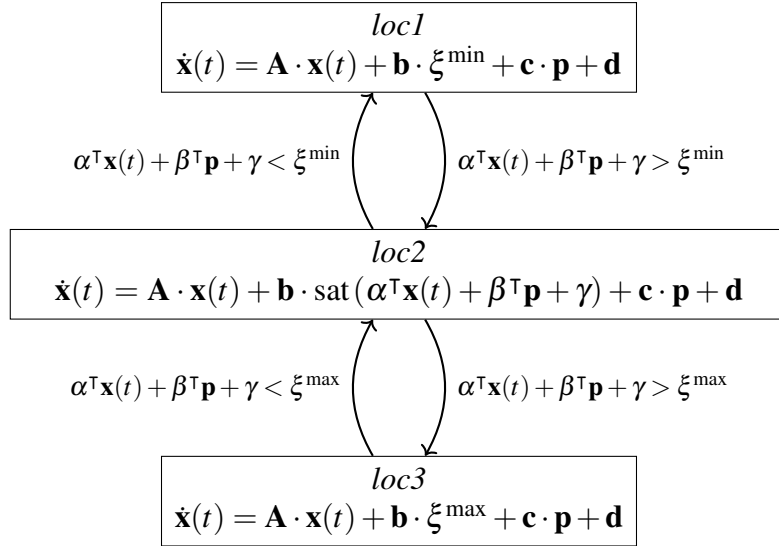


Figure 6.2: Linear automaton representation of the nonsmooth dynamic system 6.3.2 with saturations.

6.3.1 Time discretization approach

Of the possible numerical discretization methods to solve ODEs, the focus herein is on Runge-Kutta methods, for consistency with Chapter 5 (see Section 5.2.1). Given a time grid $t_k := k \cdot h$ for a time step $h > 0$ and $k = 0, \dots, N$ steps; a discretization of the switched linear dynamic system (6.3.2) using a Runge-Kutta formula with S stages (see Table 5.1) approximates the state $\mathbf{x}(t_k) \approx \mathbf{X}_k$ as follows:

$$\begin{aligned} \mathbf{x}_{k+1} &= \mathbf{x}_k + h \sum_{s=1}^S b_s \varphi_s \\ \text{with : } \varphi_s &= \mathbf{A} \cdot \left[\mathbf{x}_k + h \sum_{j=1}^S a_{sj} \varphi_j \right] \\ &\quad + \mathbf{b} \cdot \text{sat} \left(\alpha^T \left[\mathbf{x}_k + h \sum_{j=1}^S a_{sj} \varphi_j \right] + \beta^T \mathbf{p} + \gamma, \xi^{\min}, \xi^{\max} \right) \\ &\quad + \mathbf{c} \cdot \mathbf{p} + \mathbf{d}. \end{aligned} \tag{6.3.3}$$

This discretization yields a large number of equality constraints, where the only nonlinearity is due to the presence of min and max terms. The strategy used herein is to reformulate all of these terms exactly using binary variables in order to transform the constraint into mixed-integer linear form.

6.3.2 Mixed-integer linear optimization formulation

A well-known reformulation of min and max terms are mixed-integer constraints involves applying the so-called big- M method [178]. In the case of the max term $x_k = \max(x_i, x_j)$, this reformulation yields

$$\begin{aligned} x_k &\geq x_i \\ x_k &\geq x_j \\ x_k &\leq x_i + yM \\ x_k &\leq x_j + (1 - y)M \end{aligned}$$

where $y \in \{0, 1\}$ is an auxiliary binary variable; and M is a large enough constant. In practice, provided that the participating variables are bounded in the range $x_i \in [x_i^L, x_i^U]$ and $x_j \in [x_j^L, x_j^U]$, one can select the value of M automatically as follows:

$$\begin{aligned} x_k &\leq x_i + y(x_j^U - x_i^L), \\ x_k &\leq x_j + (1 - y)(x_i^U - x_j^L). \end{aligned}$$

Likewise, the min term $x_k = \min(x_i, x_j)$ with $x_i \in [x_i^L, x_i^U]$ and $x_j \in [x_j^L, x_j^U]$ can be reformulated exactly as the set of mixed-integer linear inequalities:

$$\begin{aligned} x_k &\leq x_i \\ x_k &\leq x_j \\ x_k &\geq x_i + y'(x_j^L - x_i^U) \\ x_k &\geq x_j + (1 - y')(x_i^L - x_j^U) \end{aligned}$$

where $y' \in \{0, 1\}$ is another binary variable. Moreover, by combining the min and max reformulations, one can easily derive an exact reformulation of saturation term $\text{sat}(\xi, \xi^L, \xi^U)$ using two auxiliary binary variables.

Overall, all of the ODE residual constraints, with the saturation terms linearized using binary variables, can be combined as constants in a mixed-integer linear optimization problem, where the objective function consists of minimizing or maximizing the value of a chosen state variable at a given time t_N . For example, to find the lower bound of the state x_i at t_N , the following minimization problem

is formulated:

$$\begin{aligned}
& \min_{\mathbf{x}_0, \mathbf{y} \in \{0,1\}} : x_i(t_N) \\
& \text{s.t.} : \mathbf{x}_{k+1} = \mathbf{f}(\mathbf{x}_k, \mathbf{y}, \mathbf{p}), \quad k = 0, \dots, N \\
& \quad \mathbf{x}(t_0) = \mathbf{x}_0, \\
& \quad \mathbf{x}_0 \in [\mathbf{x}_0^L, \mathbf{x}_0^U],
\end{aligned}$$

where the right-hand side functions $\mathbf{f}$ corresponds to the discretized system in Equation (6.3.3), and $\mathbf{y}$ are binary variables introduced to reformulate the saturation functions. A maximization problem can be formulated in a similar manner to compute the upper bound of x_i at t_N . For N steps in the discretization and a Runge-Kutta formula of order S , a total of $2 \cdot N \cdot S$ binary variables are introduced. It is important to note that this is not a validated approach, as the truncation errors due to discretization are discarded. Hence, while the state bounds are exact for the discretized problem, they are only approximations to the actual bounds of the switched linear dynamic system. The truncation errors are nonetheless mitigated by refining the step size h .

6.3.3 Implementation in CRONOS

The proposed optimization-based bounding approach is implemented in the library CRONOS and builds on several classes. An example of code is reported in Appendix C; which is the implementation of the case study of a PI-controlled DC motor in Section 6.4.

Two main steps are involved in this implementation. The first step entails discretization of the nonsmooth ODEs, using an explicit Runge-Kutta scheme. It is automated using methods in the class ODEBND EXPAND (see Section 5.3.3). In the second step, the ODE residuals are embedded into an optimization problem in order to bound the range of a variable at a given time. In particular, the min and max terms in the residuals are reformulated using mixed-integer linear program (MILP). This reformulation is automated in the class POLIMG of the toolkit MC++ [80] (see Section 4.3) and integrated in the class NLGO of CRONOS which provides an interface to state-of-the-art MILP solvers, such as IBM ILOG CPLEX Optimization Studio [109] and Gurobi Optimizer [110].

6.4 Case study: PI-controlled DC motor

6.4.1 Problem definition

The DC motor of a cooling fan as depicted in Fig. 6.3 [179] is considered. The equivalent circuit of the motor consists of a voltage source, a resistor, an inductor and a back electromotive force (EMF). The back EMF, e (V) depends on the rate of rotation and can be expressed as

$$e(t) = K\dot{\theta}(t) \quad (6.4.1)$$

where K (Vsrad^{-1}) is the back-EMF constant; and $\dot{\theta}$ (rads^{-1}) is the rotational speed. The dynamic equation for the circuit is

$$v(t) = Ri(t) + L\frac{di}{dt}(t) + K\dot{\theta}(t) \quad (6.4.2)$$

where v (V) is the input voltage; i (A) is the armature current; R (Ω) is the resistance; and L (H) is the inductance. The torque on the rotor, T (Nm) is proportional to the armature current, i (A) and can be expressed as

$$T(t) = Ki(t) \quad (6.4.3)$$

The dynamic equation for the rotor is

$$Ki(t) = J\frac{d\dot{\theta}}{dt}(t) + b\dot{\theta}(t) \quad (6.4.4)$$

where J (kgm^2) is the total moment of inertia of the rotor and fan blades; and b (Nm srad^{-1}) is the viscous resistance due to air resistance and bearing friction.

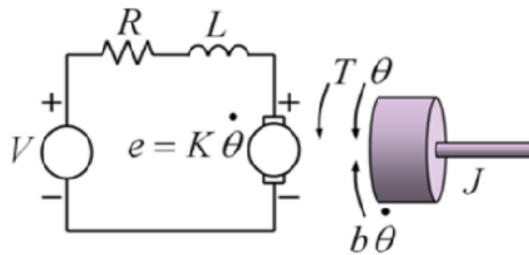


Figure 6.3: DC motor and equivalent circuit.

The DC motor is controlled in order to maintain a desired rotational speed, $\dot{\theta}^{\text{ref}}$. A simple PI

controller is used to manipulate the input voltage, which is constrained in the range $v(t) \in [v^{\min}, v^{\max}]$.

The dynamic equations for the controller with actuator saturation can be written as

$$v(t) = \max \{v^{\min}, \min \{v^{\max}, K_P [\dot{\theta}^{\text{ref}}(t) - \dot{\theta}(t)] + K_I q(t)\}\} \quad (6.4.5)$$

$$\frac{dq}{dt}(t) = \dot{\theta}^{\text{ref}}(t) - \dot{\theta}(t) \quad (6.4.6)$$

where the proportional and integral gains are set to $K_P = 40$ and $K_I = 10$.

In practice, the DC motor experiences disturbance from the environment. Assuming a time-invariant disturbance in the torque, w , the new equations for the torque are therefore

$$T(t) = Ki(t) + w. \quad (6.4.3')$$

$$Ki(t) = J \frac{d\dot{\theta}}{dt}(t) + b\dot{\theta}(t) \quad (6.4.4')$$

Rearranging the model equations (6.4.2, 6.4.3', 6.4.4', 6.4.6) yields a dynamic model comprised of three differential state variables and three ODEs that describe the behavior of the DC motor,

$$\frac{di}{dt}(t) = \frac{1}{L} (v(t) - Ri(t) - K\dot{\theta}(t)), \quad (6.4.7a)$$

$$\frac{d\dot{\theta}}{dt}(t) = \frac{1}{J} (Ki(t) - b\dot{\theta}(t) + w), \quad (6.4.7b)$$

$$\frac{dq}{dt}(t) = \dot{\theta}^{\text{ref}}(t) - \dot{\theta}(t), \quad (6.4.7c)$$

and $v(t)$ is given by Equation (6.4.5).

Table 6.1: Parameter values in the model for the DC motor.

Parameter	Value	Unit	Parameter	Value	Unit
K	8.5×10^{-2}	Vsrad^{-1}	$v_{\min}$	-6	V
R	5.5×10^{-1}	Ω	$v_{\max}$	6	V
L	2.5×10^{-2}	H	K_P	10	-
J	9.0×10^{-2}	kg m^2	K_I	20	-
b	5.0×10^{-2}	Nm srad^{-1}	$\dot{\theta}^{\text{ref}}$	1	srad^{-1}

The aim is to enclose the reachable set of the rotational speed $\dot{\theta}$ with a range of disturbance $w \in$

$[-0.1, 0.1]$, subject to the *a priori* bounds $i(t) \in [-20, 20]$, $\dot{\theta}(t) \in [-5, 5]$ and $q(t) \in [-5, 5]$; and initial conditions at the rest position, i.e. $i(0) = \dot{\theta}(0) = q(0) = 0$.

6.4.2 Implementation in CRONOS

The switched-linear dynamic system (6.4.7) is discretized using an explicit Runge-Kutta scheme and a constant step size. Then, the residuals are linearized and embedded into an MILP problem. In this way, with the classical 4th-order Runge-Kutta scheme and a step size $h = 0.02s$, computing the lower or the upper bound of any of the states, $\dot{\theta}$, q or i at time $t = 1s$ entails solving an MILP with 400 binary variables. The solver Gurobi Optimizer (version 8.1.1) [110] is used here. The full script for the implementation in CRONOS is reported in Appendix C.

6.4.3 Implementation in CORA

The switched linear dynamic system (6.4.7) is reformulated into a linear hybrid automaton with 3 locations (*loc1*, *loc2*, *loc3*), the same 4 state variables ($i, \dot{\theta}, q, w$) in each location and no input variables.

$$\frac{di}{dt} = \begin{cases} -\frac{R}{L}i - \frac{K}{L}\dot{\theta} + \frac{v_{\min}}{L}, & \text{if } K_P\dot{\theta}^{\text{ref}} - K_P\dot{\theta} + K_Iq \leq v^{\min} \\ -\frac{R}{L}i - \frac{K+K_P}{L}\dot{\theta} + \frac{K_I}{L}q + \frac{K_P\dot{\theta}^{\text{ref}}}{L}, & \text{otherwise} \\ -\frac{R}{L}i - \frac{K}{L}\dot{\theta} + \frac{v_{\max}}{L}, & \text{if } K_P\dot{\theta}^{\text{ref}} - K_P\dot{\theta} + K_Iq \geq v^{\max} \end{cases} \quad \begin{matrix} (6.4.8a) \\ (6.4.8b) \\ (6.4.8c) \end{matrix}$$

$$\frac{d\dot{\theta}}{dt} = \frac{K}{J}i - \frac{b}{J}\dot{\theta} + \frac{w}{J} \quad (6.4.8d)$$

$$\frac{dq}{dt} = \dot{\theta}^{\text{ref}} - \dot{\theta}, \quad (6.4.8e)$$

$$\frac{dw}{dt} = 0, \quad (6.4.8f)$$

Note that the disturbance w is treated as an extra state variable with zero dynamics (Equation 6.4.8f) and uncertain initial condition $w(0) \in [-0.1, 0.1]$. The invariant sets for the three locations are defined using the *a priori* bounds. In order to ensure a sufficient degree of overlap between any two adjacent locations, the guards in the CORA model are relaxed slightly from those in Equations (6.4.8a – 6.4.8c). The same sampling step of 0.02 s as in CRONOS is used in CORA for the flowpipe propagation.

6.4.4 Reachable set enclosures

The reachable set enclosures computed for (6.4.8) using the mixed-integer optimization (MIP) approach in CRONOS and as a linear hybrid automaton (LHA) in CORA are shown in Figure 6.4. In the former approach, the classical 4th-order Runge-Kutta scheme is used with a step size of 0.02 s. In the latter approach, flowpipe is generated by propagating zonotopes; and specifically in this case, rectangles with the width of the chosen step size, that is also 0.02s. Both approaches predict that the rotational speed $\dot{\theta}$ first experiences an overshoot and then stabilizes at a slightly faster speed than the desired speed, which is 1 rad s^{-1} . With the same sampling step size, the MIP approach generates tighter enclosures than the LHA approach.

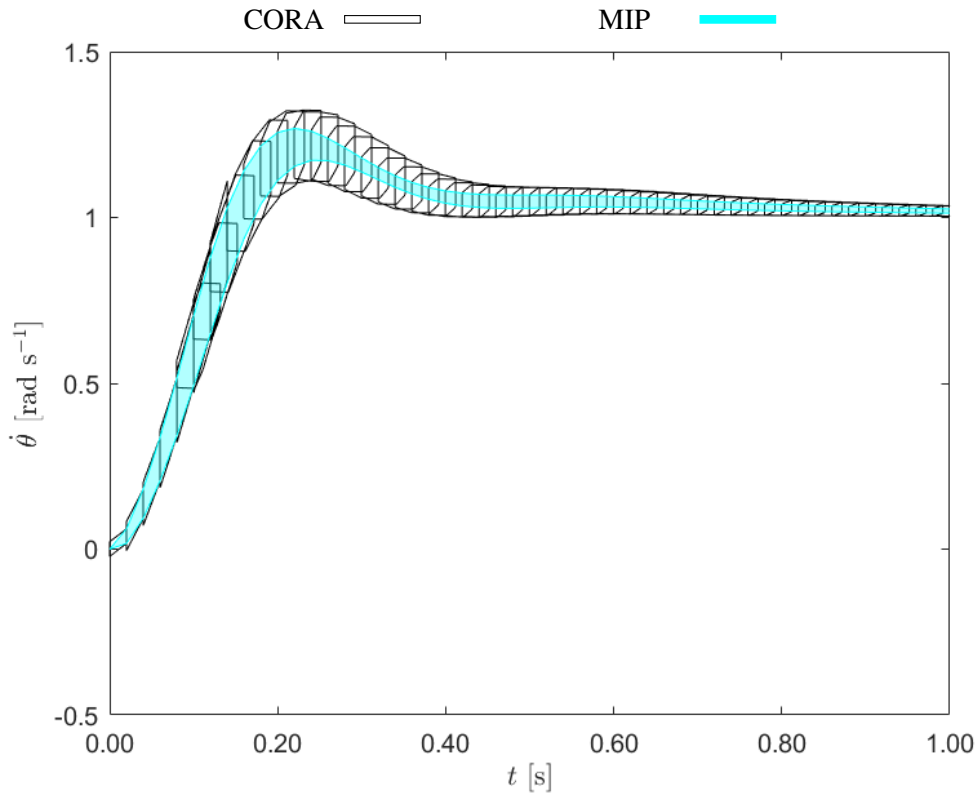


Figure 6.4: Flowpipe enclosure for $\dot{\theta}$ in (6.4.7) computed as a linear hybrid automaton (LHA) in CORA (release 2018) and as a mixed-integer optimization problem (MIP) in CRONOS, subject to $w \in [-0.1, 0.1]$. The classical 4-th order Runge-Kutta scheme with a step size of 0.02 s is adopted in the MIP approach

In the MIP approach, using Runge-Kutta schemes of different orders or step sizes impacts the calculated state bounds. This is due to the presence of discretization errors, which are not accounted for in this approach. The lower and upper bounds for the rotational speed at $t = 1 \text{ s}$, i.e. $\dot{\theta}_{\min}$ and $\dot{\theta}_{\max}$ respectively, are calculated using explicit 1st- to 4-th order Runge-Kutta schemes with a step size of either 0.02 s or 0.01 s. These results are reported in Table 6.2, and compared against those calculated in

CORA with a step size of 0.02 s. Again it is observed that the MIP approach produces tighter enclosures for $\dot{\theta}$.

Notice that in the MIP approach, some low-order Runge-Kutta schemes yield seemingly tighter state bounds than high-order schemes. For example, with the step size of 0.02 s, using 1st- or 2nd-order Runge-Kutta schemes results in tighter lower bounds than using 3rd- or 4-th order schemes. However, this seemingly "better" bound is an artefact and is attributed to the failure to account for discretization errors in the MIP approach. To use a lower-order Runge-Kutta scheme, a smaller step size should be adopted to counteract larger discretization errors. This is exemplified by the computed bounds using 2nd-order Runge-Kutta scheme with a smaller step size of 0.01 s. The new calculated $\dot{\theta}_{\min}$ is now in agreement (depending on the precision adopted here) with those calculated using 3rd- and 4-th order Runge-Kutta schemes.

Comparing the CPU time (t_{CPU}) required for the MIP and the LHA approaches, as reported in Table 6.2, it is found that the LHA approach is generally much faster than the MIP approach if the same step size is used. Yet, there is still potential for the latter to be as computationally competitive as the former if a coarse discretization scheme is adopted while producing a tight state enclosure at the same time. For example, with the step size of 0.02 s, the LHA approach is only $1/3$ faster than the MIP approach using 2nd-order Runge-Kutta scheme.

Table 6.2: Performance of using a linear hybrid automaton approach (LHA) in CORA and the mixed-integer optimization (MIP) approach in CRONOS towards enclosing solution sets to (6.4.7) at $t = 1$ s. Combinations of 1st- to 4th-order explicit Runge-Kutta schemes and step sizes of 0.01 s and 0.02 s are adopted in the MIP approach. A PC with a Intel Core i7-2600 CPU @ 3.60 GHz x 4 processor and the optimization solver Gurobi (version 8.1.1) with G++ version 7.4.0 was used.

	LHA	MIP, step = 0.02 s				MIP, step = 0.01 s			
	step = 0.02 s	RK4	RK3	RK2	RK1	RK4	RK3	RK2	RK1
$\dot{\theta}_{\min}$ [srad ⁻¹]	1.0039	1.0130	1.0130	1.0132	1.0157	1.0130	1.0130	1.0130	1.0144
$\dot{\theta}_{\max}$ [srad ⁻¹]	1.0343	1.0257	1.0257	1.0258	1.0271	1.0257	1.0257	1.0257	1.0265
CPU [s]	2.16	28.74	6.48	3.18	0.39	474.11	82.41	45.58	2.69

However, it is important to note that the MIP approach is significantly more computationally

expensive if a continuous flowpipe enclosure of the solution set is desired over the time horizon, instead of reachable set enclosures at particular instants. This is because the MIP approach needs to solve two MILP problems at each individual sampling point, which are hard to warm start from previous time points, leading to accumulating CPU time; yet the LHA approach naturally results in a continuous flowpipe during reachable set propagation.

To analyze further the computational complexity of the mixed-integer optimization approach, the CPU time (t_{CPU}) required to calculate $\hat{\theta}_{\max}$ at various time instances $t \in [0, 1]$ s is recorded and shown in Figure 6.5. The classical 4th-order Runge-Kutta method and a step size of 0.02 s are used for the analysis. The number of discretized algebraic equations and binary variables grow linearly with time horizon, the the computational time grows exponentially. This is in agreement with the fact that MILPs are NP-hard, and have exponential worst-case time complexity. Despite this complexity, it is impressive that certain MILP problem with several hundred binary variables can be solved to optimality within only a few seconds only. This is encouraging for attempting more complex case studies.

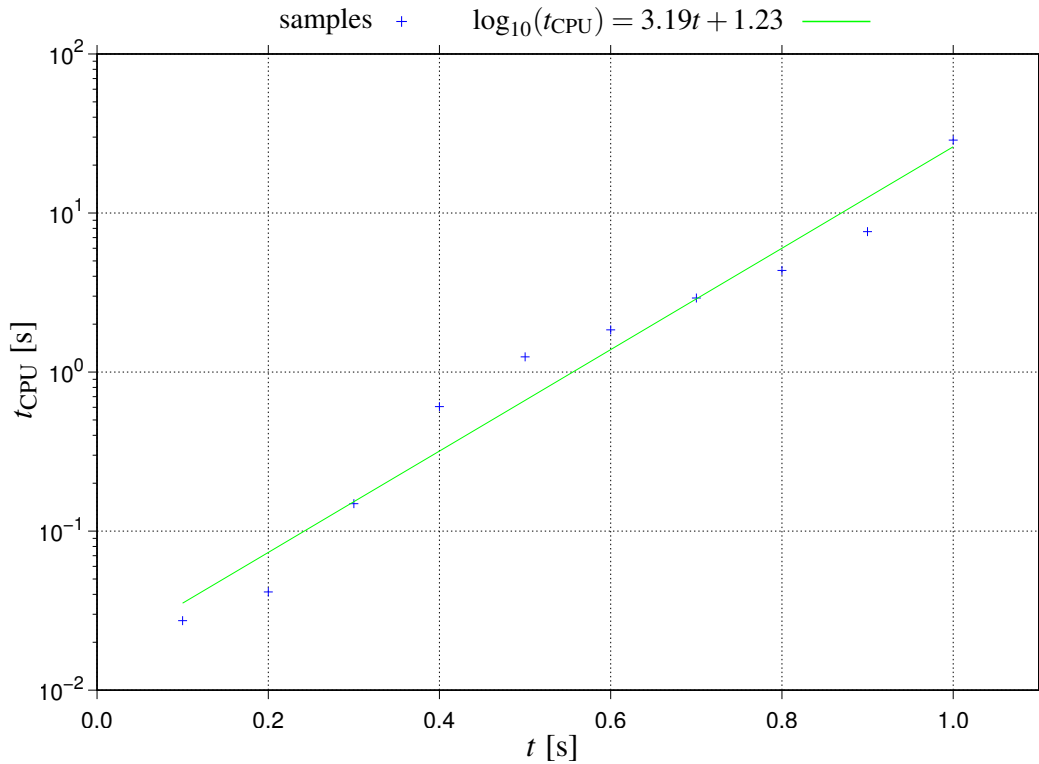


Figure 6.5: CPU time (t_{CPU}) to solve for the upper bound of $\hat{\theta}$ in (6.4.7) at various time instances (t) with the mixed-integer optimization approach, subject to $w \in [-0.1, 0.1]$. The discretization uses 4-th order Runge-Kutta scheme with a step size of 0.02 s. A PC with a Intel Core i7-2600 CPU @ 3.60 GHz x 4 processor and the optimization solver Gurobi (version 8.1.1) with G++ version 7.4.0 was used.

6.5 Case study: Piecewise-linear thermofluidic engine

As a final case study, the problem of propagating uncertainty in the thermofluidic engine NIFTE is revisited. In order to apply the mixed-integer optimization approach to bound the reachable set, the PWLTP-TL model of the NIFTE in Equations (2.3.7a'–2.3.7f, 2.3.10) is considered. This is in contrast to the case study in Section 5.5 of the previous chapter where the NTP-TL model (2.3.7a')–(2.3.7f) was used. The uncertainty is in the initial conditions of the 9 state variables, $\hat{\mathbf{x}}(0) \in [\hat{\mathbf{x}}_L, \hat{\mathbf{x}}_U]$.

6.5.1 Implementation in CRONOS

For consistency with the discretization used in Section 5.5, a classical (4th order) Runge-Kutta scheme is considered, with a step-size of $h = 0.005\hat{T}_{\text{ref}}$. For a time horizon $\hat{t} \in [0, \hat{T}_{\text{ref}}]$, the resulting MILP problem comprises 1600 binary variables.

6.5.2 Implementation in CORA

The PWLTP-TL model (2.3.7a'–2.3.7f, 2.3.10) is converted to a linear hybrid automaton with 3 locations, the same 9 thermodynamic state variables ($\hat{P}_{\text{ad}}, \hat{P}_{\text{d}}, \hat{P}_{\text{p}}, \hat{U}_{\text{f}}, \hat{U}_{\text{p}}, \hat{P}_{\text{tl},1}, \hat{P}_{\text{tl},2}, \hat{P}_{\text{tl},3}$, and $\hat{P}_{\text{tl},4}$) in each location, and no input variables. The transitions between the locations are continuous and triggered when the state $\hat{P}_{\text{d}}$ crosses the values $1/\kappa_{\text{th}}$ or $-1/\kappa_{\text{th}}$. These values are slightly relaxed in the CORA implementation in order to allow for a sufficient, yet not too large, overlap between adjacent locations - this overlap was determined by trial-and-error here. Finally, the same step-size of $h = 0.005\hat{T}_{\text{ref}}$ as in CRONOS is used in CORA for consistency. Different step sizes are attempted in both mixed-integer optimization and hybrid automaton approaches to investigate the effect.

6.5.3 Verification of model implementations

Prior to reachability analysis of the NIFTE system in the presence of uncertainty, the reformulations, which lead to the implementation of a linear hybrid automaton in CORA and a mixed-integer optimization problem, are first validated.

At the nominal operating conditions (i.e. without any uncertainty), time trajectories of the NIFTE are computed using three approaches: (1) numerical ODE integration of NTP-TL model in (2.3.7a'–

2.3.7f, 2.3.8) (labeled NLIN); (2) solution of PWLTP-TL model (2.3.7a'–2.3.7f, 2.3.10) as a linear hybrid automaton (labeled LHA); (3) solution of PWLTP-TL model (2.3.7a'–2.3.7f, 2.3.10) as a mixed-integer optimization in CRONOS (labeled MIP). The predictions are compared in Figure 6.6; where it is shown that all three approaches produce about identical results under nominal conditions and thereby validate all reformulations and implementations in CRONOS and CORA.

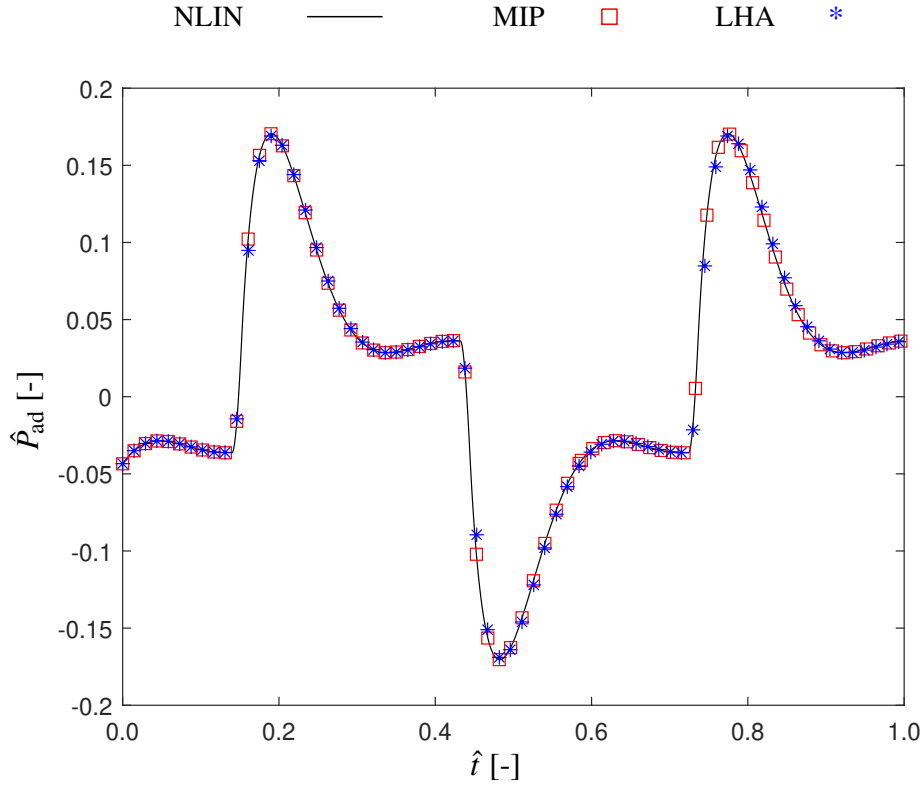


Figure 6.6: Comparing the predicted trajectories of $\hat{P}_{ad}$ using the original nonlinear NTP-TL model (2.3.7a'–2.3.7f, 2.3.8) (NLIN); the PWLTP-TL model (Equations 2.3.7a'–2.3.7f, 2.3.10) as a linear hybrid automaton (LHA) in CORA (release 2018); and the PWLTP-TL model (2.3.7a'–2.3.7f, 2.3.10) as a mixed-integer optimization problem (MIP) in CRONOS.

6.5.4 Reachable set enclosures

Having checked the various model implementations, the next step is to compute enclosures on a NIFTE's state trajectories subject to uncertainty in initial conditions using the PWLTP-TL model (2.3.7a'–2.3.7f, 2.3.10). The same set of operating parameters and uncertainty ranges is considered as in Section 5.5, where the reachability analysis was based on the original nonlinear NTP-TL model (2.3.7a'–2.3.7f, 2.3.8).

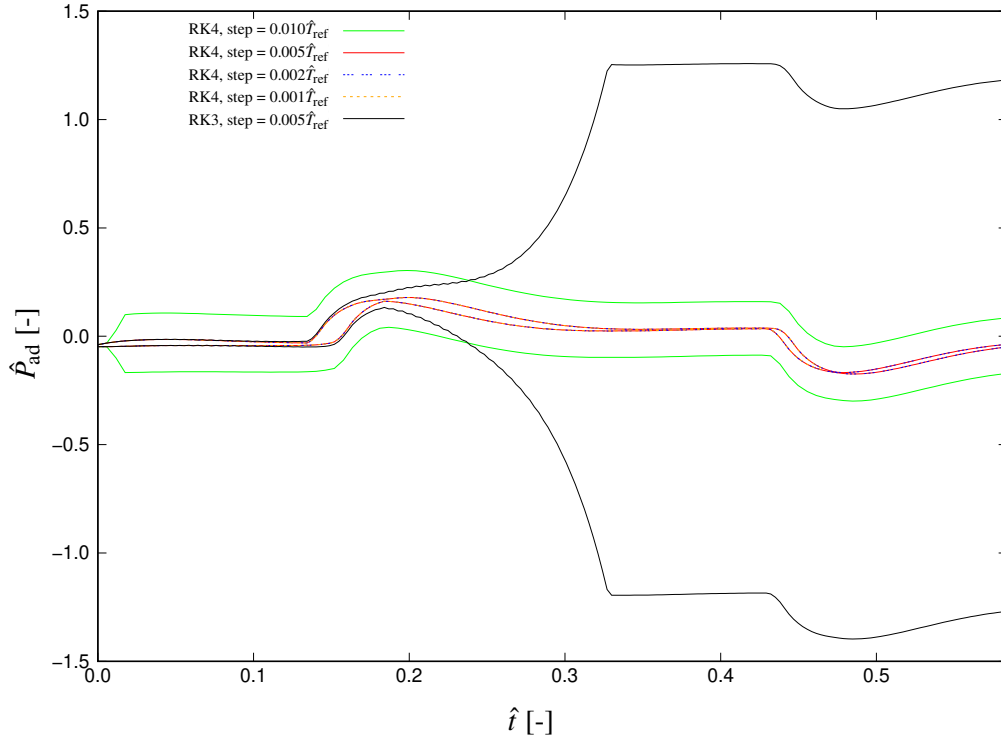


Figure 6.7: The effects of various discretization schemes on reachable set predictions for $\hat{P}_{ad}$. The linearized NTP-TL model (Equations 2.3.7a'–2.3.7f with 2.3.10) is assumed and the proposed min-max approach is applied.

6.5.4.1 Case 1: reachable set enclosure subject to $\hat{\mathbf{x}}_0 \in [0.9, 1.1] \hat{\mathbf{x}}_{ref}$

Mixed-integer optimization approach in CRONOS

The mixed-integer reformulation of the min and max terms in the ODE discretization requires *a priori* state bounds in order to apply the big- M constraint. Here, the *a priori* enclosure $[-1, 1]$ is used for $\hat{P}_{ad}$, $\hat{P}_d$, $\hat{P}_p$, $\hat{P}_{tl,1}$, $\hat{P}_{tl,2}$, $\hat{P}_{tl,3}$ and $\hat{P}_{tl,4}$, and $[-5, 5]$ for $\hat{U}_f$ and $\hat{U}_p$.

The effect of using various step sizes and Runge-Kutta schemes are compared in Figure 6.7 over one reference cycle. With a step size of $0.005\hat{T}_{ref}$, using a higher fourth-order Runge-Kutta method (RK4) prevents the bound of $\hat{P}_{ad}$ from diverging at an early stage of propagation, in contrast to third-order Runge-Kutta method (RK3). Moreover, with a RK4 scheme, using a coarser step size of $0.010\hat{T}_{ref}$ shows instability in the bounds early on. Instead, reducing the step size to $0.002\hat{T}_{ref}$ or $0.001\hat{T}_{ref}$ has a negligible effect on the predicted bounds (despite requiring drastically more computational resources). Based on this cursory analysis, the results of the mixed-integer optimization approach in the rest of this subsection consider an RK4 scheme with a step size of $0.005\hat{T}_{ref}$.

Reachable set enclosures of the five thermodynamic states, i.e. $\hat{P}_{ad}$, $\hat{P}_d$, $\hat{P}_p$, $\hat{U}_f$ and $\hat{U}_p$, computed with the mixed-integer optimization approach (MIP) are shown in Figure 6.8. They are furthermore compared with enclosures generated using the NTP-TL model (2.3.7a'–2.3.7f, 2.3.8). These latter enclosures are produced in two ways: (1) inner-approximation constructed via computing an extensive number of uncertainty scenarios in the given initial condition range (INNER); (2) outer-approximation using state-of-the-art continuous set-propagation techniques based on generalized differential inequalities (DINEQ) here with 4th-order Chebyshev model arithmetic and ellipsoidal remainders [43]. Further details regarding the variants INNER and DINEQ can be found in Section 5.5.

Both the MIP and DINEQ methods predict comparably tight reachable set enclosures for the states, with the former slightly outperforming the latter in particular at the turning points (e.g. at $\hat{t} \approx 0.18$ and $\hat{t} \approx 0.76$ for $\hat{P}_{ad}$). This is reflected in Table 6.3, where the computed bounds at $\hat{t} = \hat{T}_{ref}$ by the two methods are compared. However, the MIP method requires 20-times more CPU time than the DINEQ method in this scenario. This observation is expected given the large size of the mixed-integer problem that is generated by the full ODE discretization (1600 binary variables). It is also worth noting that all three methods describe the effect of uncertainty in initial conditions wearing off over time, which is expected since $\hat{\mathbf{x}}_{ref}$ and $\hat{T}_{ref}$ correspond to a stable limit cycle of the simulated NIFTE device.

Table 6.3: Bound and CPU time comparison in the uncertainty scenario $\hat{\mathbf{x}}_0 \in [0.9, 1.1] \hat{\mathbf{x}}_{ref}$ between: (1) the DINEQ approach with the NTP-TL model; and (2) the MIP approach using the PWLT-TL model. A fourth-order Chebyshev model with ellipsoidal remainder is propagated in the DINEQ approach. A 4th-order Runge-Kutta scheme and a step size $h = 0.005\hat{T}_{ref}$ is used for the discretization in the MIP approach.

	t_{CPU} [s]	$\Delta\hat{P}_{ad}$	$\Delta\hat{P}_d$	$\Delta\hat{P}_p$	$\Delta\hat{U}_f$	$\Delta\hat{U}_p$
DINEQ	22.76	1.27E-02	5.14E-03	4.83E-03	3.92E-02	4.46E-02
MIP	400.12	1.21E-02	4.88E-03	4.55E-03	3.57E-02	4.11E-02

	$\Delta\hat{P}_{tl,1}$	$\Delta\hat{P}_{tl,2}$	$\Delta\hat{P}_{tl,3}$	$\Delta\hat{P}_{tl,4}$
DINEQ	6.93E-04	2.42E-03	3.47E-03	5.71E-03
MIP	6.43E-04	2.23E-03	3.28E-03	5.62E-03

Samples of the required CPU time (t_{CPU}) needed to compute the reachable set enclosures at various time instances $\hat{t}$ are reported in Figure 6.9. The linear relation between $\hat{t}$ and $\log_{10}(t_{CPU})$ indicates that the complexity of the mixed-integer optimization approach grows exponentially with the number of

binary variables in the discretized problem.

Linear hybrid automaton approach in CORA

The flowpipe enclosure generated for the state $\hat{P}_{ad}$ with $\hat{\mathbf{x}}_0 \in [0.9, 1.1] \hat{\mathbf{x}}_{ref}$ is shown in Figure 6.10. An extensive number of inner-approximating samples are taken using the original NTP-TL model and presented in the shaded blue areas (labeled INNER) in the same figure for comparison. A step-size $h = 0.005\hat{T}_{ref}$ is first attempted; but it is found to result in inaccurate initialization of the flowpipe. Therefore, the step size is reduced to $h = 0.002\hat{T}_{ref}$ for flowpipe propagation using zonotope enclosures over each step size.

The results in Figure 6.10 show that CORA is able to capture the general trend of $\hat{P}_{ad}$, but the overestimation grows over time. This is due to overestimation building up during stiff transitions between the locations of the hybrid automaton. Reducing the step size to $0.001\hat{T}_{ref}$ or $0.0005\hat{T}_{ref}$ brings little benefit in reducing the overestimation.

6.5.4.2 Case 2: Reachable set enclosure subject to $\hat{\mathbf{x}}_0 \in [0.5, 2] \hat{\mathbf{x}}_{ref}$

A larger uncertainty is imposed on the initial state of the NIFTE system, which is now allowed to vary between 50% and 200% of its reference values $\hat{\mathbf{x}}_{ref}$.

Mixed-integer optimization approach in CRONOS

Having showcased that the mixed-integer optimization approach is able to closely approximate the reachable sets subject to a small disturbance in initial conditions, a larger degree of uncertainty is now imposed on the PWLTP-TL model whereby the initial conditions are set to vary between 50% and 200% from the reference cyclic steady state values. The same *a priori* state bounds as in **Case 1** are found to be suitable in the present uncertainty scenario as well, i.e. $[-1, 1]$ for $\hat{P}_{ad}$, $\hat{P}_d$, $\hat{P}_p$, $\hat{P}_{tl,1}$, $\hat{P}_{tl,2}$, $\hat{P}_{tl,3}$ and $\hat{P}_{tl,4}$, and $[-5, 5]$ for $\hat{U}_f$ and $\hat{U}_p$.

Reachable set enclosures computed over one reference cycle period, $\hat{t} \in [0, \hat{T}_{ref}]$, using the mixed-integer optimization approach for $\hat{P}_{ad}$, $\hat{P}_d$, $\hat{P}_p$, $\hat{U}_f$ and $\hat{U}_p$ are shown in Figure 6.11. The inner-approximation is constructed using the NTP-TL model and depicted in shaded blue areas on the plots (labeled INNER).

Unlike the set-propagation methods that could not propagate the state bounds beyond $\hat{t} = 0.2$ (Figure 5.5), the mixed -integer optimization method based on the PWLTP-TL model successfully computes tight approximations to the actual reachable sets in the NIFTE system. This latter method does not suffer from the wrapping effect, as opposed to the set-propagation approaches. Nevertheless, the errors due to discretization of the PWLTP-TL model are not accounted for and therefore the method is not validated.

Similar to **Case 1**, the effect of uncertainty diminishes gradually with time, as reflected in the reduced size of state enclosures at $\hat{t} = \hat{T}_{\text{ref}}$ compared to those at $\hat{t} = 0$. The complexity of computing reachable set enclosures increases again exponentially with the length of the time horizon.

Linear hybrid automaton approach in CORA

The same step size $h = 0.002\hat{T}_{\text{ref}}$ is adopted. The flowpipe enclosures computed by CORA shown in Figure 6.12 fails to effectively enclose the state trajectories after approximately $\hat{t} = 0.1$. Reducing the step size to $0.001\hat{T}_{\text{ref}}$ or $0.0005\hat{T}_{\text{ref}}$ does not prevent the flowpipe to diverge after $\hat{t} = 0.1$.

6.6 Summary

In this chapter, the focus was on reachability analysis of switched linear dynamic systems, which conventionally are treated as hybrid automata. An alternative approach based on mixed-integer optimization has been developed, where the dynamics are first discretized using explicit Runge-Kutta schemes to yield a large system of algebraic equations which are nonlinear due to min and max terms only. These min and max terms are subsequently reformulated as mixed-integer linear cuts by introducing auxiliary binary variables. Finally, the cuts are embedded into a mixed-integer linear program to find the minimum or maximum value of a chosen state at a given time. This contribution is furthermore automated in the C++ toolkit CRONOS.

When applied to a small-scale switched linear dynamic system, the mixed-integer optimization generated comparable enclosures to state-of-the-art tools for hybrid automata, CORA in this case. It could also be competitive with these tools in terms of computational burden when a low order discretization scheme was used. The advantage of the proposed method became more prominent when the larger-scale NIFTE system was considered based on the PWLTP-TL model. The benchmark tool CORA suffered seriously from wrapping effects, while the mixed-integer optimization approach was still

able to tightly enclose the thermodynamic states under large uncertainty in the initial conditions. As for the complexity, computational burden of solving the mixed integer linear program grows exponentially with the length of the time horizon. Nevertheless, it is still impressive that such large-scale MILP models with over 1000 binary variables can be solved in a relatively short time.

As future directions, it would be useful to compare with implicit discretization schemes, such as implicit Runge-Kutta, as these may enable larger time steps without compromising the numerical accuracy of the approximations. In principle, there is no extra complexity in applying an implicit Runge-Kutta scheme compared to an explicit scheme, because they both result in residuals with the same number of min and max terms.

Another promising direction would be combining the mixed-integer optimization approach with a set-propagation method, similar in essence to the propagation methods developed earlier in Chapter 5. One particular area of improvement here would be the ability to warm-start the MILP solution as the time horizon is progressively extended.

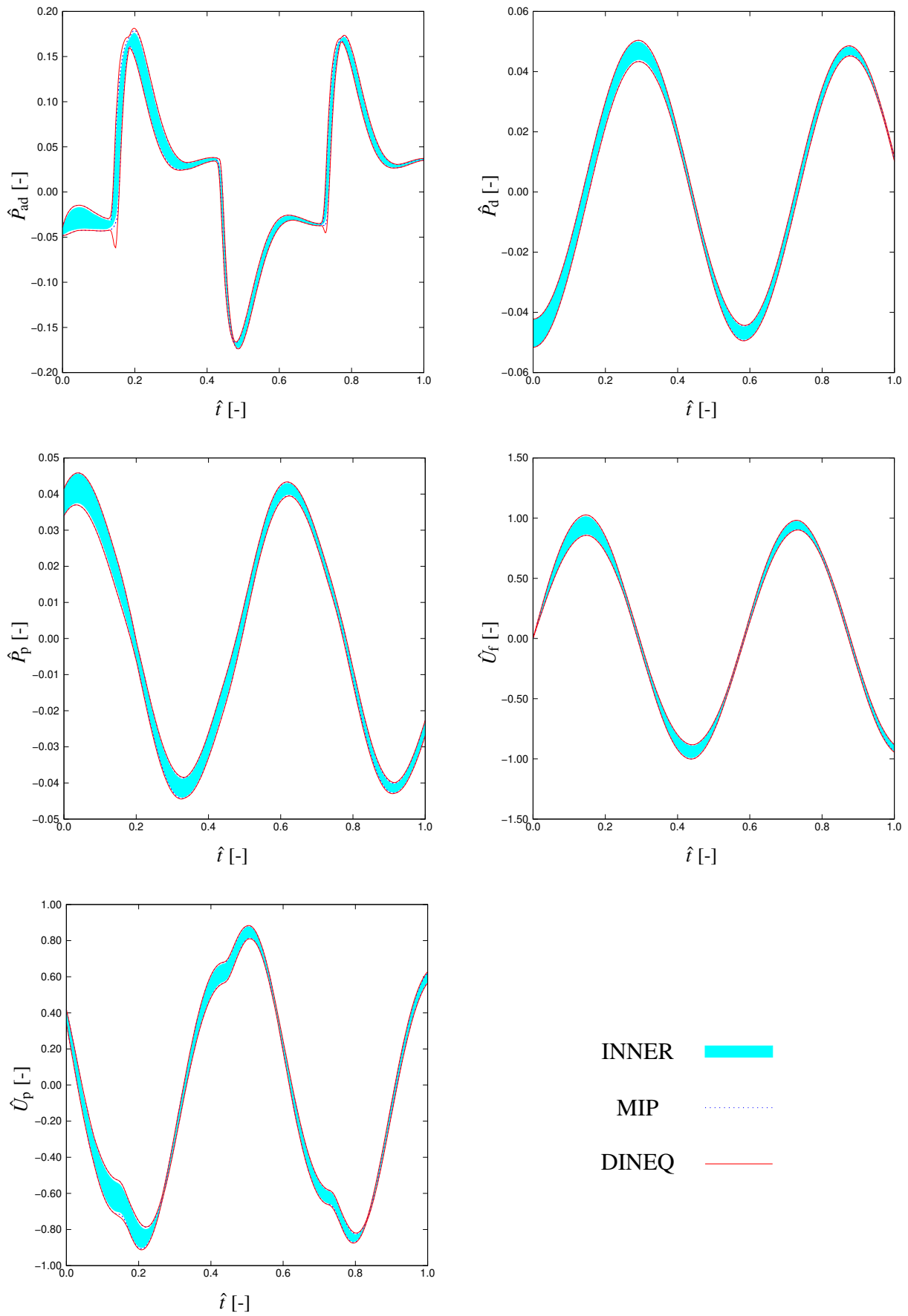


Figure 6.8: Reachable sets for $\hat{P}_{ad}$, $\hat{P}_d$, $\hat{P}_p$, $\hat{U}_f$, and $\hat{U}_p$ in the NIFTE subject to $\hat{\mathbf{x}}_0 \in [0.9, 1.1] \hat{\mathbf{x}}_{ref}$ using: (1) the NTP-TL model with inner-approximation (INNER); (2) the NTP-TL model with set-propagation methods (DINEQ); and (3) the PWLTP-TL model with the mixed-integer optimization approach (MIP).

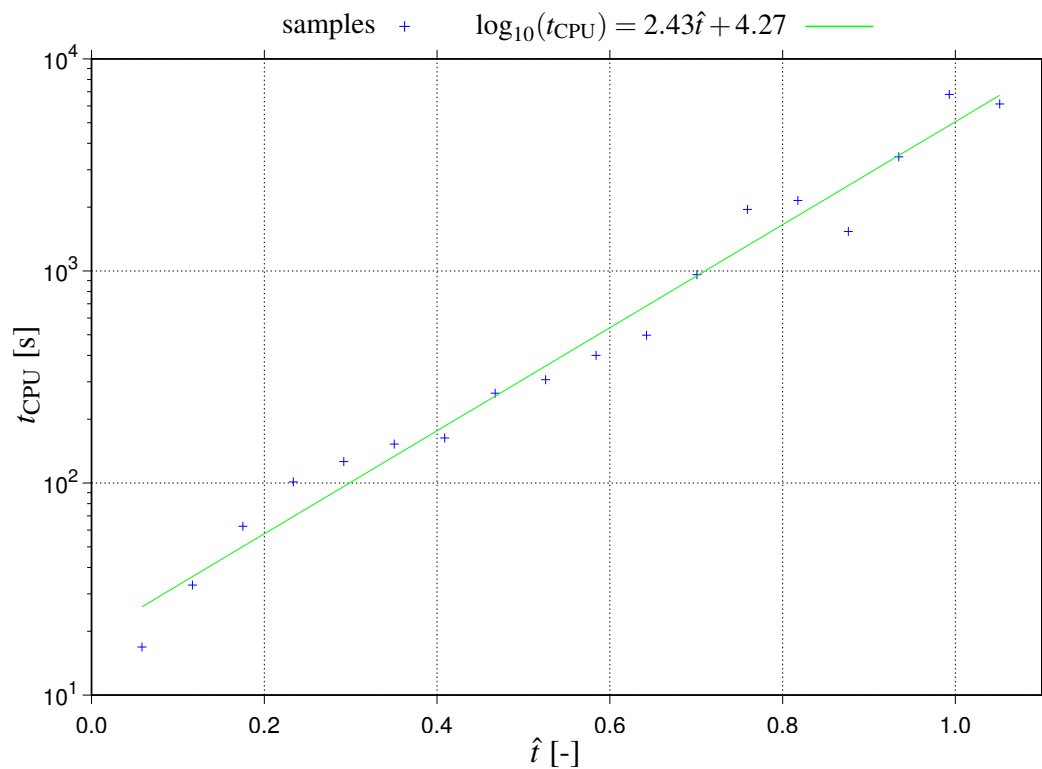


Figure 6.9: CPU time (t_{CPU}) to solve for the upper bound of $\hat{P}_{\text{ad}}$ in the PWLTP-TL model (2.3.7a'–2.3.7f, 2.3.10) at various time instances $\hat{t}$ with the mixed-integer optimization approach, subject to $\hat{\mathbf{x}}_0 \in [0.9, 1.1] \hat{\mathbf{x}}_{\text{ref}}$. The discretization uses 4-th order Runge-Kutta scheme with a step size of 0.005. A PC with a Intel Core i7-2600 CPU @ 3.60 GHz x 4 processor and the optimization solver Gurobi (version 8.1.1) with G++ version 7.4.0 was used.

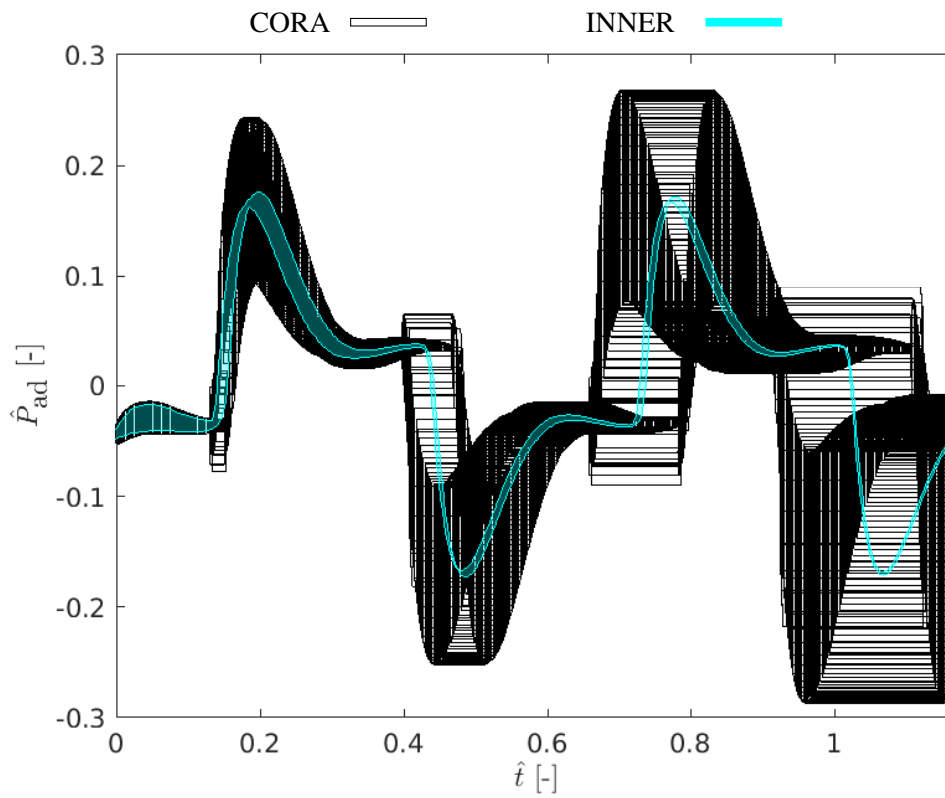


Figure 6.10: Flowpipe enclosure for $\hat{P}_{ad}$ using the PWLTP-TL model (2.3.7a'–2.3.7f, 2.3.10) computed with the toolbox CORA (release 2018), subject to $\hat{\mathbf{x}}_0 \in [0.9, 1.1] \hat{\mathbf{x}}_{ref}$.

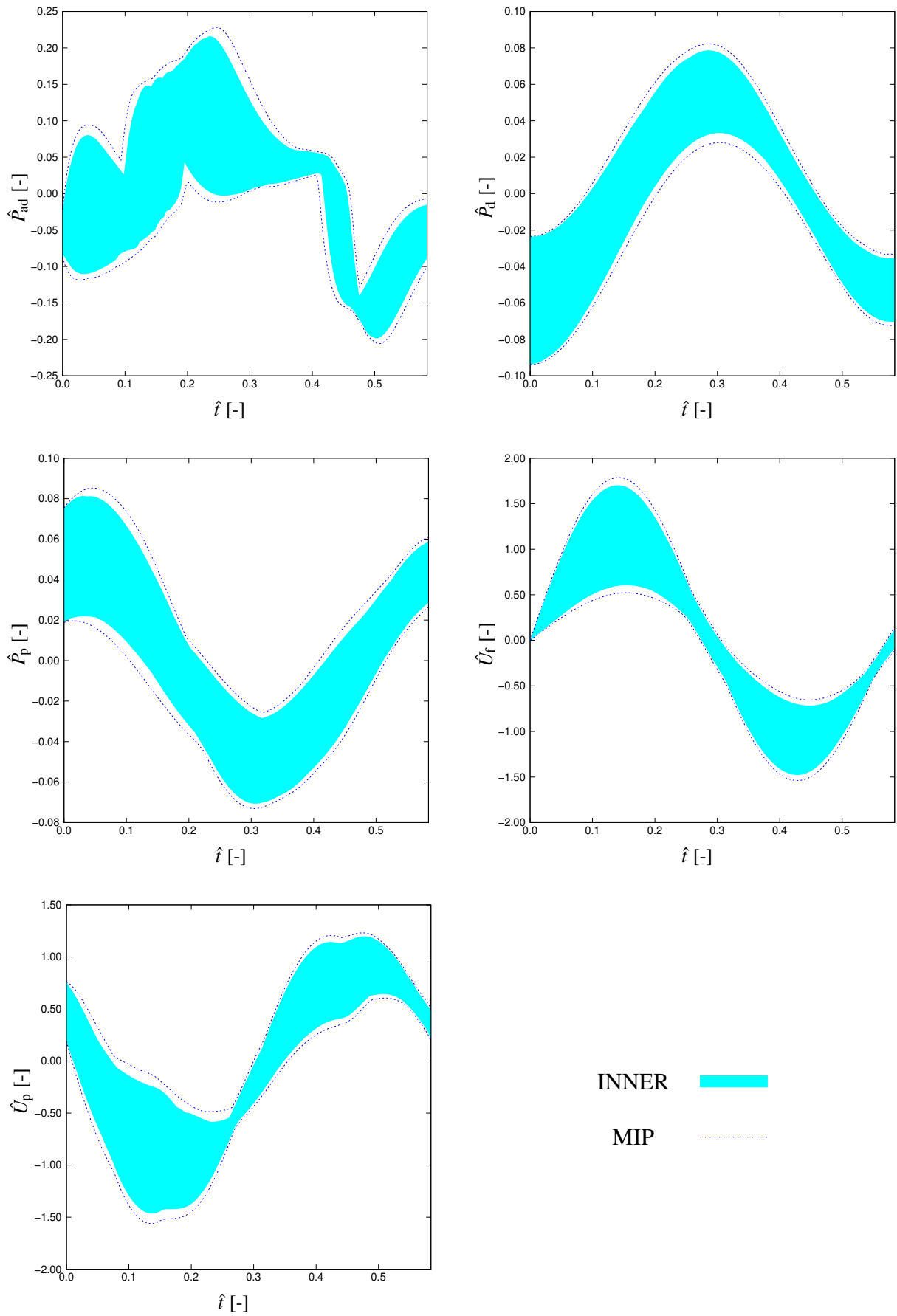


Figure 6.11: Reachable sets for $\hat{P}_{ad}$, $\hat{P}_d$, $\hat{P}_p$, $\hat{U}_f$, and $\hat{U}_p$ in the NIFTE subject to $\hat{\mathbf{x}}_0 \in [0.5, 2] \hat{\mathbf{x}}_{ref}$ using: (1) the NTP-TL model with inner-approximation (INNER); and (2) the PWLTP-TL model with the mixed-integer optimization approach (MIP).

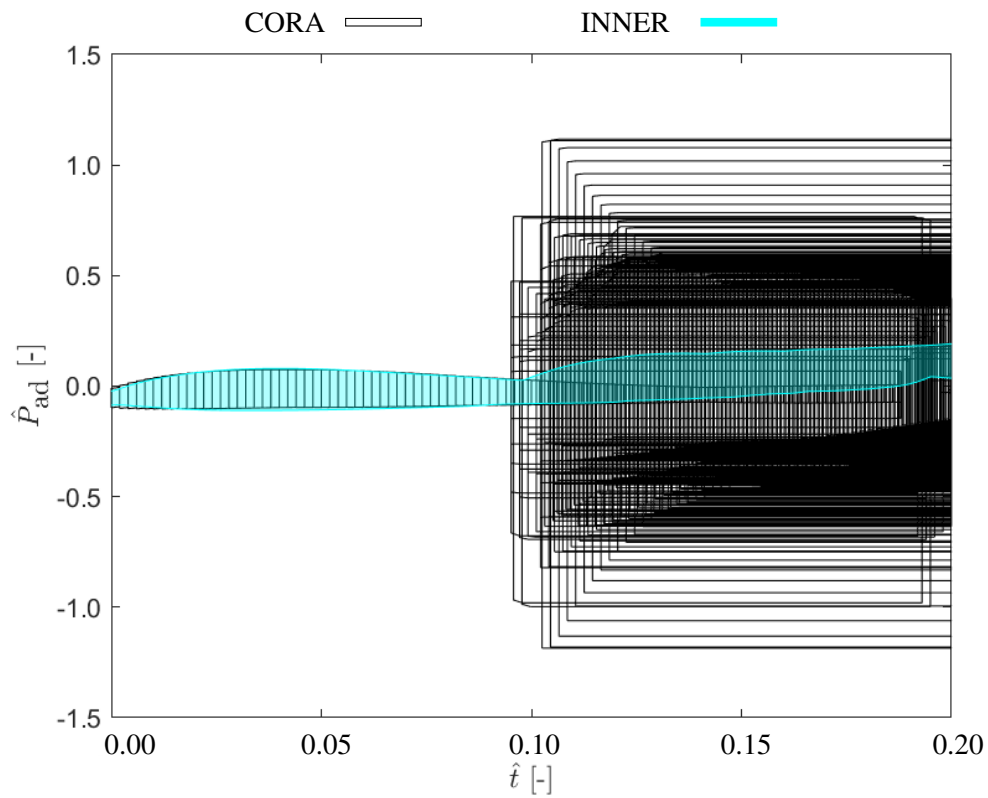


Figure 6.12: Flowpipe enclosures for $\hat{P}_{ad}$ using the PWLTP-TL model (2.3.7a'–2.3.7f, 2.3.10) computed with the toolbox CORA (release 2018), subject to $\hat{\mathbf{x}}_0 \in [0.5, 2] \hat{\mathbf{x}}_{\text{ref}}$.

Chapter 7

Conclusions

7.1 Summary of contributions

The overall objective of this thesis has been the development and application of advanced algorithms for the analysis and optimization of thermofluidic engines, both under nominal conditions and in the presence of uncertainty. The specific focus has been on the non-inertive-feedback thermofluidic engine (NIFTE).

The first contribution in Chapter 2 has been an extension of the existing NTP model of the NIFTE to account for irreversible thermal losses, resulting in a new model called NTP-TL as well as a piecewise linear variant called PWLTP-TL. The same set of assumptions for the fluid behavior and the temperature profile was adopted. By expanding the existing electrical analogy of the NIFTE device, nine additional electrical parameters and four state variables were introduced. It was demonstrated that the adapted model predicted 75% lower device exergetic efficiency under nominal operating conditions, without modifying the oscillation frequency significantly. This indicated an effective incorporation of the thermal-loss components into the NTP-TL and PWLTP-TL models.

The NTP-TL model then formed the backbone for the system analysis conducted in Chapter 3. A dynamic optimization-based methodology was devised for detecting the cyclic steady states of the NIFTE. A novel finding for thermofluidic engines is that multiple cyclic steady states can coexist in certain engine configurations. Moreover, an unstable CSS may significantly outperform a stable CSS in terms of power output or exergetic efficiency for the same set of operating conditions. A multi-objective optimization was also conducted by varying the most sensitive design parameters of the NIFTE

device. These parameters can be achieved by modifying the geometric properties of the key parts of the NIFTE, i.e. the displacer and power cylinders, the feedback connection tube, and the heat source and the heat sink. By constructing the Pareto frontiers between the device exergetic efficiency and the useful work output given different combinations of the design parameters, it was found that a near-optimal performance can be achieved by adjusting three key parameters only, which are determined by the dimensions of the feedback tube and the power cylinder. The recommended parameter configuration yielded more than 50% improvement in either the useful work output or the exergetic efficiency than the nominal configuration. These improvements are particularly relevant for the development of NIFTE pumps to support small-holder irrigation in the developing world [180].

The co-existence of multiple cyclic steady states in the NIFTE has stressed the need for efficient global optimization and global analysis methods. However, the current capability of global dynamic optimization technology is insufficient. Therefore, the rest of the thesis has focused on developing more effective and efficient techniques for ODE bounding, which plays a central role in global dynamic optimization.

Chapter 5 has investigated a new bounding approach based on ODE discretization which heavily relies upon effective bounding of algebraic equation systems. The main contribution has been the development of bounding techniques that can take advantage of an equation system's block structure in order to accelerate the computation. A particular emphasis has been on effecting preconditioning methods which are essential to algebraic equation bounding. Since a simple approach based on diagonal block preconditioning can lead to significant wrapping of the solution enclosures, a new recursive block preconditioning approach has been developed. This new approach is equivalent to the original approach and results in significant CPU time reduction. This recursive block preconditioning has been applied to both iterative methods (Gauss-Seidel and Krawczyk methods) and Gaussian elimination in the case in which the blocks are partially linear in the dependent variables. This new approach has been implemented in the class AEBND as part of the toolkit CRONOS.

Another main contribution in Chapter 5 has been automating the discretization of nonlinear ODE systems and bounding of the large-scale systems of residual equations using the new block decomposition technique using recursive preconditioning. This approach has been automated in another class ODEBND_EXPAND of the toolkit CRONOS with two possible set representations, namely intervals and polynomial models. It is important to note that this is not a validated approach for ODE bounding as truncation errors associated with the discretization was not accounted for. The performance of the proposed discretization method was compared against state-of-the-art set-propagation methods. In small scale problems, the former was shown to outperform the latter in

terms of computational efficiency with careful choices of discretization scheme and number of blocks per subproblem. However, in handling larger scale problems such as the NIFTE problem, the former method could result in a higher computational burden than the latter due to the large numbers of algebraic operations during the preconditioning. To reduce the size of the discretized system, the step size would need to be enlarged; calling for more numerically stable discretization schemes as well as adaptive time-stepping.

In the final contribution chapter, an alternative approach has been pursued to bounding of the dynamic trajectories of the NIFTE by exploiting the switched linear dynamic model, PWLTP-TL, where the temperature profile is described as a piecewise linear function. A new mixed-integer optimization approach has been developed where the switched linear ODEs are first discretized, and the resulting residuals with min and max terms are reformulated into mixed-integer cuts for reachability analysis via the introduction of auxiliary binary variables. This construction was again automated in the toolkit CRONOS, by building upon the new class ODEBND_EXPAND in combination with the existing class NLGO for nonlinear global optimization.

With a relatively low level of uncertainties imposed, this new bounding approach has been shown to be as effective as state-of-the-art set-propagation methods. But with a larger level of uncertainty that causes the set-propagation methods to fail, the mixed-integer optimization approach managed to closely enclose the NIFTE trajectories over a complete cycle. This advantage, however, comes at the cost of significant computational burden, since the CPU time scales exponentially with the time horizon considered. The CPU time is also affected by the level of uncertainty and the *a priori* state bounds.

7.2 Future directions

The findings about thermofluidic engines and limitations of the proposed methods present opportunities for a number of areas, some of which are detailed below.

The first recommended follow-ups to the work presented in this thesis include the execution of dedicated experimental campaigns to further calibrate the NIFTE models, and the implementation of optimized NIFTE designs to verify the model-based predictions. In doing so, one should keep in mind the simplicity of the electrical analogy models used relative to the complexity of actual NIFTE devices. As more experimental data of NIFTE become available, it would be interesting to attempt the development of hybrid semi-parametric models that enable more reliable predictions by combining a physics-based models with data-driven components [181].

For the NIFTE technology to be adopted for low-grade heat recovery at larger scale, minimization of the irreversible thermal losses via further design modifications would be worth investigating, given that these losses entail a dramatic reduction in the exergetic efficiency and work output. Control schemes may be introduced that take advantage of the high performance associated with unstable cyclic steady states under certain conditions. Furthermore, the proposed optimization-based approach may be extended to system analysis of other thermofluidic engines with more complex designs, e.g. the so-called Up-THERM engine which also experiences a nonlinear temperature profile [16].

Global optimization of nonlinear dynamical models remains a very challenging task. Although the discretization techniques proposed developed through this thesis proved useful in some scenarios to bound the solution of either nonlinear ODEs or switched-linear ODEs, they remain prone to fail or require excessive computational power. An alternative approach may be to approximate the nonlinear dynamical models for thermofluidic engines with response surfaces via non-dynamical surrogate models, which can be constructed using sampling methods [182, 183]. An accurate mechanical models for the engine is essential; numerous simulation results subject to an extensive range of operating conditions are needed to establish and refine the surrogate model. The optimization-based approach for cyclic steady state detection could be incorporated as the preliminary step to determine if multiple limit cycles are present, prior to constructing a realistic surrogate model. Effective derivative-free optimization methods, such as Bayesian optimization algorithms, may also be investigate to find globally optimal solutions [184, 185].

Furthermore, in this thesis the focus has been on the use of explicit discretization schemes on reachability analysis of nonlinear dynamic systems. It would be worth considering implicit schemes too, as they may allow larger time steps while retaining numerical stability compared to explicit schemes. Recently, Wilhelm et al. [125] has demonstrated that combining implicit discretization schemes and single block decomposition helps reduce computational burden for stiff parametric ODE-IVPs. Therefore, as a future direction for continuous linear or nonlinear ODEs, the recursive block decomposition strategy developed in this thesis can be extended to handle the resulting implicitly discretized systems. On the other hand, for switched linear dynamic systems, adopting implicit discretization schemes does not introduce extra complexity because they result in residuals with the same number of min and max terms as explicit schemes. Furthermore, discretization schemes with adaptive step sizes could be alternatives too, such as the famous Runge-Kutta-Felhberg method which uses two explicit schemes of different orders. However, such schemes generate more residuals, and thus may give rise to more complicated systems. The benefit gained from constantly adjusting step sizes has to be weighed against the cost associated with extra computational complexity.

As for reachability analysis of switched linear dynamic systems, the proposed mixed-integer optimization approach has a computational complexity that grows exponentially with problem size and length of time horizon in concern. In the MIP approach the bounds at every time instance is computed independently, unlike in set-propagation methods where computations are warm-started based on the results from previous instances. Therefore, a promising research direction would be to incorporate the mixed-integer optimization approach with a set-propagation method, such that converged results from previous stages serve as initial guesses for MIPs at the current stage.

Lastly, dimensionality reduction has been developed in the literature to approximate high dimensional functions using functions of reduced dimensions [186, 187]. These functions from decomposition may be univariate, bivariate or even multivariate; and this is known as high dimensional model representation (HDMR). More recently, the domain-reduction concept has been further extended to the so-called interval superposition arithmetics [188]. In this approach, solution set enclosures to a nonlinear factorable function are computed by propagating nonconvex sets, known as interval superposition models, through the function's computational graph. However, this approach is so far limited to the use in factorable functions; and not applicable to non-factorable functions, including implicit functions and differential equations. The interval superposition arithmetic has been shown useful to guaranteed parameter estimation in small-scale systems [188]; and its application may be expanded to the studies of thermofluidic engines which normally experience high nonlinearity and complex dynamics.

Bibliography

- [1] Ilona Johnson, William T. Choate, and Amber Davidson. Waste heat recovery. technology and opportunities in u.s. industry. Technical report, BCS, Inc., Laurel, MD (United States), 3 2008.
- [2] Ziya Söğüt, Zuhul Oktay, and Hikmet Karakoç. Mathematical modeling of heat recovery from a rotary kiln. *Applied Thermal Engineering*, 30(8):817 – 825, 2010.
- [3] Huijuan Chen, D. Yogi Goswami, and Elias K Stefanakos. A review of thermodynamic cycles and working fluids for the conversion of low grade heat. *Renew Sustain Energy Rev*, 14:3059–3067, 2010.
- [4] Yasmine Ammar, Sharon Joyce, Rosemary Norman, Yaodong Wang, and Anthony P. Roskilly. Low grade thermal energy sources and uses from the process industry in the UK. *Applied Energy*, 89(1):3–20, 2012.
- [5] Christos N. Markides. Low-concentration solar-power systems based on organic Rankine cycles for distributed-scale applications: Overview and further developments. *Frontiers in Energy Research*, 3:47, 2015.
- [6] Thomas C B Smith. Power dense thermofluidic oscillators for high load applications. In *Proceedings of the 2nd International Energy Conversion Engineering Conference*, number August 16-19, pages 1–15, Providence (RI), USA, 2004.
- [7] Thomas C B Smith. Asymmetric heat transfer in vapour cycle liquid-piston engines. In *Proceedings of the 12th International Stirling Engine Conference and Technology Exhibition*, number September 7-9, pages 302–314, Durham, U K, 2005.
- [8] Thomas C B Smith. *Thermally driven oscillations in dynamic applications*. PhD thesis, University of Cambridge, UK, 2006.
- [9] Christos N. Markides and Thomas C B Smith. A dynamic model for the efficiency optimization of an oscillatory low grade heat engine. *Energy*, 36(12):6967–6980, 2011.

- [10] Roochi Solanki, Richard Mathie, Amparo Galindo, and Christos N. Markides. Modelling of a two-phase thermofluidic oscillator for low-grade heat utilisation: Accounting for irreversible thermal losses. *Applied Energy*, 106:337–354, 2013.
- [11] Salvador Acha, Arthur Mariaud, Nilay Shah, and Christos N. Markides. Optimal design and operation of distributed low-carbon energy technologies in commercial buildings. *Energy*, 142:578–591, 2018.
- [12] Richard B. Peterson, H. Wang, and T. Herron. Performance of a small-scale regenerative Rankine power cycle employing a scroll expander. *Proceedings of the Institution of Mechanical Engineers, Part A: Journal of Power and Energy*, 222(3):271–282, 2008.
- [13] Kai Wang, Daming Sun, Jie Zhang, Ya Xu, Jiang Zou, Ke Wu, Limin Qiu, and Zhiyi Huang. Operating characteristics and performance improvements of a 500 W traveling-wave thermoacoustic electric generator. *Applied Energy*, 160:853–862, 2015.
- [14] Aly I. Taleb, Michael A.G. Timmer, Mohamed Y. El-Shazly, Aleksandr Samoilov, Valeriy A. Kirillov, and Christos N. Markides. A single-reciprocating-piston two-phase thermofluidic prime-mover. *Energy*, 104:250–265, 2016.
- [15] Christoph J. W. Kirmse, Oyeniya A. Oyewunmi, Andrew J. Haslam, and Christos N. Markides. Comparison of a novel organic-fluid thermofluidic heat converter and an organic Rankine cycle heat engine. *Energies*, 9(7):479, 2016.
- [16] Christoph J. W. Kirmse, Oyeniya A. Oyewunmi, Aly I. Taleb, Andrew J. Haslam, and Christos N. Markides. A two-phase single-reciprocating-piston heat conversion engine: Non-linear dynamic modelling. *Applied Energy*, 186:359–375, 2017.
- [17] Oyeniya A. Oyewunmi, Christoph J.W. Kirmse, Andrew J. Haslam, Erich A. Müller, and Christos N. Markides. Working-fluid selection and performance investigation of a two-phase single-reciprocating-piston heat-conversion engine. *Applied Energy*, 186:376–395, 2017.
- [18] Sindhu Preetham Burugupally and Leland Weiss. Power generation via small length scale thermo-mechanical systems: Current status and challenges, a review. *Energies*, 11(9), 2018.
- [19] Peter H. Ceperley. A pistonless Stirling engine-the traveling wave heat engine. *J Acoust Soc Am*, 66:1508–1513, 1979.
- [20] S. Backhaus and G. W. Swift. A thermoacoustic-Stirling heat engine: detailed study. *J Acoust Soc Am*, 107:3148–3166, 2000.

- [21] Roochi Solanki, Amparo Galindo, and Christos N. Markides. Dynamic modelling of a two-phase thermofluidic oscillator for efficient low grade heat utilization: Effect of fluid inertia. *Applied Energy*, 89(1):156–163, 2012.
- [22] Roochi Solanki, Amparo Galindo, and Christos N. Markides. The role of heat exchange on the behaviour of an oscillatory two-phase low-grade heat engine. *Applied Thermal Engineering*, 53(2):177–187, 2013.
- [23] Karthikeyan Palanisamy, Aly I Taleb, and Christos N Markides. Optimizing the non-inertive-feedback thermofluidic engine for the conversion of low-grade heat to pumping work. *Heat Transfer Engineering*, 36(March 2017), 2015.
- [24] Christos N. Markides, Adebayo Osuolale, Roochi Solanki, and Guy Bart V Stan. Nonlinear heat transfer processes in a two-phase thermofluidic oscillator. *Applied Energy*, 104:958–977, 2013.
- [25] L. S. Pontryagin, V. G. Boltyanskii, R. V. Gamkredlidze, and E. F. Mishchenko. *The Mathematical Theory of Optimal Processes*. Interscience Publishers Inc., New Your, NY, 1962.
- [26] Richard Bellman. *Dynamic Programming*. Princeton University Press, Princeton, NJ, USA, 1 edition, 1957.
- [27] Benoît Chachuat, Adam B. Singer, and Paul I. Barton. Global methods for dynamic optimization and mixed-integer dynamic optimization. *Industrial and Engineering Chemistry Research*, 45(25):8373–8392, 2006.
- [28] Benoît C. Chachuat. *Nonlinear and Dynamic Optimization: From Theory to Practice*. Laboratoire d’Automatique, Ecole Polytechnique Federale de Lausanne, 2007.
- [29] Lorenz T. Biegler. An overview of simultaneous strategies for dynamic optimization. *Chemical Engineering and Processing: Process Intensification*, 46(11):1043–1053, 2007.
- [30] G. P. Pollard and R. W H Sargent. Off line computation of optimum controls for a plate distillation column. *Automatica*, 6(1):59–76, 1970.
- [31] T. H. Tsang, D. M. Himmelblau, and T. F. Edgar. Optimal control via collocation and nonlinear programming. *Int. J. Control*, 21(April):763–768, 1975.
- [32] J. A. Nelder and R. Mead. A Simplex Method for Function Minimization. *The Computer Journal*, 7(4):308–313, 01 1965.
- [33] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, 1983.

- [34] John H. Holland. *Genetic Algorithms and Adaptation*, pages 317–333. Springer US, Boston, MA, 1984.
- [35] Martin Pelikan, David E. Goldberg, and Erick Cantú-Paz. Boa: The bayesian optimization algorithm. In *Proceedings of the 1st Annual Conference on Genetic and Evolutionary Computation - Volume 1*, GECCO'99, page 525–532, San Francisco, CA, USA, 1999. Morgan Kaufmann Publishers Inc.
- [36] Arnold Neumaier. Complete search in continuous global optimization and constraint satisfaction. *Acta Numerica*, 13:271–369, 2004.
- [37] William R. Esposito and Christodoulos A. Floudas. Deterministic global optimization in nonlinear optimal control problems. *Journal of Global Optimization*, 17:97–126, 2000.
- [38] Ioannis Papamichail and Claire S. Adjiman. A rigorous global optimization algorithm for problems with ordinary differential equations. *Journal of Global Optimization*, 24(1):1–33, 2002.
- [39] B. Chachuat and M. A. Latifi. A new approach in deterministic global optimisation of problems with ordinary differential equations. In *Frontiers In Global Optimization*, pages 1–23. Kluwer Academic Publishers, 2003.
- [40] Joseph K. Scott, Benoît Chachuat, and Paul I. Barton. Nonlinear convex and concave relaxations for the solutions of parametric ODEs. *Optimal Control Applications and Methods*, 34(2):145–163, 2013.
- [41] Jean-Pierre Aubin. *Viability Theory*. Birkhauser Boston Inc., Cambridge, MA, USA, 1991.
- [42] Boris Houska, Filip Logist, Jan Van Impe, and Moritz Diehl. Robust optimization of nonlinear dynamic systems with application to a jacketed tubular reactor. *Journal of Process Control*, 22(6):1152–1160, 2012.
- [43] Mario E. Villanueva, Boris Houska, and Benoît Chachuat. Unified framework for the propagation of continuous-time enclosures for parametric nonlinear ODEs. *Journal of Global Optimization*, 62(3):575–613, 2015.
- [44] Lorenz T. Biegler, Arturo M. Cervantes, and Andreas Waechter. Advances in simultaneous strategies for dynamic process optimization. *Chemical Engineering Science*, 57(4):575–593, 2002.
- [45] William R. Esposito and Christodoulos A. Floudas. Global Optimization for the Parameter Estimation of Differential Algebraic Systems. *Ind. Eng. Chem. Res.*, 39, 2000.

- [46] Antonio Flores-Tlacuahuac, Sebastain Terrazas Moreno, and Lorenz T. Biegler. Global Optimization of Highly Nonlinear Dynamic Systems. *Industrial & Engineering Chemistry Research*, 47:2643–2655, 2008.
- [47] Michal Čižniar, Marián Podmajerský, Tomáš Hirmajer, Miroslav Fikar, and Abderrazak M. Latifi. Global optimization for parameter estimation of differential-algebraic systems. *Chemical Papers*, 63(3):274–283, 2009.
- [48] Anton Miro, Carlos Pozo, Gonzalo Guillen-Gosalbez, Jose a Egea, and Laureano Jiménez. Deterministic global optimization algorithm based on outer approximation for the parameter estimation of nonlinear dynamic biological systems. *BMC Bioinformatics*, 13(1):90, 2012.
- [49] Ali Mohammad Sahlodin. *Global Optimization of Dynamic Process Systems using Complete Search Methods*. PhD thesis, McMaster University, 2012.
- [50] Spencer Daniel Schaber. *Tools for dynamic model development*. PhD thesis, Massachusetts Institute of Technology, 2014.
- [51] Nikola Peric. *Complete search methods and tools for robust parameter estimation*. PhD thesis, Imperial College London, 2018.
- [52] Yukun Wang, Christos N Markides, and Benoît Chachuat. Optimization-based investigations of a two-phase thermofluidic oscillator for low-grade heat conversion. *BMC Chemical Engineering*, 1(1):20, 2019.
- [53] B. J. Huang and M. D. Chuang. System design of orifice pulse-tube refrigerator using linear flow network analysis. *Cryogenics*, 36:889–902, 1996.
- [54] C. S. Daw, M. B. Kennel, C. E. A. Finney, and F. T. Connolly. Observing and modeling nonlinear dynamics in an internal combustion engine. *Phys Rev E*, 57:2811–2819, 1998.
- [55] Lingen Chen, Fengrui Sun, and Chih Wu. Influence of heat transfer law on the performance of a Carnot engine. *Appl Therm Eng*, 17:277–282, 1997.
- [56] Karl Heinz Hoffmann, Josef Maximilian Burzler, and Sven Schubert. Endoreversible thermodynamics. *J Non-Equilib Thermodyn*, 22:311–355, 1997.
- [57] Fengrui Sun, Wenzhen Chen, Lingen Chen, and Chih Wu. Optimal perfomance of an endoreversible Carnot heat pump. *Energy Convers Mgmt*, 38:1439–1443, 1997.

- [58] Christos N. Markides, Roochi Solanki, and Amparo Galindo. Working fluid selection for a two-phase thermofluidic oscillator: Effect of thermodynamic properties. *Applied Energy*, 124:167–185, 2014.
- [59] Roochi Solanki. *Modelling of a two-phase thermofluidic oscillator for low-grade heat utilisation*. Phd thesis, Imperial College London, 2014.
- [60] Thomas A. Henzinger. The theory of hybrid automata. In *Proceedings 11th Annual IEEE Symposium on Logic in Computer Science*, pages 278–292, July 1996.
- [61] Olaf Stursberg and Bruce H. Krogh. Efficient representation and computation of reachable sets for hybrid systems. In *Hybrid Systems: Computation and Control*, pages 482–497, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg.
- [62] Paul I. Barton and Constantinos C. Pantelides. Modeling of combined discrete/continuous processes. *AIChE Journal*, 40(6):966–979, 1994.
- [63] MIn Oh and Constantinos C. Pantelides. A modelling and simulation language for combined lumped and distributed parameter systems. *Computers & Chemical Engineering*, 20(6):611 – 633, 1996.
- [64] Process Systems Enterprise. gPROMS, 1997-2020. www.psenterprise.com/gproms.
- [65] C. Pantelides, Constantinos and Paul I. Barton. Equation-oriented dynamic simulation current status and future perspectives. *Computers & Chemical Engineering*, 17:S263 – S285, 1993.
- [66] Taeshin Park and Paul I. Barton. State event location in differential-algebraic models. *ACM Trans. Model. Comput. Simul.*, 6(2):137–165, April 1996.
- [67] Dragan Nikolić, Eustathios S. Kikkinides, and Michael C. Georgiadis. Optimization of Multibed Pressure Swing Adsorption Processes. *Industrial & Engineering Chemistry Research*, 48(11):5388–5398, 2009.
- [68] George N. Nikolaidis, Eustathios S. Kikkinides, and Michael C. Georgiadis. *Modelling and Optimization of Pressure Swing Adsorption (PSA) Processes for Post-combustion CO2 Capture from Flue Gas*, chapter 13, pages 343–369. John Wiley & Sons, Ltd, 2017.
- [69] Yukun Wang, Christos N. Markides, and Benoît Chachuat. Optimization-based Investigations of a Thermofluidic Engine for Low-grade Heat Recovery. *IFAC-PapersOnLine*, 51(18):690 – 695, 2018.

- [70] Andreas Wächter and Lorenz T. Biegler. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical Programming*, 106(1):25–57, Mar 2006.
- [71] Alan C Hindmarsh, Peter N Brown, Keith E Grant, Steven L Lee, Radu Serban, Dan E Shumaker, and Carol S Woodward. SUNDIALS: Suite of nonlinear and differential/algebraic equation solvers. *ACM Transactions on Mathematical Software (TOMS)*, 31(3):363–396, 2005.
- [72] Dieter Grass, Jonathan P. Caulkins, Gustav Feichtinger, Gernot Tragler, and Doris A. Behrens. *Optimal Control of Nonlinear Processes: With Applications in Drugs, Corruption, and Terror*. Springer, Berlin, Germany, 2008.
- [73] Gerald Teschl. *Ordinary Differential Equations and Dynamical Systems*. Graduate studies in mathematics. American Mathematical Society, Providence (RI), USA, 2012.
- [74] Annick Dhooge, Willy Govaerts, and Yuri A. Kuznetsov. MatCont: A MATLAB package for numerical bifurcation analysis of ODEs. *ACM Transactions on Mathematical Software*, 29:141–164, 2003.
- [75] Willy Govaerts, Yuri A. Kuznetsov, and Annick Dhooge. Numerical continuation of bifurcations of limit cycles in MATLAB. *SIAM Journal on Scientific Computing*, 27:231–252, 2005.
- [76] Kaisa Miettinen. *Nonlinear Multiobjective Optimization*. Kluwer Academic Publishers, Dordrecht, the Netherlands, 1999.
- [77] Andrew R. Conn, K. Scheinberg, and Luis N. Vicente. *Introduction to Derivative-Free Optimization*. MOS-SIAM Series on Optimization, Philadelphia, PA, 2009.
- [78] Ilya. M. Sobol’, Danil Asotsky, Alexander Kreinin, and Sergei Kucherenko. Construction and comparison of high-dimensional Sobol’ generators. *Wilmott Magazine*, Nov:64–79, 2012.
- [79] K. Pyragas. Continuous control of chaos by self-controlling feedback. *Phys Lett A*, 170(6):421–428, 1992.
- [80] Benoît Chachuat and OMEGA Research Group. MC++ (version 2.0): Toolkit for Construction, Manipulation and Bounding of Factorable Functions, 2019. Available from <https://omega-icl.github.com/cronos>.
- [81] Chachuat, Benoît and OMEGA Research Group. CRONOS - Complete seaRch sOlutions for NOnlinear Systems, 2019. Available from <https://bitbucket.org/omegasoftware/cronos>.

- [82] Robert E. Pugh. A language for nonlinear programming problems. *Mathematical Programming*, 2(1):176–206, 1972.
- [83] Gershon Kedem. Automatic differentiation of computer programs. *ACM Trans. Math. Software*, 6(2):150–165, 1980.
- [84] Ramon E. Moore, Ralph B. Kearfott, and Michael J. Cloud. *Introduction to Interval Analysis*, volume 22. SIAM, Philadelphia, 2009.
- [85] Markus Neher. From Interval Analysis to Taylor Models - An Overview. In *International Association for Mathematics and Computers in Simulation*, pages 102–110, Paris, 2005.
- [86] Ali M. Sahlodin and Benoît Chachuat. Convex/concave relaxations of parametric ODEs using Taylor models. *Computers and Chemical Engineering*, 35(5):844–857, 2011.
- [87] Martin Berz. From Taylor Series to Taylor models. In *AIP Conference Proceedings*, number 405, pages 1–23, 1997.
- [88] Martin Berz and Georg Hoffstätter. Computation and Application of Taylor Polynomials with Interval Remainder Bounds. *Reliable Computing*, 4(1):83–97, 1998.
- [89] Kyoko Makino and Martin Berz. Efficient control of the dependency problem based on taylor model methods. *Reliable Computing*, 5(1):3–12, Feb 1999.
- [90] Kyoko Makino and Martin Berz. Taylor models and other validated functional inclusion methods. *International Journal of Pure and Applied Mathematics*, 4(4):379–456, 2003.
- [91] Jai Rajyaguru, Mario E. Villanueva, Boris Houska, and Benoît Chachuat. Higher-order inclusions of nonlinear systems by Chebyshev models. In *AIChE Annual Meeting 2014*, 2014. Paper 385510.
- [92] Tomáš Dzetkulič. Rigorous integration of non-linear ordinary differential equations in chebyshev basis. *Numerical Algorithms*, 69(1):183–205, 2015.
- [93] J. Rajyaguru, M. E. Villanueva, B. Houska, and B. Chachuat. Chebyshev model arithmetic for factorable functions. *Journal of Global Optimization*, 68(2):413–438, 2017.
- [94] Mohit Tawarmalani and Nikolaos V. Sahinidis. *Convexification and Global Optimization in Continuous and Mixed-Integer Nonlinear Programming*. Springer US, 2002.
- [95] Garth P. McCormick. Computability of global solutions to factorable nonconvex programs: Part I - Convex underestimating problems. *Mathematical Programming*, 10(1):147–175, 1976.

- [96] Alexander Mitsos, Benoît Chachuat, and Paul I. Barton. McCormick-Based Relaxations of Algorithms. *SIAM Journal on Optimization*, 20(2):573–601, 2009.
- [97] Joseph K. Scott, Matthew D. Stuber, and Paul I. Barton. Generalized McCormick relaxations. *Journal of Global Optimization*, 51(4):569–606, 2011.
- [98] Garth P. McCormick. *Nonlinear Programming: Theory, Algorithms, and Applications*. John Wiley & Sons, Inc., New York, NY, USA, 1983.
- [99] C. S. Adjiman, I. P. Androulakis, and C. A. Floudas. A global optimization method, aBB, for general twice-differentiable constrained NLPs-II. Implementation and computational results. *Computers and Chemical Engineering*, 22(9):1159–1179, 1998.
- [100] C. S. Adjiman, S. Dallwig, C. A. Floudas, and A. Neumaier. A global optimization method, aBB, for general twice-differentiable constrained NLPs-I. Theoretical advances. *Computers and Chemical Engineering*, 22(9):1137–1158, 1998.
- [101] Christodoulos A. Floudas. *Deterministic Global Optimization: Theory, Methods, and Applications*. Kluwer Academic Publishers, , Boston, Mass, USA, 2000.
- [102] Costas D. Maranas and Christodoulos A. Floudas. A deterministic global optimization approach for molecular structure determination. *The Journal of Chemical Physics*, 100(2):1247, 1994.
- [103] Olaf Knüppel. PROFIL/BIAS—A fast interval library. *Computing*, 53(3):277–287, Sep 1994.
- [104] Michael Lerch, German Tischler, Jürgen Wolff Von Gudenberg, Werner Hofschuster, and Walter Krämer. FILIB++, a Fast Interval Library Supporting Containment Computations. *ACM Trans. Math. Softw.*, 32(2):299–324, June 2006.
- [105] Edward M. B. Smith and Constantinos C. Pantelides. A symbolic reformulation/spatial branch-and-bound algorithm for the global optimisation of nonconvex MINLPs. *Computers and Chemical Engineering*, 23(4-5):457–478, 1999.
- [106] Mohit Tawarmalani and Nikolaos V. Sahinidis. Global optimization of mixed-integer nonlinear programs: A theoretical and computational study. *Mathematical Programming*, 99(3):563–591, 2004.
- [107] Mohit Tawarmalani and Nikolaos V. Sahinidis. A polyhedral branch-and-cut approach to global optimization. *Mathematical Programming*, 103(2):225–249, 2005.
- [108] Faiz A Al-Khayyal and James E Falk. Jointly Constrained Biconvex Programming. *Mathematics of Operations Research*, 8(2):273–286, 1983.

- [109] *CPLEX Optimizer*, 2019 (accessed 22 Sep, 2019). <https://www.ibm.com/analytics/cplex-optimizer>.
- [110] *Gurobi Optimizer*, 2019 (accessed 22 Sep, 2019). <https://www.gurobi.com/products/gurobi-optimizer/>.
- [111] Ramon E. Moore and Fritz Bierbaum. *Methods and Applications of Interval Analysis (SIAM Studies in Applied and Numerical Mathematics)*. Soc for Industrial & Applied Math, 1979.
- [112] Arnold Neumaier. *Interval Methods for Systems of Equations*. Cambridge University Press, 1990.
- [113] Ralph Baker Kearfott and Xiaofa Shi. Optimal Preconditioners for Interval Gauss-Seidel Methods. In *Scientific Computing and Validated Numerics*, pages 173–178. Akademie Verlag, 1996.
- [114] Jai Rajyaguru and Benoît Chachuat. Taylor models in deterministic global optimization for large-scale systems with few degrees of freedom. *Computer Aided Chemical Engineering*, 32:973–978, 2013.
- [115] Jai Rayjaguru, Mario E. Villanueva, Boris Houska, and Benoît Chachuat. Continuous-time enclosures for uncertain implicit differential equations. *IFAC-PapersOnLine*, 48(8):94 – 99, 2015. 9th IFAC Symposium on Advanced Control of Chemical Processes ADCHEM 2015.
- [116] Ralph Baker Kearfott. *Rigorous Global Search: Continuous Problems Nonconvex Optimization and Its Applications*. 1st edition, 1996.
- [117] Matthew D. Stuber and Paul I. Barton. Robust simulation and design using semi-infinite programs with implicit functions. *Int. J. Reliability and Safety*, 5(3/4):378–397, 2011.
- [118] Eldon Hansen. Interval Arithmetic in Matrix Computations , Part I. *Journal of the Society for Industrial and Applied Mathematics: Series B, Numerical Analysis*, 2(2):308–320, 1965.
- [119] Eldon Hansen and Roberta Smith. Interval Arithmetic in Matrix Computations , Part II. *SIAM Journal on Numerical Analysis*, 4(1):1–9, 1967.
- [120] Eldon Hansen and G. William Walster. *Global Optimization Using Interval Analysis: Revised and Expanded*. 2nd edition, 2003.
- [121] Wolfgang Walter. *Differential and Integral Inequalities*. Springer, Berlin, 1970.
- [122] Boris Houska, Filip Logist, Jan Van Impe, and Moritz Diehl. Robust optimization of nonlinear dynamic systems with application to a jacketed tubular reactor. *Journal of Process Control*, 22(6):1152 – 1160, 2012.

- [123] Mario Villanueva, Radoslav Paulen, Boris Houska, and Benoît Chachuat. Enclosing the Reachable Set of Parametric ODEs using Taylor Models and Ellipsoidal Calculus. In *23rd European Symposium on Computer Aided Process Engineering*, volume 32 of *Computer Aided Chemical Engineering*, pages 979 – 984. Elsevier, 2013.
- [124] Boris Houska, Mario Eduardo Villanueva, and Benoît Chachuat. A validated integration algorithm for nonlinear ODEs using Taylor models and ellipsoidal calculus. In *Proceedings of the IEEE Conference on Decision and Control*, pages 484–489, Florence, Italy, 2013.
- [125] Matthew E. Wilhelm, Anne V. Le, and Matthew D. Stuber. Global optimization of stiff dynamical systems. *AIChE Journal*, n/a(n/a):e16836, 2019.
- [126] N.S. Nedialkov, K.R. Jackson, and G.F. Corliss. Validated solutions of initial value problems for ordinary differential equations. *Applied Mathematics and Computation*, 105(1):21 – 68, 1999.
- [127] Youdong Lin and Mark A. Stadtherr. Validated solutions of initial value problems for parametric ODEs. *Applied Numerical Mathematics*, 57(10):1145–1162, 2007.
- [128] Matthias Althoff, Olaf Stursberg, and Martin Buss. Reachability analysis of nonlinear systems with uncertain parameters using conservative linearization. In *Proceedings of the IEEE Conference on Decision and Control*, pages 4042–4048. IEEE, 2008.
- [129] Matthias Althoff. Reachability analysis of nonlinear systems using conservative polynomialization and non-convex sets. In *HSCC 2013 - Proceedings of the 16th International Conference on Hybrid Systems: Computation and Control, Part of CPSWeek 2013*, pages 173–182, 2013.
- [130] Xuejiao Yang and Joseph K. Scott. Efficient reachability bounds for discrete-time nonlinear systems by extending the continuous-time theory of differential inequalities. *2018 Annual American Control Conference (ACC)*, pages 6242–6247, 2018.
- [131] Boris Houska, Hans Joachim Ferreau, and Moritz Diehl. Acado toolkit—an open-source framework for automatic control and dynamic optimization. *Optimal Control Applications and Methods*, 32(3):298–312, 2011.
- [132] Matthias Althoff. An Introduction to CORA 2015. In *Proc. of the Workshop on Applied Verification for Continuous and Hybrid Systems*, 2015.
- [133] Matthias Althoff and Dmitry Grebenyuk. Implementation of interval arithmetic in CORA 2016. In *Proc. of the 3rd International Workshop on Applied Verification for Continuous and Hybrid Systems*, pages 91–105, 2016.

- [134] Matthias Althoff, Dmitry Grebenyuk, and Niklas Kochdumper. Implementation of Taylor Models in CORA 2018. In *Proc. of the 5th International Workshop on Applied Verification for Continuous and Hybrid Systems*, pages 145–173, 2018.
- [135] Jens Hoefkens, Martin Berz, and Kyoko Makino. Computing Validated Solutions of Implicit Differential Equations. *Advances in Computational Mathematics*, 19(1):231–253, 2003.
- [136] Andreas Rauh, Michael Brill, and Clemens Günther. A Novel Interval Arithmetic Approach for Solving Differential-Algebraic Equations with ValEncIA-IVP. *Int. J. Appl. Math. Comput. Sci.*, 19(3):381–397, September 2009.
- [137] Joseph K. Scott and Paul I. Barton. Interval bounds on the solutions of semi-explicit index-one DAEs. Part 1: analysis. *Numerische Mathematik*, 125(1):1–25, 2013.
- [138] Matthew D. Stuber, Joseph K. Scott, and Paul I. Barton. Convex and concave relaxations of implicit functions. *Optimization Methods and Software*, 30(3):424–460, 2015.
- [139] John C. Butcher. *Numerical Methods for Ordinary Differential Equations*. 2 edition, 2008.
- [140] Arieh Iserles. *A First Course in the Numerical Analysis of Differential Equations*. Cambridge Texts in Applied Mathematics. Cambridge University Press, 2 edition, 2008.
- [141] Kendall E. Atkinson, Weimin Han, and David Stewart. *Numerical Solution of Ordinary Differential Equations*. 2011.
- [142] Lorenz T. Biegler, Ignacio E. Grossmann, and Arthur W. Westerberg. *Systematic methods of chemical process design*. Prentice Hall PTR, Upper Saddle River, N.J, 1997.
- [143] Robert Tarjan. Depth-First Search and Linear Graph Algorithms. *SIAM Journal on Computing*, 1(2):146–160, 1972.
- [144] HSL. A collection of Fortran codes for large scale scientific computation. <http://www.hsl.rl.ac.uk/>. Accessed: 2019-11-12.
- [145] Ali Baharev, Ferenc Domes, and Arnold Neumaier. A robust approach for finding all well-separated solutions of sparse systems of nonlinear equations. *Numerical Algorithms*, 76(1):163–189, Sep 2017.
- [146] Ali Baharev, Arnold Neumaier, and Hermann Schichl. A manifold-based approach to sparse global constraint satisfaction problems. *Journal of Global Optimization*, 75(4):949–971, Dec 2019.

- [147] Arnold Neumaier. *Interval Methods for Systems of Equations*. Cambridge University Press, 1990.
- [148] H. A. Luther. Further explicit fifth-order runge-kutta formulas. *SIAM Review*, 8(3):374–380, 1966.
- [149] John C. Butcher. On runge-kutta processes of high order. *Journal of the Australian Mathematical Society*, 4(2):179–194, 1964.
- [150] G. J. Cooper and J. H. Verner. Some Explicit Runge-Kutta Methods of High Order. *SIAM Journal on Numerical Analysis*, 9(3):389–405, 1972.
- [151] Iain S. Duff. Algorithm 575: Permutations for a zero-free diagonal [F1]. *ACM Trans. Math. Softw.*, 7(3):387–390, Sep 1981.
- [152] Iain S. Duff and John K. Reid. An Implementation of Tarjan’s Algorithm for the Block Triangularization of a Matrix. *ACM Trans. Math. Softw.*, 4(2):137–147, Jun 1978.
- [153] Claus Bendtsen and Ole Stauning. FADBAD++: Flexible Automatic differentiation using templates and operator overloading in C++, 2007. Available from <http://www.fadbad.com/fadbad.html>.
- [154] E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen. *LAPACK Users’ Guide*. Society for Industrial and Applied Mathematics, Philadelphia, PA, third edition, 1999.
- [155] Antoine Girard. Reachability of uncertain linear systems using zonotopes. In *Hybrid Systems: Computation and Control*, pages 291–305, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.
- [156] Goran Frehse. PHAVer: algorithmic verification of hybrid systems past HyTech. *International Journal on Software Tools for Technology Transfer*, 10(3):263–279, Jun 2008.
- [157] Thao Dang and Oded Maler. Reachability analysis via face lifting. In *Hybrid Systems: Computation and Control*, pages 96–109, Berlin, Heidelberg, 1998. Springer Berlin Heidelberg.
- [158] Eugene Asarin, Olivier Bournez, Thao Dang, and Oded Maler. Approximate reachability analysis of piecewise-linear dynamical systems. In *Hybrid Systems: Computation and Control*, pages 20–31, Berlin, Heidelberg, 2000. Springer Berlin Heidelberg.
- [159] Alongkrit Chutinan and Bruce H. Krogh. Computational techniques for hybrid system verification. *IEEE Transactions on Automatic Control*, 48(1):64–75, Jan 2003.

- [160] Alexander B. Kurzanski and Pravin Varaiya. Ellipsoidal techniques for reachability analysis. In *Hybrid Systems: Computation and Control*, pages 202–214, Berlin, Heidelberg, 2000. Springer Berlin Heidelberg.
- [161] Alex A. Kurzanskiy and Pravin Varaiya. Ellipsoidal techniques for reachability analysis of discrete-time linear systems. *IEEE Transactions on Automatic Control*, 52(1):26–38, Jan 2007.
- [162] Colas Le Guernic and Antoine Girard. Reachability analysis of linear systems using support functions. *Nonlinear Analysis: Hybrid Systems*, 4(2):250 – 262, 2010.
- [163] Goran Frehse and Rajarshi Ray. Flowpipe-guard intersection for reachability computations with support functions. *IFAC Proceedings Volumes*, 45(9):94 – 101, 2012.
- [164] Goran Frehse, Rajat Kateja, and Colas Le Guernic. Flowpipe approximation and clustering in space-time. In *HSCC 2013 - Proceedings of the 16th International Conference on Hybrid Systems: Computation and Control, Part of CPSWeek 2013*, volume 1, pages 203–212, 2013.
- [165] Antoine Girard, Colas Le Guernic, and Oded Maler. Efficient computation of reachable sets of linear time-invariant systems with inputs. In *Hybrid Systems: Computation and Control*, pages 257–271, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.
- [166] Antoine Girard and Colas Guernic. Zonotope/hyperplane intersection for hybrid systems reachability analysis. In *Proceedings of the 11th International Workshop on Hybrid Systems: Computation and Control*, HSCC '08, pages 215–228, Berlin, Heidelberg, 2008. Springer-Verlag.
- [167] Pavithra Prabhakar and Mahesh Viswanathan. A dynamic algorithm for approximate flow computations. In *HSCC'11 - Proceedings of the 2011 ACM/SIGBED Hybrid Systems: Computation and Control*, pages 133–142, 2011.
- [168] Kyoko Makino and Martin Berz. Rigorous Integration of Flows and ODEs Using Taylor Models. In *Proceedings of the 2009 Conference on Symbolic Numeric Computation*, SNC '09, pages 79–84, New York, NY, USA, 2009. ACM.
- [169] Parasara Sridhar Duggirala and Mahesh Viswanathan. Parsimonious, simulation based verification of linear systems. In *Computer Aided Verification*, pages 477–494, Cham, 2016. Springer International Publishing.
- [170] Stanley Bak and Parasara Sridhar Duggirala. HyLAA: A Tool for Computing Simulation-Equivalent Reachability for Linear Systems. In *Proceedings of the 20th International Conference on Hybrid Systems: Computation and Control*, HSCC '17, pages 173–178, New York, NY, USA, 2017. ACM.

- [171] Xin Chen and Sriram Sankaranarayanan. Decomposed reachability analysis for nonlinear systems. In *2016 IEEE Real-Time Systems Symposium (RTSS)*, pages 13–24, Nov 2016.
- [172] Eugene Asarin, Thao Dang, and Antoine Girard. Hybridization methods for the analysis of nonlinear systems. *Acta Informatica*, 43(7):451–476, Feb 2007.
- [173] Thao Dang, Oded Maler, and Romain Testylier. Accurate hybridization of nonlinear systems. In *Proceedings of the 13th ACM International Conference on Hybrid Systems: Computation and Control*, HSCC '10, pages 11–20, New York, NY, USA, 2010. ACM.
- [174] Goran Frehse. SpaceEx state space explorer, 2010. Verimag, Grenoble, <http://spaceex.imag.fr>.
- [175] Goran Frehse, Colas Le Guernic, Alexandre Donzé, Scott Cotton, Rajarshi Ray, Olivier Lebeltel, Rodolfo Ripado, Antoine Girard, Thao Dang, and Oded Maler. SpaceEx: Scalable verification of hybrid systems. In *Proc. 23rd International Conference on Computer Aided Verification (CAV)*, LNCS. Springer, 2011.
- [176] Stefan Schupp, Erika Abraham, Ibtissem Ben Makhlof, and Stefan Kowalewski. HyPro: A C++ Library for State Set Representations for Hybrid Systems Reachability Analysis. In *Proc. of the 9th NASA Formal Methods Symposium (NFM'17)*, volume 10227 of LNCS, pages 288–294. Springer International Publishing, April 2017.
- [177] Xin Chen, Erika Ábrahám, and Sriram Sankaranarayanan. Flow*: An analyzer for non-linear hybrid systems. In *Computer Aided Verification*, pages 258–263, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.
- [178] Igor Griva, Stephen Gregory Nash, and Ariela Sofer. *Linear and Nonlinear Optimization: Second Edition*. Other Titles in Applied Mathematics. Society for Industrial and Applied Mathematics, 2009.
- [179] Gerasimos G. Rigatos. Adaptive fuzzy control of dc motors using state and output feedback. *Electric Power Systems Research*, 79(11):1579 – 1592, 2009.
- [180] Thermofluidics Ltd. <http://www.thermofluidics.co.uk>. Accessed: 2019-04-08.
- [181] Michael L. Thompson and Mark A. Kramer. Modeling chemical processes using prior knowledge and neural networks. *AIChE Journal*, 40(8):1328–1340, 1994.

- [182] Zhiguang Qian, Carolyn Conner Seepersad, V. Roshan Joseph, Janet K. Allen, and C. F. Jeff Wu. Building Surrogate Models Based on Detailed and Approximate Simulations. *Journal of Mechanical Design*, 128(4):668–677, 06 2005.
- [183] Atharv Bhosekar and Marianthi Ierapetritou. Advances in surrogate based modeling, feasibility analysis, and optimization: A review. *Computers and Chemical Engineering*, 108:250–267, 2018.
- [184] Donald R. Jones, Matthias Schonlau, and William J. Welch. Efficient global optimization of expensive black-box functions. *Journal of Global Optimization*, 13(4):455–492, Dec 1998.
- [185] Donald R. Jones. A taxonomy of global optimization methods based on response surfaces. *Journal of Global Optimization*, 21(4):345–383, Dec 2001.
- [186] Ilya M. Sobol'. Theorems and examples on high dimensional model representation. *Reliability Engineering & System Safety*, 79(2):187 – 193, 2003. SAMO 2001: Methodological advances and innovative applications of sensitivity analysis.
- [187] H. Xu and S. Rahman. A generalized dimension-reduction method for multidimensional integration in stochastic mechanics. *International Journal for Numerical Methods in Engineering*, 61(12):1992–2019, 2004.
- [188] Junyan Su, Yanlin Zha, Kai Wang, Mario E. Villanueva, Radoslav Paulen, and Boris Houska. Interval superposition arithmetic for guaranteed parameter estimation. *12th IFAC Symposium on Dynamics and Control of Process Systems, including Biosystems DYCOPS 2019*, 52(1):574 – 579, 2019.

Appendix A

gPROMS ModelBuilder script for the NIFTE PWLTP-TL model

```
1 PARAMETER
2   P0          AS REAL DEFAULT 1.013E5
3   U0          AS REAL DEFAULT 8E-4
4   tau         AS REAL DEFAULT 5
5   Lambda      AS REAL DEFAULT 330
6   K           AS REAL DEFAULT 1.37
7
8   # RLC parameters
9   R_l         AS REAL DEFAULT 4.0809E6
10  R_f         AS REAL DEFAULT 2.1307E6
11  R_th        AS REAL DEFAULT 5.0212E8
12  L_d         AS REAL DEFAULT 1.8027E5
13  L_l         AS REAL DEFAULT 1.2710E7
14  L_p         AS REAL DEFAULT 3.7739E5
15  L_f         AS REAL DEFAULT 4.7428E6
16  C_ad        AS REAL DEFAULT 1.7618E-9
17  C_d         AS REAL DEFAULT 7.3513E-8
18  C_p         AS REAL DEFAULT 7.4280E-8
19
20  R_TL1       AS REAL DEFAULT 8.8203e+06
21  C_TL1       AS REAL DEFAULT 8.6087e-10
22  R_TL2       AS REAL DEFAULT 1.7178e+07
```

```

23  C_TL2      AS REAL DEFAULT 5.9528e-09
24  R_TL3      AS REAL DEFAULT 3.8277e+07
25  C_TL3      AS REAL DEFAULT 1.7939e-08
26  R_TL4      AS REAL DEFAULT 2.8117e+08
27  C_TL4      AS REAL DEFAULT 2.6578e-08
28  R_TL5      AS REAL DEFAULT 3.1500e+07
29
30 VARIABLE
31   # key thermodynamic states
32   P_ad       AS NonDimensionisedValues
33   P_d        AS NonDimensionisedValues
34   P_p        AS NonDimensionisedValues
35   U_f        AS NonDimensionisedValues
36   U_p        AS NonDimensionisedValues
37   P_TL1      AS NonDimensionisedValues
38   P_TL2      AS NonDimensionisedValues
39   P_TL3      AS NonDimensionisedValues
40   P_TL4      AS NonDimensionisedValues
41
42   # initial values
43   P_ad_0     AS NonDimensionisedValues
44   P_d_0      AS NonDimensionisedValues
45   P_p_0      AS NonDimensionisedValues
46   U_f_0      AS NonDimensionisedValues
47   U_p_0      AS NonDimensionisedValues
48   P_TL1_0    AS NonDimensionisedValues
49   P_TL2_0    AS NonDimensionisedValues
50   P_TL3_0    AS NonDimensionisedValues
51   P_TL4_0    AS NonDimensionisedValues
52
53 SELECTOR
54   Regime AS ( Lower , Middle , Upper ) DEFAULT Middle
55
56 EQUATION
57   $P_d = (tau*U0*U_f) / (P0*C_d);
58   $P_p = (tau*U0*U_p) / (P0*C_p);
59   $U_f = ((tau*P0/U0) * (L_l*P_p-L_p*P_ad-(L_p+L_l)*P_d)
60         - tau*(L_l*R_f+L_p*(R_f+R_l))*U_f-tau*L_p*R_l*U_p)
61         / (L_l*(L_d+L_f)+L_p*(L_d+L_f+L_l));

```

```

62  $U_p = ((tau*P0/U0)*(L_l*P_d-(L_d+L_f)*P_ad-(L_d+L_f+L_l)*P_p)
63        + tau*(L_l*R_f-(L_d+L_f)*R_l)*U_f-tau*(L_d+L_f)*R_l*U_p)
64        / (L_l*(L_d+L_f)+L_p*(L_d+L_f+L_l));
65
66  $P_TL1 = (tau*(P_ad-P_TL1-P_TL2-P_TL3-P_TL4))/(R_TL5*C_TL1)
67           - (tau*P_TL1)/(R_TL1*C_TL1);
68  $P_TL2 = (tau*(P_ad-P_TL1-P_TL2-P_TL3-P_TL4))/(R_TL5*C_TL2)
69           - (tau*P_TL2)/(R_TL2*C_TL2);
70  $P_TL3 = (tau*(P_ad-P_TL1-P_TL2-P_TL3-P_TL4))/(R_TL5*C_TL3)
71           - (tau*P_TL3)/(R_TL3*C_TL3);
72  $P_TL4 = (tau*(P_ad-P_TL1-P_TL2-P_TL3-P_TL4))/(R_TL5*C_TL4)
73           - (tau*P_TL4)/(R_TL4*C_TL4);
74
75  CASE Regime OF
76    WHEN Lower:
77      $P_ad = tau*((-K- P_ad)/(R_th*C_ad)
78                + U0*(U_f+U_p)/(P0*C_ad)
79                - (P_ad-P_TL1-P_TL2-P_TL3-P_TL4)/(R_TL5*C_ad));
80      SWITCH TO Middle IF (Lambda*P_d) > -1;
81    WHEN Middle:
82      $P_ad = tau((((K*Lambda*P_d)-P_ad)/(R_th*C_ad)
83                + U0*(U_f+U_p)/(P0*C_ad)
84                - (P_ad-P_TL1-P_TL2-P_TL3-P_TL4)/(R_TL5*C_ad));
85      SWITCH TO Upper IF (Lambda*P_d)>1;
86      SWITCH TO Lower IF (Lambda*P_d)<-1;
87    WHEN Upper:
88      $P_ad = tau*((K-P_ad)/(R_th*C_ad)
89                + U0*(U_f+U_p)/(P0*C_ad)
90                - (P_ad-P_TL1-P_TL2-P_TL3-P_TL4)/(R_TL5*C_ad));
91      SWITCH TO Middle IF (Lambda*P_d)<1;
92  END

```

Listing A.1: Script for the Models entity PWLTP_TL in gPROMS ModelBuilder for the NIFTE PWLTP-TL model

```

1  PWLTP_TL_proc AS    PWLTP_TL
2
3  ASSIGN
4    WITHIN NTPTL_proc_sim DO
5      P_ad_0 := 0.01;

```

```

6      P_d_0      := 0;
7      P_p_0      := 0;
8      U_f_0      := 0.01;
9      U_p_0      := 0;
10     P_TL1_0    := 0;
11     P_TL2_0    := 0;
12     P_TL3_0    := 0;
13     P_TL4_0    := 0;
14     END
15
16 INITIAL
17     WITHIN NTPTL_proc_sim DO
18         P_ad = P_ad_0;
19         P_d  = P_d_0;
20         P_p  = P_p_0;
21         U_f  = U_f_0;
22         U_p  = U_p_0;
23         P_TL1 = P_TL1_0;
24         P_TL2 = P_TL2_0;
25         P_TL3 = P_TL3_0;
26         P_TL4 = P_TL4_0;
27     END
28
29 INITIALISATION_PROCEDURE
30     # Start Default Initialisation Procedure
31     USE
32         : DEFAULT;
33     END
34     SAVE "Cycle_InitialGuess";
35     # End Default Initialisation Procedure
36
37 SOLUTIONPARAMETERS
38     gRMS :=      ON;
39
40 SCHEDULE
41     CONTINUE FOR 20.

```

Listing A.2: Script for the Processes entity PWLTP_TL_proc in gPROMS ModelBuilder for the NIFTE PWLTP-TL model

Appendix B

Example code snippet for the full discretization set-propagation approach implemented in CRONOS

```
1  #include "interval.hpp"
2  #include "cmodel.hpp"
3  #include "odebnd_expand.hpp"
4  typedef mc::Interval I;
5  typedef mc::CModel<I> PM;
6  typedef mc::CVar<I> PV;
7
8  //Constant Parameters
9  const double b1dR = 2.5e3;
10 const double b2dR = 5.0e3;
11 const double a1 = 4.0e3;
12 const double a2 = 6.2e5;
13 const double TL = 298.;
14 const double TU = 398.;
15 const double tf = 1.;
16
17 int main()
18 {
19     mc::FFGraph DAG;
```

```

20  const unsigned NS = 1;
21  double TS[NS+1]; TS[0] = 0.;
22  for( unsigned k=0; k<NS; k++ ) TS[k+1] = TS[k] + tf/(double)NS;
23  mc::FFVar T; T.set( &DAG );
24  const unsigned NP = 1; // Parameters
25  mc::FFVar P[NP];
26  for( unsigned int i=0; i<NP; i++ ) P[i].set( &DAG );
27  const unsigned NX = 2; // States
28  mc::FFVar X[NX];
29  for( unsigned int i=0; i<NX; i++ ) X[i].set( &DAG );
30  mc::FFVar RHS[NX*NS]; // Dynamics
31  for( unsigned k=0; k<NS; k++ ){
32  mc::FFVar K1 = a1 * exp( -b1dR / P[k] );
33  mc::FFVar K2 = a2 * exp( -b2dR / P[k] );
34  RHS[NX*k+0] = - K1 * sqr( X[0] );
35  RHS[NX*k+1] = K1 * sqr( X[0] ) - K2 * X[1];
36  }
37
38  mc::FFVar IC[NX] = { 1., 0. }; // Initial conditions
39  I Ip[NP]; // Parameter bounds
40  for( unsigned k=0; k<NP; k++ ) Ip[k] = I( TL, TU );
41  PM PMEnv( NP, 4 ); // Setup polynomial model environment
42  PV PMp[NP];
43  for( unsigned i=0; i<NP; i++ ) PMp[i].set( &PMEnv, i, Ip[i] );
44
45  // Problem setup
46  mc::ODEBND_EXPAND<I,PM,PV> IVP;
47  IVP.set_dag( &DAG );
48  IVP.set_time( NS, TS, &T );
49  IVP.set_parameter( NP, P );
50  IVP.set_state( NX, X );
51  IVP.set_differential( 1, NX, RHS );
52  IVP.set_initial( NX, IC );
53  IVP.options.INTMETH = mc::ODEBND_EXPAND<I,PM,PV>::Options::
    METHOD::RK; // Discretization method
54  IVP.options.TORD = 4; // Order of discretization
55  IVP.options.HO = 0.02; // Step size
56  IVP.options.LBLK = IVP.options.DBLK = 50;

```

```

57  IVP.options.AEBND.MAXIT    = 100;
58  IVP.options.AEBND.RTOL    = IVP.options.AEBND.ATOL    = 1e-10;
59  IVP.options.AEBND.BOUNDER = mc::AEBND<I,PM,PV>::Options::
    ALGORITHM::GS;;          // Bounding method
60  IVP.options.AEBND.PRECOND = mc::AEBND<I,PM,PV>::Options::
    PRECONDITIONING::INVMB;   // Preconditioner
61  IVP.options.AEBND.BLKDEC  = mc::AEBND<I,PM,PV>::Options::
    DECOMPOSITION::RECUR;    // Block decomposition strategy
62  IVP.setup();
63
64  PV PMx0[NX] = { I(0.,1.), I(0.,1.) } // A priori state bounds
65  PV *PMxk[2];
66  for( unsigned k=0; k<=NS; k++ ){
67  PMxk[k] = new PV[NX];
68  for( unsigned i=0; i<NX; i++ )
69  PMxk[k][i] = PMx0[i];
70  }
71  IVP.bounds( PMp, PMxk ); // Compute polynomial model enclosure
72
73  return 0;
74  }
75

```

Listing B.1: Implementation for computing polynomial model-based enclosures for (5.4.1) using the header file `odebend_expand.hpp` in CRONOS

Appendix C

Example code snippet for the mixed-integer optimization approach implemented in CRONOS

```
1 #include "odebnd_expand.hpp"
2 #include "nlgo.hpp"
3 #include "interval.hpp"
4 typedef mc::Interval I;
5 #include "cmodel.hpp"
6 typedef mc::CModel<I> PM;
7 typedef mc::CVar<I> PV;
8
9 int main()
10 {
11     mc::FFGraph DCMotor;
12     const double K = 8.5e-2;
13     const double R = 5.5e-1;
14     const double L = 2.5e-2;
15     const double J = 9.0e-2;
16     const double b = 5.0e-2;
17     const double theta_ref = 1.;
18     const double KP = 10;
19     const double KI = 20;
20     const double vmax = 6.;
```

```

21  const double vmin = -6.;
22
23  const unsigned int NP = 1;  // Parameter dimension
24  mc::FFVar P[NP];
25  for( unsigned int i=0; i<NP; i++ ) P[i].set( &DCMotor );
26  mc::FFVar &w = P[0];  // disturbance in current
27  const unsigned int NX = 3;  // State dimension
28  mc::FFVar X[NX];
29  for( unsigned int i=0; i<NX; i++ ) X[i].set( &DCMotor );
30  mc::FFVar &i = X[0], &theta = X[1], &q = X[2];
31
32  mc::FFVar v, RHS[NX];  // Right-hand side function
33  v = max( vmin, min( vmax, KP*(theta_ref - theta) + KI*q ) ) ;
34  RHS[0] = (1/L) * ( v - R*i - K*theta );  // rate of change of i
35  RHS[1] = (1/J) * ( K*i - b*theta + w);  // rate of change of theta
36  RHS[2] = theta_ref - theta;  // rate of change of q
37
38  mc::FFVar IC[NX];
39  for(unsigned i=0; i<NX; i++)
40      IC[i] = 0.;  // Initial conditions
41
42  I Ip[NP];
43  Ip[0] = I(-0.1,0.1);  // Parameter bounds
44  I Ix[NX];  // State a priori bounds
45  Ix[0] = I( -20., 20. );
46  Ix[1] = I(-5., 5.);
47  Ix[2] = I(-5., 5.);
48
49  // Step 1: ODE Discretization
50  mc::ODEBND_EXPAND<I,PM,PV> ODEDISCR;
51  ODEDISCR.set_dag( &DCMotor );
52  ODEDISCR.set_time( 0., 1. );
53  ODEDISCR.set_parameter( NP, P );
54  ODEDISCR.set_state( NX, X );
55  ODEDISCR.set_differential( NX, RHS );
56  ODEDISCR.set_initial( NX, IC );
57  ODEDISCR.options.INTMETH = mc::ODEBND_EXPAND<I,PM,PV>::Options::
    METHOD::RK;
58  ODEDISCR.options.TORD = 4;

```

```

59  ODEDISCR.options.H0          = 0.005;
60  ODEDISCR.options.LBLK       = 200;
61  ODEDISCR.setup();
62
63  ODEDISCR.vVAR()[NP+NX+1].set( ODEDISCR.options.H0 );
64  std::vector<I> IALL( Ip, Ip+NP );
65  for( unsigned k=0; k<=ODEDISCR.options.LBLK; k++ )
66      IALL.insert( IALL.end(), Ix, Ix+NX );
67
68  int status( 0 );
69
70  // Step 2: MIP formulation
71  for( unsigned NT=0; NT<=ODEDISCR.options.LBLK; NT++ ){
72      mc::NLGO<I> NLP;
73      NLP.options.RELMETH          = mc::NLGO<I>::Options::DRL;
74      NLP.options.MAXITER          = 50;
75      NLP.options.DEPSUSE          = false;
76      NLP.options.MIPDISPLAY       = 1;
77      NLP.options.POLIMG.BREAKPOINT_DISC = true;
78      NLP.set_dag( &DCMotor );
79      NLP.set_var( NP, ODEDISCR.vVAR().data() );
80      NLP.set_dep( NX*(NT+1), ODEDISCR.vDEP().data(), ODEDISCR.vRES().
data() );
81
82      // Minimize the value of theta at final time
83      NLP.set_obj( mc::BASE_NLP::MIN, ODEDISCR.vDEP()[NX*NT+1] );
84      NLP.setup( IALL.data() );
85      status = NLP.relax( IALL.data() );
86      const double theta_L = NLP.get_objective();
87      std::cout << "relaxed objective: " << theta_L << std::endl;
88
89      // Maximize the value of theta at final time
90      NLP.set_obj( mc::BASE_NLP::MAX, ODEDISCR.vDEP()[NX*NT+1] );
91      NLP.setup( IALL.data() );
92      status = NLP.relax( IALL.data() );
93      const double theta_U = NLP.get_objective();
94      std::cout << "relaxed objective: " << theta_U << std::endl;
95  }
96

```

```
97     return status;  
98 }
```

Listing C.1: Script for reachability analysis of the PI-controlled DC motor using the mixed-integer optimization approach implemented in CRONOS

Appendix D

Copyright permissions

Permissions have been requested to produce figures protected by copyright. License numbers are presented in Table D.1.

Table D.1: License numbers for figures that are reproduced and protected by copyright

Figure	License number
2.1	4730850714656
2.2a	4740860282528
2.2d	4730850714656
4.1	4738940146934