# HIERARCHICAL POWER MANAGEMENT IN VEHICLE SYSTEMS

BY

JUSTIN PETER KOELN

DISSERTATION

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Mechanical Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2016

Urbana, Illinois

# Abstract

This dissertation presents a hierarchical model predictive control (MPC) framework for energy management onboard vehicle systems. High performance vehicle systems such as commercial and military aircraft, on- and off-road vehicles, and ships present a unique control challenge, where maximizing performance requires optimizing the generation, storage, distribution, and utilization of energy throughout the entire system and over the duration of operation. The proposed hierarchical approach decomposes control of the vehicle among multiple controllers operating at each level of the hierarchy. Each controller has a model of a corresponding portion of the system for predicting future behavior based on current and future control decisions and known disturbances. To capture the energy storage and power flow throughout the vehicle, a graph-based modeling framework is proposed, where vertices represent capacitive elements that store energy and edges represent paths for power flow between these capacitive elements. For systems with a general nonlinear form of power flow, closed-loop stability is established through local subsystem analysis based on passivity. The ability to assess system-wide stability from local subsystem analysis follows from the particular structure of the interconnections between each subsystem, their corresponding controller, and neighboring subsystems. For systems with a linear form of power flow, robust feasibility of state and actuator constraints is achieved using a constraint tightening approach when formulating each MPC controller. Finally, the hierarchical control framework is applied to an example thermal fluid system that represents the fuel thermal management system of an aircraft. Simulation and experimental results clearly demonstrate the benefits of the proposed hierarchical control approach and the practical applicability to real physical systems with nonlinear dynamics, unknown disturbances, and actuator delays.

*To my family, friends, and teachers.*

# Acknowledgements

First, I would like to thank my advisor, Dr. Andrew Alleyne, for his invaluable and continuing commitment to help me reach my greatest potential. His mentorship over the last several years has allowed me to make tremendous strides as a researcher. As a shining example of what it means to be a leader, Dr. Alleyne has taught me priceless lessons that I will take with me throughout my career. I am very honored to have his continued mentorship throughout my career. I would also like to acknowledge the members of my doctoral examination committee, Geir Dullerud, Philip Krein, Dusan Stipanovic, and Manfred Morari, who have helped guide and strengthen my research with their own expertise.

I must also thank my family. My parents have always set an outstanding example and truly are my biggest role models. They instilled in me the value of education and the curiosity and determination required to become a successful researcher. They have always been my biggest supporters and I would not be where I am without them.

These acknowledgements would not be complete without thanking my fellow Alleyne Research Group (ARG) members. I consider myself extremely lucky to have had the opportunity to work alongside 32 current, past, and visiting members of the ARG. Thank you to Megan, Joey, and Sarah for such a great and memorable start to my grad school experience with daily lunch breaks on Green street discussing the “interesting” TED talk I watched the night before and our Friday evening pool sessions. Thank you to Kira, Dave, Neera, Tim, Vikas, Bin, and Yangmin for making me feel so welcome when I first joined the lab and for creating such a great lab dynamic with a social yet hardworking atmosphere. The example you all set is something I will remember most about my time in ARG. Thank you to Matt, Herschel, Bryan, Nanjun, Erick, Katie, and Lindsey for helping me carry on the wonderful ARG culture and traditions. Your friendships and support kept me motivated and excited to come to work each day. It is rare to find such an amazing group of people who, even at the end of a long day of working together, you want to go and grab dinner or a drink with. Thank you to Ashley, Spencer, Xavier, Malia, Pamela, Oyuna, Nate, Sarah, Spencer, and Sunny, the future of ARG, for reminding me how exciting grad school can be, constantly leaving me pleasantly surprised, and reassuring me that the ARG culture and traditions that I enjoyed so much will carry on. Finally, thank you to Kasper, Ugo, Ehsan, Kaveh, and Jose, for bringing fresh perspectives and energy to the lab. I want to thank each one of you for making my time at UIUC so enjoyable and hope we can continue to keep in touch as we go in our separate directions. I especially want to thank Matt Williams and Herschel Pangborn for their years of truly enjoyable daily collaboration and support.

Finally, I would like to acknowledge the support of the National Science Foundation (NSF) Graduate Research Fellowship Program (GRFP), the Air Force Research Laboratory (AFRL), and the NSF Engineering Research Center for Power Optimization of Electro Thermal Systems (POETS) with cooperative agreement EEC-1449548. Their support has given me the flexibility to conduct research which both interests me and is of value to society.

# Table of Contents

CHAPTER Page

LIST OF FIGURES ..... xi

CHAPTER 1 INTRODUCTION ..... 1

1.1 Motivation and Background ..... 1

1.2 Research Objectives ..... 3

1.2.1 Problem Statement ..... 3

1.2.2 Dissertation Scope ..... 4

1.3 Organization of Dissertation ..... 5

1.4 Notation ..... 6

CHAPTER 2 GRAPH-BASED POWER FLOW SYSTEMS ..... 7

2.1 Power Flow Systems ..... 7

2.2 Modeling Objectives ..... 8

2.3 Graph-based Modeling ..... 9

2.4 Generic System Formulation ..... 10

2.5 Modeling Features ..... 13

2.5.1 Modularity ..... 13

2.5.2 Energy Domain Agnostic ..... 15

2.5.3 Timescale Agnostic ..... 16

2.5.4 Variable Fidelity ..... 17

2.6 Chapter Summary ..... 18

CHAPTER 3 HIERARCHICAL MODEL PREDICTIVE CONTROL ..... 19

3.1 Conceptual Framework........19
3.2 Hierarchical Control Advantages........21
3.3 Hierarchical Control Development Procedure........24
3.4 Graph-based System Model........25
3.5 System Decomposition........26
3.6 Graph-based Model Reduction........29
3.7 Controller Development........32
3.8 Numerical Example........35
3.9 Chapter Summary........40

CHAPTER 4 PASSIVITY-BASED STABILITY........42

4.1 Motivation........42
4.2 Background........42
4.3 Nonlinear Graph System Model........43
4.4 Main Results........48
4.4.1 Passivity of Subsystems........48
4.4.2 Passivity of the System........52
4.4.3 Decentralized Closed-loop Stability........54
4.4.4 Hierarchical Closed-loop Stability........57
4.5 Numerical Example........58
4.6 Chapter Summary........63

CHAPTER 5 ROBUST FEASIBILITY........65

5.1 Motivation........65
5.2 Background........65
5.3 Linear Graph System Model........67
5.3.1 System Dynamics........67
5.3.2 Dynamic Timescales........70
5.3.3 Local Constraints........70
5.3.4 Nominal System........70
5.3.5 Control Objective........71
5.3.6 Feedback Integralization........71

5.4 Hierarchical Control Structure ........ 72
5.4.1 Subsystem Interconnections ........ 72
5.4.2 Control Structure ........ 73
5.4.3 Nominal Subsystems for Level *N* Controllers ........ 74
5.4.4 Nominal Reduced Subsystems for Level *i* Controllers, $i \in [1, N-1]$ ........ 74
5.5 Level *N* Controller Formulation ........ 77
5.5.1 Error Subsystem ........ 77
5.5.2 Constraint Tightening ........ 79
5.5.3 Level *N* MPC Problem ........ 80
5.6 Level *i* Controller Formulation ($i \neq N$) ........ 81
5.6.1 Constraint Tightening ........ 81
5.6.2 Level *i* MPC Problem ($i \neq N$, $i \neq 1$) ........ 82
5.6.3 Level 1 MPC Problem ........ 83
5.7 Recursive Feasibility ........ 85
5.8 Numerical Example ........ 87
5.9 Chapter Summary ........ 94
5.10 Chapter Appendix ........ 94
CHAPTER 6 GRAPH-BASED MODELING OF A THERMAL FLUID SYSTEM ........ 96
6.1 Motivation ........ 96
6.2 Background ........ 96
6.3 Graph-based Modeling ........ 97
6.3.1 Generic Graph Formulation ........ 97
6.3.2 Hydraulic Graph Modeling ........ 98
6.3.3 Thermal Graph Modeling ........ 99
6.3.4 Multi-graph System Representation ........ 100
6.4 Conservation-based Modeling ........ 101
6.4.1 Mass Conservation ........ 102
6.4.2 Thermal Energy Conservation ........ 106
6.5 Linearization and Discretization ........ 108
6.5.1 Hydraulic Graph Linearization ........ 108

6.5.2 Hydraulic Graph Discretization ........ 109
6.5.3 Thermal Graph Linearization ........ 110
6.5.4 Thermal Graph Discretization ........ 111
6.5.5 Actuator Dynamics ........ 112

6.6 Experimental System Description ........ 112
6.6.1 Overall System ........ 113
6.6.2 Individual Components ........ 113

6.7 Graph-based System Representation ........ 115
6.7.1 Mass Conservation System ........ 117
6.7.2 Thermal Energy Conservation System ........ 117

6.8 Model Validation ........ 118
6.8.1 Hydrodynamic Validation ........ 118
6.8.2 Thermal Validation ........ 120

6.9 Chapter Summary ........ 121

CHAPTER 7 HIERARCHICAL CONTROL OF A THERMAL FLUID SYSTEM ........ 123

7.1 Motivation ........ 123
7.2 Hierarchical Control Framework ........ 124
7.2.1 Thermal Control Layer ........ 124
7.2.1 Hydraulic Control Layer ........ 126
7.2.1 Actuator Control Layer ........ 128

7.3 Simulation Results ........ 129
7.4 Experimental Results ........ 133
7.5 Chapter Conclusions ........ 136

CHAPTER 8 CONCLUSION ........ 137

8.1 Summary of Research Contributions ........ 137
8.2 Future Work ........ 138
8.2.1 Techniques ........ 139
8.2.2 Theory ........ 140
8.2.3 Application ........ 142

REFERENCES....................................................................................................143

APPENDIX ROBUST HIERARCHICAL CONTROLLER CODE ...............................149

# List of Figures

Figure 1.1 Historical growth of onboard power for aircraft suggests effective power management will continue to be critical to the overall increase in capabilities of both military and commercial aircraft [6]. ........ 2

Fig 1.2 Vehicle systems are a complex combination of interacting systems and subsystems over multiple timescales. ........ 3

Figure 1.3 Outline of developments required for the realization of hierarchical control of power flow in vehicle systems. ........ 5

Figure 2.1 Notional system exemplifying the graph-based power flow representation with key power flows and states highlighted in red. Dashed lines indicate elements that serve as disturbances to the system. ........ 11

Figure 2.2 Simple example system used to demonstrate the modularity of a graph-based modeling framework. ........ 13

Figure 2.3 Component graphs for each component in the example system. ........ 14

Figure 2.4 Example system graph. ........ 15

Figure 2.5 Two graph models for a cold plate heat exchanger, where the fluid temperature is either represents as a single lumped temperature $T$ or three distinct temperatures $T_a, T_b, T_c$ along the length of the heat exchanger. ........ 17

Figure 3.1 Notional 5-level hierarchy with notional controller update rates for an electrical and thermal system. ........ 20

Figure 3.2 The effects of $\Delta T$ on a centralized controller with regard to large known disturbances and small high-frequency unknown disturbances compared to a hierarchical control approach. ........ 22

Figure 3.3 A system with two interconnected subsystems used to demonstrate the ability of a hierarchical control framework to directly account for the coupling between systems and subsystems. ....24
Figure 3.4 Example system graph used to demonstrate the hierarchical control development and performance. ....27
Figure 3.5 Example three-level hierarchy, and corresponding information flow, with a single vehicle-level controller, two system-level controllers, and four subsystem-level controllers. ....27
Figure 3.6 Individual subsystem graph representations for the example system. ....29
Figure 3.7 System graph representations for the example system. ....30
Figure 3.8 Vehicle graph representation for the example system. ....31
Figure 3.9 Convergence results for centralized, decentralized, and hierarchical controllers. ....36
Figure 3.10 Disturbance profiles. ....37
Figure 3.11 Disturbance rejection results for centralized, decentralized, and hierarchical controllers. ....37
Figure 3.12 Comparison of reference tracking error for centralized, decentralized, and hierarchical controllers. ....39
Figure 3.13 Comparison of controller computation times for the centralized controller with $\Delta t = 10$ and the hierarchical controllers. ....40
Figure 4.1 Notional subsystem exemplifying the graph-based power flow representation with key power flows and states highlighted in red. Dashed lines indicate elements that serve as disturbances to the subsystem. ....45
Figure 4.2 Notional interconnection between three subsystems demonstrating the key interactions and relevant variables. ....49
Figure 4.3 Block diagram for the subsystems from Fig. 4.2. ....50
Figure 4.4 Block diagram showing the negative feedback connection between the subsystem $\mathbf{S}_i$ and the controller $\mathbf{C}_i$. For clarity of presentation, controllers $\mathbf{C}_{i-1}$ and $\mathbf{C}_{i+1}$ for subsystems $\mathbf{S}_{i-1}$ and $\mathbf{S}_{i+1}$ are omitted from the diagram. ....54

Figure 4.5 Example hierarchical control structure used to improve control performance via coordination among subsystems. Only controllers $\mathbf{C}_{3,1-4}$ require passivity constraints, forming a stability-assurance layer. ............58
Figure 4.6 Graph for example fluid tank system with four subsystems. ............59
Figure 4.7 Decomposition of the example system graph into four subsystem graphs used to develop the four decentralized MPC controllers. ............60
Figure 4.8 Simple control hierarchy for the example system where $\mathbf{C}_{1-4}$ are passivity-constrained decentralized MPC controllers and $\mathbf{C}_0$ is a centralized reference generator (signal coloring is the same as Fig. 4.5). ............62
Figure 4.9 Representative state trajectories for $x_2$ and $x_{10}$ with stable and unstable reference generator formulations and for the nominal and passivity-constrained MPC designs. ............63
Figure 4.10 Trajectories for $z_1$ for subsystem $\mathbf{S}_1$ with the stable and unstable reference generator formulations and for the nominal and passivity-constrained MPC designs. Note that these trajectories are plotted separately due to the disparity in the magnitudes and sign of the trajectories for the nominal and passivity-constrained scenarios. ............64
Figure 5.1 Notional system exemplifying the graph-based power flow representation with key power flows and states highlighted in red. Dashed lines indicate elements that serve as disturbances to the system. ............69
Figure 5.2 (a) The upper-level controller plans a feasible state trajectory at the slow time step $k_1$ but the lower-level controller is unable to track this trajectory at the faster time step $k_2$ without violating state constraints. (b) Using the feedback integralization control law from (5.15), the system follows piecewise linear state trajectories; thus any trajectory that is feasible at the slower time step $k_1$ is also feasible at the faster time step $k_2$ ............73
Figure 5.3 (a) An example system graph decomposed into subsystems. (b) The interconnection of these subsystems is acyclic. ............74

Figure 5.4 Example 3-level hierarchy where $N=3$, $N_{sub}=4$, $\nu_2=\nu_3=3$, $I_{2,1}=\{1,2\}$, $I_{2,2}=\{3,4\}$, and $I_{1,1}=\{1,2,3,4\}$. The notation $x(\mathbf{k}_i)$ refers to a sequence of $x$ values at time steps $\{k_i,\ldots,k_i+N_p^i\}$. ....................75
Figure 5.5 Example system graph for numerical example. ....................88
Figure 5.6 Subsystem graphs for numerical example. ....................88
Figure 5.7 System graphs for numerical example. ....................89
Figure 5.8 Disturbance profile for numerical example with desired and actual inlet power flows. ....................90
Figure 5.9 Nominal and actual inputs for edges $e_8$ and $e_{16}$ with the nominal input constraint sets $\mathcal{U}_{8,16}$, the tightened input constraint set $\hat{\mathcal{U}}_{8,16}$ used by the subsystem controllers at Level 3 and, and the tighten input constraint set $\hat{\mathcal{U}}_{r,8,16}$ used by the vehicle controller at Level 1. ....................91
Figure 5.10 Nominal and actual state trajectories for vertex $v_{11}$ with error set showing the bounds on the deviation between $x_{11}$ and $\hat{x}_{11}$ for which the hierarchical controller is robust. ....................92
Figure 5.11 Nominal and desired power flow trajectories along edge $e_4$ with error set showing the bounds on the deviation between $\bar{P}_4$ and $\hat{P}_4$ for which the hierarchical controller is robust. ....................92
Figure 5.12 Deviations between the desired and nominal power flows for edges $e_8$ and $e_{14}$, which form $\Delta P^{in}$ for Subsystem 3, with the set $\Delta\mathcal{P}^{in}$ for Subsystem 3. ....................93
Figure 5.13 Nominal and actual state trajectories for vertex $v_{10}$ with error set showing the bounds on the deviation between $\hat{x}_{10}$ and $\hat{x}_{des,10}^{low}$ determined by the Vehicle controller at Level 1. ....................94
Figure 6.1 Sample interconnection of thermal (top) and hydraulic (middle) graphs, with actuator dynamics (bottom) affecting the hydraulic edge inputs. ....................100
Figure 6.2 Hydraulic and thermal graphs for individual components. ....................103

Figure 6.3 Example pump head map. 105
Figure 6.4 Candidate thermal power architecture for simulation and experimental validation. 114
Figure 6.5 Individual components and specification with a 6” ruler for scale. 115
Figure 6.6 Hydraulic and thermal graphs for the example experimental system configuration. 116
Figure 6.7 Pump PWM duty cycle inputs for hydrodynamic validation. 119
Figure 6.8 Selected outputs for hydrodynamic validation of experimental data with linear graph-based models. 119
Figure 6.9 Pump and heater inputs for thermodynamic validation. 120
Figure 6.10 Selected temperatures for thermodynamic validation of experimental data with linear graph-based models. 121
Figure 7.1 Three-level graph-based control hierarchy and signals. 124
Figure 7.2 Thermal disturbances for the simulation example, consisting of step changes in heat load to each cold plate (CP) heat exchanger. 131
Figure 7.3 States of the linear plant in the closed-loop simulation example, including cold plate wall temperatures (top) and a selection of outlet fluid temperatures (bottom). 131
Figure 7.4 Fluid pressures at the outlet of each pump and inlet to heat exchanger 2 (top), mass flow rates at the outlet of each pump and inlet to heat exchanger 2 (middle), and pump input signals (bottom). 132
Figure 7.5 Close-up view of state signals for pump 5 showing the pump state reference from the Hydraulic Control Layer, the commanded pump input by the Actuator Control Layer, and the achieved pump state. 133
Figure 7.6 Measured heat load applied to each cold plate matching the disturbance profile from Fig 7.2. 134
Figure 7.7 Measured temperatures from closed-loop control of the experimental system, including cold plate wall temperatures (top) and a selection of outlet fluid temperatures (bottom). 134

Figure 7.8 Measured fluid pressures at the outlet of each pump and inlet to heat exchanger 2 (top), measured mass flow rates at the outlet of each pump and inlet to heat exchanger 2 (middle), and pump input signals (bottom). ........................................ 135
Figure 8.1 Outline of developments required for the realization of hierarchical control of power flow in vehicle systems. .................................................................... 139
Figure A.1 Structure of Matlab files used to generate the robust hierarchical controller. ....... 149
Figure A.2 Structure of Matlab files called within the Simulink model used to simulate the robust hierarchical controller. ..................................................................... 150
Figure A.3 Simulink model used to simulate the robust hierarchical controller. .................... 177

# Chapter 1
# Introduction

## 1.1 Motivation and Background

Electrification of power systems is a societal megatrend, especially for vehicle systems such as aircraft, on- and off-road vehicles, and ships. For example, the onboard power for both military and commercial aircraft has grown rapidly over the last several decades (Fig. 1.1) and this growth is expected to accelerate, with an anticipated order-of-magnitude increase in power over the next 10 years [1]. With the majority of this power dedicated to onboard electrical systems, managing the heat generated by these systems has already become a major barrier. With the fuel in the F-35 and the cockpits of army helicopters overheating [2], [3], these highly advanced systems are unable to achieve their intended function at the risk of navigation and flight control systems shutting down midflight. In fact, over 50% of military electronics failures are attributed to thermal management problems [4].

Technological growth is currently limited by inadequate thermal management, and intelligent coordinated control is a crucial part of overcoming this barrier. In the absence of system redesign, cooperative control of electrical and thermal systems is the key to overcoming these barriers and maximizing the capability of these systems and the overall vehicle. As the demand for both performance and efficiency of these systems grows, the optimization of power generation, storage, distribution, and utilization becomes vital. Each vehicle is a system-of-systems, where power flow occurs in various modalities such as electrical, mechanical, thermal, and hydraulic. These systems interact with each other over a wide-range of timescales (Fig. 1.2) including the sub-millisecond time frame of voltage regulation in an electrical system [5] to the

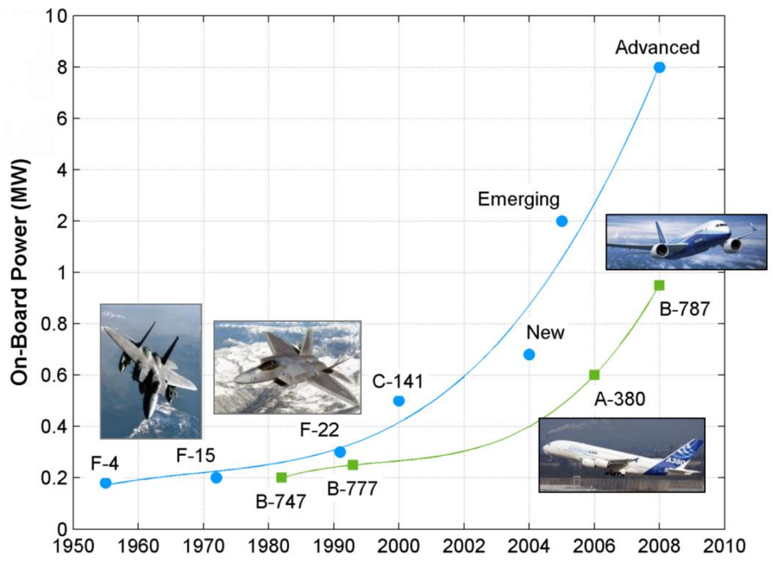


**Figure 1.1 Historical growth of onboard power for aircraft suggests effective power management will continue to be critical to the overall increase in capabilities of both military and commercial aircraft** [6]**.**

minutes time frame of fuel temperature changes in a thermal system [7]. Due to the size and complexity of these systems, the system and control designs often occur in a “siloed” framework, where each system is designed in isolation with limited design consideration regarding the dynamic interactions between these systems [8].

Future system and control design must adopt an alternative design procedure where system and subsystem interactions are directly considered and exploited in the design in order to achieve greater performance and efficiency. With increasing electrification, the opportunity for maximizing the performance of the aircraft as a whole hinges on the ability to coordinate the electrical and thermal systems. Due to the complexity of these systems and the need for robust operation under component failures, a distributed control approach is required. In such an approach, various parts of the system are operated by dedicated controllers, which coordinate via communication to meet system-wide objectives. Such control approaches will not only increase the total power and power density of these systems, they will also make these systems easier, safer, and cheaper to operate.

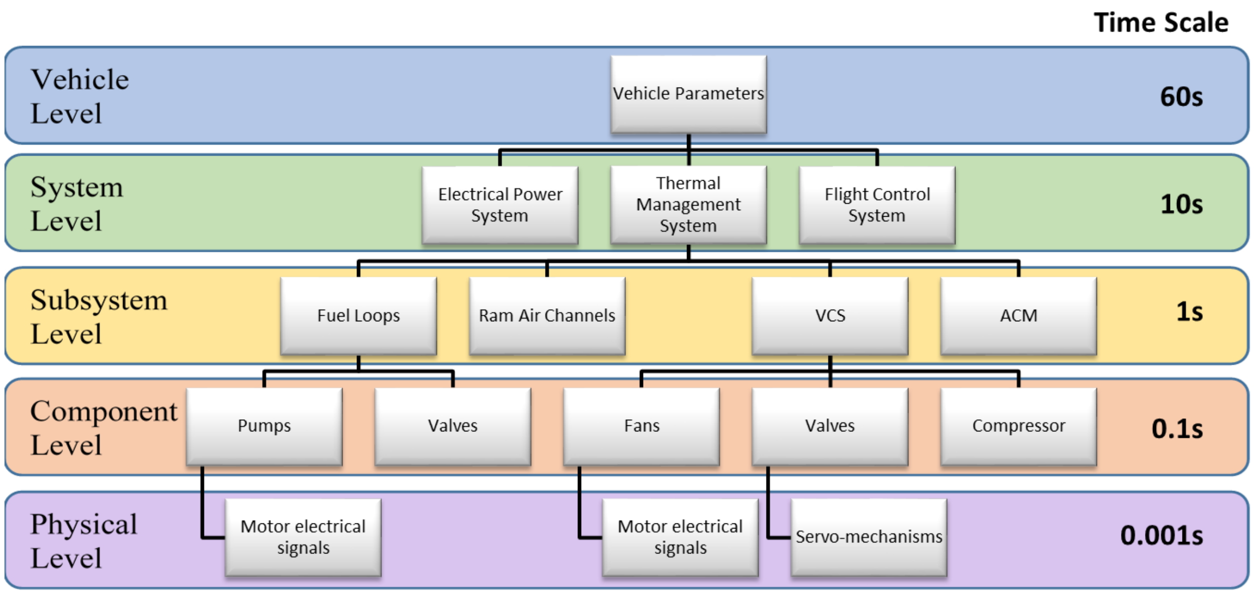


**Fig 1.2 Vehicle systems are a complex combination of interacting systems and subsystems over multiple timescales.**

## 1.2 Research Objectives

### 1.2.1 Problem Statement

Due to the long service life of many advanced vehicle systems, these systems are often tasked with operating well outside their initial intended design space. In the absence of system redesigns, the control system is responsible for improving vehicle capability, including increasing range, maximizing system operation duty-cycles, and expanding the overall operating envelop.

What is needed is a model-based anticipatory control strategy that directly considers interactions between systems and can determine energy/power allocation strategies for each system over a wide range of timescales. The **primary objective** of this dissertation is the development and evaluation of a model predictive control (MPC) based hierarchical control strategy specifically designed to optimize the power flow throughout the systems and subsystems of a vehicle over multiple timescales. While the focus of this dissertation is on thermal management in aircraft, the proposed hierarchical framework is developed to be:

- *widely applicable* to heterogeneous power flow systems of various energy domains, architectures, and components,
- *scalable* to large systems with many actuators, states, measurements, and control objectives,
- *robust* to model and signal uncertainty,
- *high performance*, via fast transient response, efficient operation, and constraint satisfaction, and
- *computationally efficient* for reduced computational cost and faster control decisions.

### 1.2.2 Dissertation Scope

In order to meet this primary objective, five secondary objectives have been identified which define the scope of this dissertation and, when achieved, will provide a generic hierarchical control framework which can be adopted to improve the capabilities of a wide-range of vehicle systems. These five secondary objectives are:

1. the formulation of a generic graph-based modeling framework that captures the energy storage and power flow dynamics in multiple energy domains and timescales with primary development and validation in the thermal domain,
2. the definition of a hierarchical control development algorithm with a step-by-step procedure for generating a hierarchical controller, applicable to a wide-range of vehicle systems with different architectures,
3. the formal analysis of the proposed hierarchical controller with respect to stability and robust feasibility,
4. the evaluation of hierarchical control performance using a series of example systems including simple educational examples and more realistic examples representative of vehicle thermal management systems, and
5. the experimental demonstration of a hierarchical controller to test the applicability of the proposed control approach to real-world thermal-fluid system dynamics including nonlinearity, unknown disturbances, and time delays.

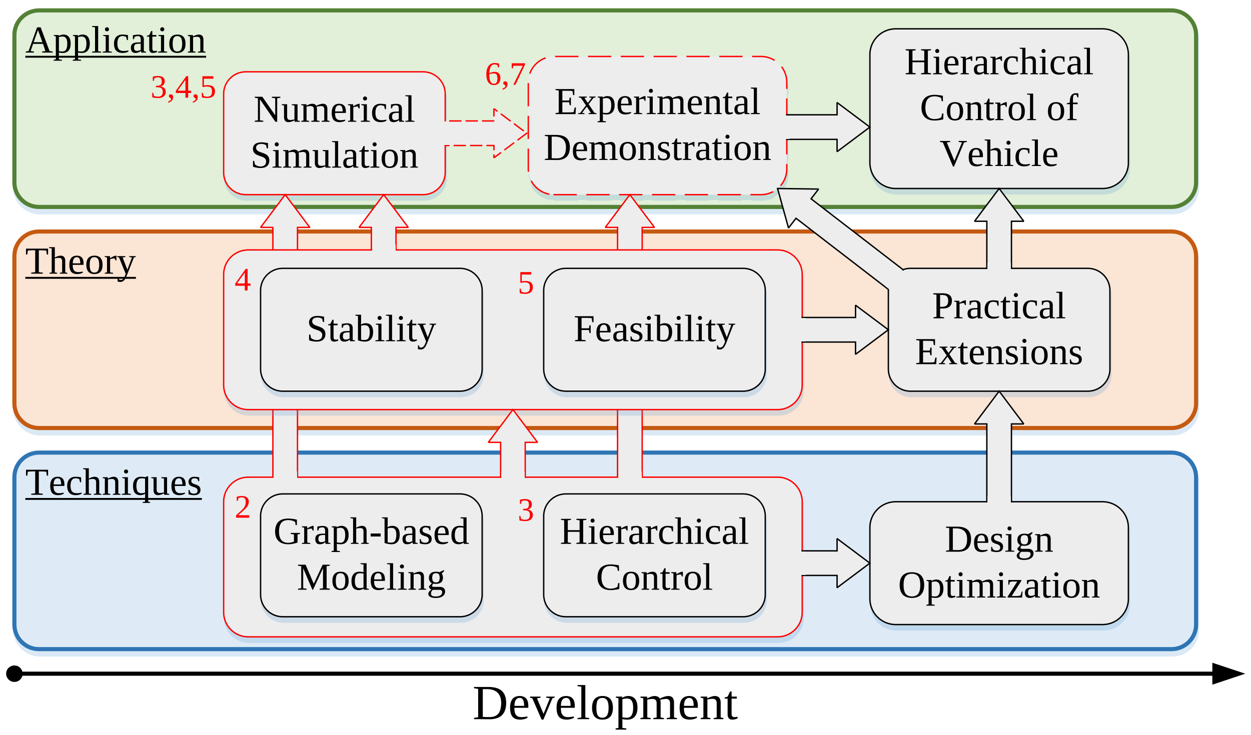


**Figure 1.3 Outline of developments required for the realization of hierarchical control of power flow in vehicle systems.**

## 1.3 Organization of Dissertation

Fig. 1.3 shows an outline of the techniques, theory, and application development that is needed to realize the overall goal of this dissertation: hierarchical power management in vehicle systems. The boxes and arrows outlined in red represent the developments and connections presented in this dissertation, where the red number represents the corresponding Chapter number. Chapter 2 introduces the general class of power flow systems and the graph-based modeling framework used to capture the energy storage and routing throughout the vehicle system. The model-based hierarchical control framework is presented in Chapter 3 with detailed procedures for graph modeling, system decomposition, model reduction, controller structure design, and optimization problem formulation. The procedures presented in Chapters 2 and 3 represent a set of generic techniques for modeling and control that are demonstrated through numerical simulation and built upon throughout the following Chapters. These generic modeling and control approaches are applied to a realistic thermal fluid system in simulation and experiment in Chapters 6 and 7. Specific formulations of these general techniques are used in Chapters 4 and 5 and are tested in numerical simulation only. In particular, Chapter 4 analyzes the closed-loop stability for a specific class of nonlinear graph-based power flow systems using the notion of passivity. Using a constraint tightening procedure, Chapter 5 develops a robustly hierarchical controller for linear graph-based power flow systems that guarantees feasibility in the presence of model and disturbance signal uncertainty.

The three black boxes to the right of Fig. 1.3 represent additional developments that build upon the work presented in this dissertation, which are required to achieve highly functional hierarchical control of vehicle systems. The Experimental Demonstration block and arrow from Numerical Simulation are outlined with dashed red lines to denote the fact that the basic techniques from Chapter 2 and 3 have been demonstrated on an experimental system, while practical extensions are required to directly apply the theoretical results from Chapters 4 and 5 to the physical system in Chapters 6 and 7. These extensions are discussed in greater detail with the concluding remarks and future research directions provided in Chapter 8.

## 1.4 Notation

The symbol $\mathbb{R}$ denotes the set of real numbers. The notation $[1,N]$ denotes the set of integers from 1 to $N$. A vector $v$ with elements $v_i$ is defined as $v=[v_i]$. Similarly, a matrix $M$ with elements $m_{jk}$ in the $j^{th}$ row and $k^{th}$ column is defined as $M=[m_{jk}]$. For the scalar function $f(x)$, $\mathcal{N}(f(x))=\{x \mid f(x)=0\}$ denotes the zero set of $f(x)$. The eigenvalues of matrix $A\in\mathbb{R}^{n\times n}$ are $\lambda_k(A)$, $k\in[1,n]$ and their real part is denoted $\operatorname{Re}\lambda_k(A)$, $k\in[1,n]$. For sets $\mathcal{X},\mathcal{Y}\subset\mathbb{R}^n$, the Minkowski sum is $\mathcal{X}\oplus\mathcal{Y}\triangleq\{x+y \mid x\in\mathcal{X}, y\in\mathcal{Y}\}$ and for sets $\mathcal{Y}\subset\mathcal{X}$, the Pontryagin difference is $\mathcal{X}\ominus\mathcal{Y}\triangleq\{x\in\mathbb{R}^n \mid \mathcal{Y}+x\subset\mathcal{X}\}$. For a set $\mathcal{X}\subset\mathbb{R}^n$ and the linear mapping $A:\mathbb{R}^n\to\mathbb{R}^m$, $A\mathcal{X}\triangleq\{Ax \mid x\in\mathcal{X}\}$. A set $\mathcal{X}\subset\mathbb{R}^n$ is robust positively invariant (RPI) for a system $x(k+1)=f(x(k),w(k))$ if and only if for all $x\in\mathcal{X}$ and all $w\in\mathcal{W}$ it holds that $f(x(k),w(k))\in\mathcal{X}$. The right inverse of $A\in\mathbb{R}^{n\times m}$ is defined as $A^{\dagger}=A^T\left(AA^T\right)^{-1}$.

# Chapter 2
# Graph-based Power Flow Systems

## 2.1 Power Flow Systems

Power flow systems are a wide class of systems where modeling and control is motivated by the need to manage the storage and routing of energy. Examples include thermal energy systems [9]–[11], water distribution networks [12]–[14], electrical power grids [15], [16], chemical process networks [17], [18], and multi-energy domain vehicle systems such as aircraft [19]–[21], mining equipment [22], [23], on- and off-road vehicles [24], [25]. In general, these systems function based on the storage, conversion, and routing of conserved quantities such as mass and energy. For example, electrical systems store energy in capacitors and batteries, route energy through wires using switches, and convert electrical energy into thermal energy in the form of heat. Thermal fluid systems conserve both mass and thermal energy which are stored in fluid tanks and heat exchangers, routed through pipes using pumps and valves, and reject heat to the surrounding environment. A key feature of power flow systems is the dynamic interaction between systems and subsystems of various energy domains through the conversion of conserved energy.

For vehicle systems in particular, achieving peak performance often requires controlling various systems and subsystems at the limit of their operating envelopes. Thus, system operation is often characterized by operation against actuator and state constraints. Additionally, the operation of these systems is highly coupled. For example, an electrical system generates heat that is managed by the thermal system, but this thermal system uses electrical power to operate. Thus higher electrical power results in higher heat generation which results in more electrical consumption by the thermal management system. Additionally, systems such as the electrical system and the hydraulic system on an aircraft both take power off the engine, creating a resource allocation problem based on the needs of each system and the maximum allowable power draw from the engine.

The dynamics of each system and the interactions between systems of various energy domains occur over a wide range of timescales including the sub-millisecond time frame of voltage regulation in an electrical system [5] to the minutes time frame of fuel temperature changes in a thermal system [7]. Effectively controlling the power flow system at each timescale is critical to achieving robustness to disturbances and faults, maximizing transient performance, and preventing constraint violations.

Due to the size and complexity of many power flow systems, the system and control designs often occur in a "siloed" framework, where each of the systems and subsystems is designed in isolation with limited design consideration regarding the interactions between these systems and energy domains. Future system and control design must adopt an alternative design procedure where the interactions are directly considered and exploited to achieve greater performance and efficiency. A unified modeling framework that captures the dynamics of system with multiple interacting energy domains is the enabling first step to achieving this coordination.

## 2.2 Modeling Objectives

Conventional approaches to modeling and control of complex system-of-systems are often limited to decentralized high-fidelity modeling and robust, low performance proportional-integral and logic-based control [26]. The proposed model-based hierarchical control approach aims to improve performance through coordination among subsystems and timescales. With a hierarchical MPC framework, each controller in the hierarchy requires a model of the system dynamics under its control to predict future state trajectories and determine optimal control sequences. For many power flow systems, holistic modeling, analysis, and control design is inhibited by the complexity and size of the systems, especially when dynamics evolve over a wide range of timescales and energy domains. Thus the main desired features for a control-oriented modeling framework are:

- *modularity* – such that large systems can be built from the combination of individual components,
- *energy domain agnostic* – allowing systems with multiple, interacting energy domains to be represented using a single, unifying modeling framework,
- *timescale agnostic* – providing a generic approach that captures dynamics over a wide range of dynamically interacting timescales, and
- *variable fidelity* – both in terms of the number of states used to capture the dynamics of each component and the complexity of the relationships used to represent power flow (linear vs nonlinear vs bilinear).

As shown in the following Sections, a graph-based modeling framework provides each of these features, resulting in an ideal framework for control-oriented modeling and the development of hierarchical controllers for power flow systems.

## 2.3 Graph-based Modeling

From a bond graph perspective [27], power is the product of effort and flow, $P = e \cdot f$. Typical forms of effort include force and torque in mechanical systems, voltage potential in electrical systems, pressure difference in hydraulic systems, and temperature difference in thermal systems. The corresponding forms of flow are linear or angular velocity, current, volumetric flow rate, and entropy flow rate, respectively. With power representing the transport of energy, each domain also has the ability to store energy in the form of linear or angular momentum, electrical charge, mass, or thermal energy.

While bond graphs are a powerful tool and can be used to derive the governing differential equations for a dynamic system [27], an alternative, graph-based, system representation and modeling technique has been widely adopted. Compared to bond graphs, graph-based system model more readily captures the structure of the governing mass and energy conservation laws for these systems. As will be shown in Chapters 3-5, the structure of these graphs can be directly used for control architecture design and analysis. Graph-based modeling approaches have been used in a variety of application areas such as chemical processing plants [28], [29], building thermal systems [30], [31], electronic circuits [32], and flow control systems [33]. In this graph-based framework, vertices, or nodes, represent capacitive elements that store energy and edges represent paths for power flow between these capacitive elements. The following Section presents the generic graph-based modeling formulation and Section 2.5 demonstrates how this approach addresses each of the desired modeling features.

## 2.4 Generic System Formulation

Let a power flow system $\mathbf{S}$ be represented by an oriented graph $\mathcal{G} = (V, E)$ of order $N_v$ with set of vertices $V = \{v_i\}$, $i \in [1, N_v]$ and of size $N_e$ with set of edges $E = \{e_j\}$, $j \in [1, N_e]$. Each oriented edge $e_j \in E$ represents a path for power flow in $\mathbf{S}$, where positive power $P_j$ flows from the tail vertex $v_j^{tail}$ to the head vertex $v_j^{head}$. Each vertex $v_i \in V$ has an associated state $x_i$ that represents the amount of energy stored in that vertex. Thus the dynamic for the state of each $v_i$ satisfies the energy conservation equation

$$C_i \dot{x}_i = \sum_{e_j \in E_i^{in}} P_j - \sum_{e_j \in E_i^{out}} P_j, \tag{2.1}$$

where $C_i > 0$ is the energy storage capacitance of vertex $v_i$ and $E_i^{in} = \{e_j \mid v_j^{head} = v_i\}$ and $E_i^{out} = \{e_j \mid v_j^{tail} = v_i\}$ are the sets of edges oriented into and out of the vertex $v_i$, respectively.

The most general form for the algebraic relationship between the power flows along edge $e_j$ and the states $x_j^{tail}$ and $x_j^{head}$ is

$$P_j = f_j\left(x_j^{tail}, x_j^{head}, u_j\right), \tag{2.2}$$

where $u_j$ is the actuator input associated with the edge. Common, increasingly simple, forms for this power flow relationship are the nonlinear, input affine form

$$P_j = f_j\left(x_j^{tail}, x_j^{head}\right) + g_j\left(x_j^{tail}, x_j^{head}\right) u_j, \tag{2.3}$$

where $f_j$ and $g_j$ are nonlinear, the bilinear form

$$P_j = \overline{f}_j\left(x_j^{tail}, x_j^{head}\right) + \overline{g}_j\left(x_j^{tail}, x_j^{head}\right)u_j, \tag{2.4}$$

where $\overline{f}_j$ and $\overline{g}_j$ are linear, and the linear form

$$P_j = a_j x_j^{tail} + b_j x_j^{head} + c_j u_j + d_j. \tag{2.5}$$

In general, the system $\mathbf{S}$ has states $x \in \mathbb{R}^{N_v}$ that each satisfy (2.1) and power flows $P \in \mathbb{R}^{N_e}$ that each satisfy (2.2). The disturbances to $\mathbf{S}$ consist of how power enters and exits the system with inlet power flows $P^{in} \in \mathbb{R}^{N_s}$ and sink states $x^t \in \mathbb{R}^{N_t}$. As indicated by the dashed lines in Fig. 2.1, the inlet power flow edges are not included in $\mathcal{G}$. These power flows into the system are analogous to *demand* in the network flow literature [33], [34] and *inflows* from the compartmental systems literature [35].

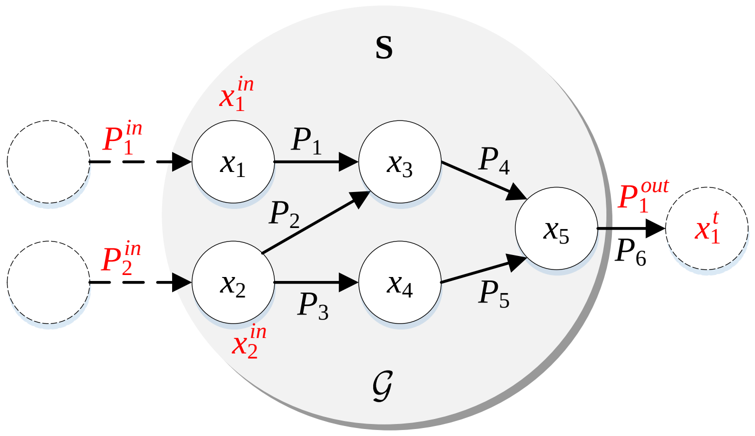


**Figure 2.1 Notional system exemplifying the graph-based power flow representation with key power flows and states highlighted in red. Dashed lines indicate elements that serve as disturbances to the system.**

Also indicated by dashed lines in Fig. 2.1, the sink states are *not* states of $\mathbf{S}$, but the sink vertices and the edges connecting $\mathbf{S}$ to the sink vertices *are* included in $\mathcal{G}$. Power flows along this type of edge, denoted $P^{out} \in \mathbb{R}^{N_t}$, each follow the relationship from (2.2). These sink vertices represent the surrounding environment and are referred to as *external nodes* in the compartmental systems literature [33]. Finally, each system has a subset $x^{in} \in \mathbb{R}^{N_s}$ of the states $x$ which represent the states directly affected by the inlet power flows $P^{in}$.

Let $M = \left[ m_{i,j} \right] \in \mathbb{R}^{(N_v + N_t) \times N_e}$ be the incidence matrix of graph $\mathcal{G}$ [36] where

$$m_{i,j} = \begin{Bmatrix} +1 & \text{if } v_i \text{ is the tail of } e_j \\ -1 & \text{if } v_i \text{ is the head of } e_j \\ 0 & \text{else} \end{Bmatrix}. \tag{2.6}$$

Then, based on (2.1), the system dynamics are

$$\begin{bmatrix} C\dot{x} \\ \dot{x}^t \end{bmatrix} = -MP + \begin{bmatrix} D \\ 0 \end{bmatrix} P^{in}, \tag{2.7}$$

where $C = diag\left( \left[ C_i \right] \right)$ is a diagonal matrix of the vertex capacitances and $D = \left[ d_{i,j} \right] \in \mathbb{R}^{N_v \times N_s}$ where

$$d_{i,j} = \begin{Bmatrix} 1 & \text{if } v_i \text{ is the head of } P_j^{in} \\ 0 & \text{else} \end{Bmatrix}. \tag{2.8}$$

Since $x^t$ are disturbances to the system, not states, $M$ is partitioned as $M = \begin{bmatrix} \bar{M} \\ \underline{M} \end{bmatrix}$, with $\bar{M} \in \mathbb{R}^{N_v \times N_e}$ and $\underline{M} \in \mathbb{R}^{N_t \times N_e}$, resulting in

$$C\dot{x} = -\bar{M}P + DP^{in}. \tag{2.9}$$

From this generic formulation, specific formulations are derived in Chapters 3-6.

For this dissertation, all inputs $u_j$ are assumed to be continuous between lower bound $\underline{u}_j$ and upper bound $\bar{u}_j$. As discussed in Section 8.2, future work should extend the proposed modeling and hierarchical control approaches to systems with continuous as well as discrete actuators, where the possible inputs values form a set of discrete values, such as the on/off states of an electrical switch, $u_j \in \{0,1\}$. While the control optimization problem for a linear system with continuous actuators can be formed as a quadratic programming (QP) problem, controllers of systems with discrete actuators are formulated as mixed integer quadratic programming (MIQP) problems, which require significantly more computational resources to solve.

The system **S** is assumed to have $N$ dynamic timescales, motivating the use of hierarchical control, where each level of the hierarchy is responsible for control decisions for a corresponding timescale. For a graph-based system, the timescales can be roughly identified by the capacitance of the vertices. Thus the state vector is subdivided as

$$x = \begin{bmatrix} \boldsymbol{x}_1^T & \boldsymbol{x}_2^T & \dots & \boldsymbol{x}_N^T \end{bmatrix}^T \tag{2.10}$$

where $\boldsymbol{x}_i \in \mathbb{R}^{N_v^i}$ denotes a vector of states with the $i^{th}$ timescale, $\sum_{i=1}^{N} N_v^i = N_v$, and $C_j < C_i < C_k$ for $x_j \in \boldsymbol{x}_j$, $x_i \in \boldsymbol{x}_i$, $x_k \in \boldsymbol{x}_k$, $j > i > k$.

## 2.5 Modeling Features

The following demonstrates the features that make a graph-based modeling framework well suited to modeling the dynamics of energy storage and power flow in a vehicle system.

### 2.5.1 Modularity

Each component of the system is modeled with a component graph of various vertices and edges and the overall system graph is simply constructed through the connection of the individual component graphs based on the system architecture. For a simple demonstration, consider the system in Fig. 2.2 consisting of a pump, cold plate heat exchanger, reservoir, and liquid-to-liquid heat exchanger.

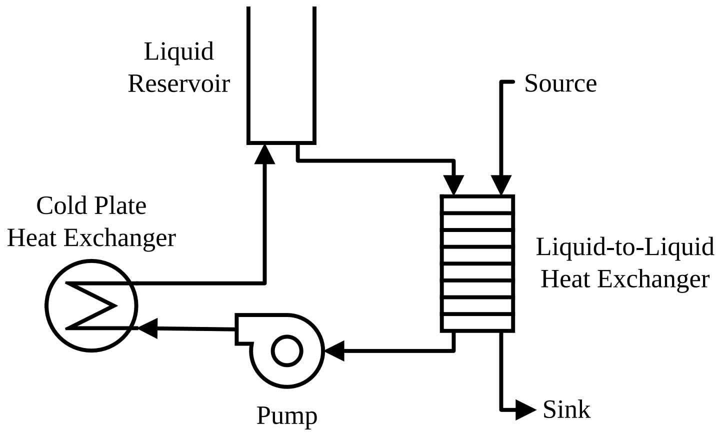


**Figure 2.2 Simple example system used to demonstrate the modularity of a graph-based modeling framework.**

To capture the energy storage and transport in the system each component can be modeled as a graph where the states $x_i$ represent temperatures and the power flows $P_j$ represent the flow of energy. Fig. 2.3 shows the graphs used to capture the energy dynamics of each component. The temperature $T$ of the fluid in the reservoir is modeled with a single vertex. The fluid entering the reservoir adds energy at the rate $P^{in} = \dot{m}_1 c_p T_1$ where $T_1$ is the temperature of the incoming fluid. The fluid exiting the reservoir removes energy at the rate $P^{out} = \dot{m}_2 c_p T$. From a conservation perspective, the pump graph is identical to the reservoir graph. Only the capacitance $C$ associated with the vertices would be different. An additional source vertex and edge could be added to the pump graph to represent the heat added to the fluid due to inefficiencies of the pump. The cold plate heat exchanger is represented with 2 vertices representing the fluid temperature $T$ and the cold plate wall temperature $T_w$. Energy enters the cold plate in the form of the heat load $P_2^{in} = Q$ and the inlet fluid flow $P_1^{in} = \dot{m}_1 c_p T_1$. Energy exits the cold plate with the outlet fluid flow $P^{out} = \dot{m}_2 c_p T$. Energy is transferred between the two vertices of the cold plate through convection between the wall and the fluid $P = hA_s\left(T_w - T\right)$. The liquid-to-liquid heat exchanger is modeled similarly to the cold plate but with two liquids, $a$ and $b$, that exchange energy through the heat exchanger wall.

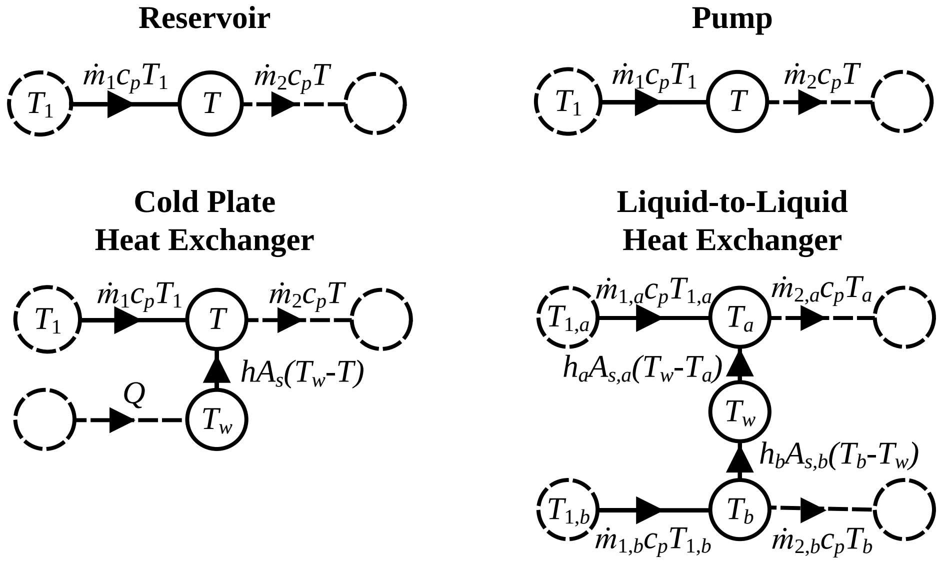


**Figure 2.3 Component graphs for each component in the example system.**

Each solid lined vertex in Fig. 2.3 follows the conservation law from (2.1) and has a corresponding state $x_i$ and capacitance $C_i$. The dashed line vertices represent sources and sinks of power flow that correspond to the vertices of neighboring components in the system or external disturbances such as the surrounding environment. Each solid lined edge in Fig. 2.3 follows the generic power flow equation from (2.2) that relates variables such as liquid properties, temperature, and mass flow rate to power flow.

With the vertex and edge parameters defined for each individual component graph model, the entire system graph can be formulated based on the structure of component interconnections. Fig. 2.4 shows the system graph for the example system from Fig. 2.2. This system has 7 states, $N_v = 7$, 8 edges, $N_e = 8$, two source power flows, $N_s = 2$, and 1 sink power flow $N_t = 1$. The ability to define the parameters for each vertex and edge individually, create component graphs, and combine component graphs to make system graphs provides the desired modularity and scalability. Further demonstration of this modeling approach is presented for an experimental thermal fluid system in Chapters 6.

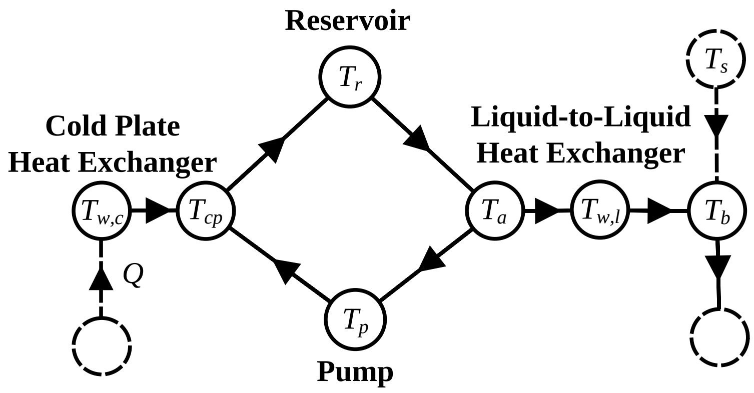


**Figure 2.4 Example system graph.**

Note that a similar graph could be constructed to capture the mass conservation and fluid flow dynamics of the system. Chapters 6 demonstrates the relationship between the energy and mass conservation graphs when modeling and controlling a thermal fluid system.

### 2.5.2 Energy Domain Agnostic

A vertex represents the storage of energy regardless of whether it is thermal energy stored in the fuel of a fuel tank, electrical energy stored in a battery, mechanical energy stored in the rotation of an engine, or pneumatic energy stored in a piston cylinder. An edge represents a path for energy to flow between capacitive elements. Only the equation used to capture the relationships between this power flow and the neighboring states and associated actuator input depend on the energy domain. This power flow equation can be made general enough to closely capture these relationships for multiple energy domains with a single relationship.

The thermal fluid system from Section 2.5.1 provided an introduction to modeling a system as a graph. For a thermal system where the state $x$ represents a temperature $T$, the capacitance $C$ corresponds to the total thermal capacitance of the component, $C = \rho V c_p$ where the component has density $\rho$, volume $V$, and specific heat $c_p$. For thermal systems, power flow generally takes one of two forms. Power flow due to advection, the transport of fluid, has the form $P = \dot{m} c_p T$ where $\dot{m}$ is the fluid flow rate. Power flow due to convection has the from $P = hA\left(T^{tail} - T^{head}\right)$ where $h$ is the average heat transfer coefficient over area $A$ and $T^{tail}$ and $T^{head}$ are the temperatures of the tail and head vertices for that edge.

With the general form for power flow from (2.2), a large variety of power flow relationships of various energy domains can be captured within the graph-based modeling framework. While this dissertation primarily focuses on the development and analysis of hierarchical control for arbitrary graphs and the application to thermal fluid systems, ongoing and future work is extending this modeling approach to include electrical, pneumatic, hydraulic, and mechanical energy domains.

### 2.5.3 Timescale Agnostic

Each vertex $v_i$ has an associated capacitance $C_i$ which relates how the net power flow into a vertex affects the rate of change for the associated state $x_i$ of that vertex. A large capacitance represents a dynamic with a slow timescale while a small capacitance represents a dynamic with a fast timescale. Thus a single graph can have vertices with capacitances of highly varying magnitude but these magnitudes do not affect the modeling approach or edge power flow equations. When determining timescales based on the magnitude of the capacitance, it is important to consider the units and magnitude of the associated state, especially when including multiple energy domains in a single graph.

### 2.5.4 Variable Fidelity

A graph-based approach is a specific type of a lumped parameter approach where a single vertex, and associated state, is used to represent time-varying aspects of a component that might also vary spatially. For example, a single vertex could be used to represent the energy stored, and the corresponding temperature, of the walls of a heat exchanger. While in reality, the temperature of the walls can vary significantly by location between the inlet and outlet of the heat exchanger, a single representative temperature can often be used to capture the heat transfer dynamics for the heat exchanger. However, if additional model accuracy is required, additional vertices, and thus states, can be added to the graph model of the component. For example, the cold plate heat exchanger graph model from Fig. 2.3 represents the fluid temperature in the heat exchanger as a single vertex with a single temperature. However, in reality, the fluid is continuously changing temperature as it flows through the heat exchanger. Fig. 2.5 shows how additional vertices could be added to the heat exchanger graph model to increase the fidelity of the model.

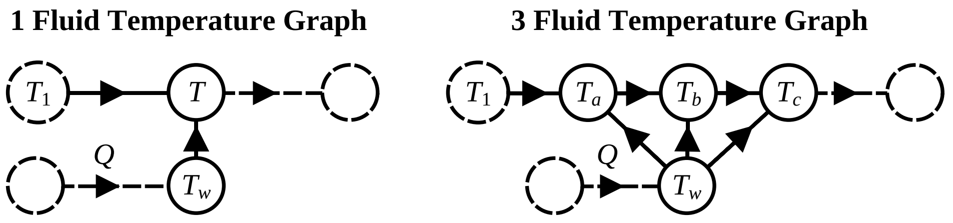


**Figure 2.5 Two graph models for a cold plate heat exchanger, where the fluid temperature is either represents as a single lumped temperature $T$ or three distinct temperatures $T_a, T_b, T_c$ along the length of the heat exchanger.**

Additionally, a graph-based approach can achieve variable fidelity based on the form of the power flow equation for each edge. At the most general, this power flow is nonlinear. However, to be more amenable to analysis and controller development, bilinear and linear approximations of this power flow relationship can be used at the cost of model accuracy. For example the power flow for advection is $P = \dot{m}c_pT$. If $\dot{m}$ is considered the input $u$ for this power flow and $T$ is the state of the tail vertex, this power flow relationship is bilinear. If a linear power flow relationship is desired, the bilinear power flow equation can be linearized about the nominal mass flow rate $\dot{m}_0$ and temperature $T_0$ as

$$\begin{aligned} P = \dot{m}c_pT &\approx \dot{m}_0c_pT_0 + \dot{m}_0c_p\left(T - T_0\right) + c_pT_0\left(\dot{m} - \dot{m}_0\right), \\ &= \dot{m}_0c_pT + c_pT_0\dot{m} - \dot{m}_0c_pT_0. \end{aligned} \tag{2.11}$$

## 2.6 Chapter Summary

Based on the features of the generic graph-based modeling framework presented in this Chapter, Chapters 3-6 use specific graph formulations to:

1. establish a generic hierarchical control development procedure,
2. assess the stability of closed-loop graph-based nonlinear power flow systems,
3. formulate and analyze a robustly feasible hierarchical control framework for linear systems, and
4. demonstrate the generic approach on a realistic thermal fluid system, respectively.

Each Chapter uses different example systems to best illustrate the specific contributions of each Chapter, demonstrating how this generic graph-based modeling framework can be easily tailored to specific classes of systems.

# Chapter 3
# Hierarchical Model Predictive Control

## 3.1 Conceptual Framework

When a power flow system is decomposed as Fig. 1.2 does for an aircraft system, the dynamics and resulting control decisions form a natural hierarchy. The overall vehicle is composed of multiple systems (e.g. electrical, thermal, flight control, etc.) and each system is composed of multiple subsystems. A thermal management system, for example, may consist of a fuel system, an air conditioning system for the cockpit/cabin, a vapor compression system, and/or an air cycle machine. Each of these subsystems, contains multiple components such as pumps, valves, fans, and heat exchangers. Finally, many components have actuators and sensors that might have their own dynamics. This forms the five-level hierarchy shown in Fig. 1.2 where the levels are referred to as the Vehicle, System, Subsystem, Component, and Physical Levels. From this hierarchical decomposition of a system, the control decisions can be decomposed similarly, resulting in a hierarchical controller similar to the one shown in Fig. 3.1. While this structure is fairly generic, a control hierarchy can consist of more or fewer control levels, with varying numbers of controllers at each level and a single controller at the top. The term hierarchy refers to the communication structure where controllers only communicate with the controllers directly above and below them in the hierarchy. Thus controllers at the same level do not communicate, significantly reducing the total information communication throughout the hierarchy.

Starting from the bottom of Fig. 3.1, the power flow system (Plant) has dynamics that evolve continuously, potentially over a wide range of timescales and energy domains. The bottom level of the hierarchical controller (Physical Level) determines the appropriate control signal to send to the system's actuators to achieve the desired actuation. This desired actuation is the control decision made by the controller one level higher in the hierarchy (Component Level). At this level, each controller determines how to best utilize a corresponding component to achieve the desired performance determined by the subsystem controller one level higher in the hierarchy (Subsystem Level). The subsystem controllers are responsible for determining how a subsystem should operate to achieve the overall desired operation of the system. Similarly, the system controllers are responsible for determining how a system should operate to achieve the overall desired operation of the vehicle. The Vehicle Level controller uses information about the performance and efficiency objectives for the overall vehicle and any available information about know disturbances to coordinate the behaviors of the constitutive systems in achieving these objectives.

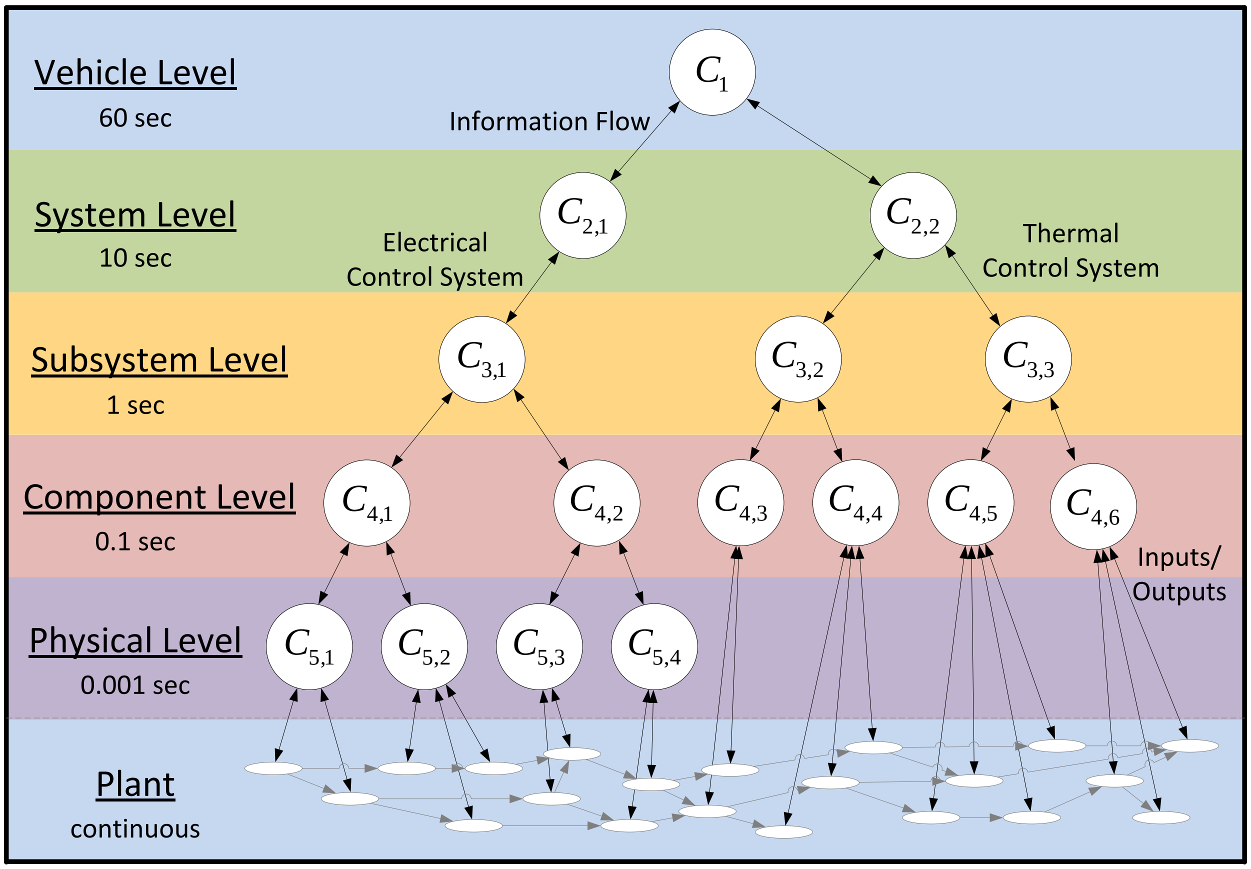


**Figure 3.1 Notional 5-level hierarchy with notional controller update rates for an electrical and thermal system.**

In addition to this functional decomposition based on systems, subsystems, and components, the hierarchy also provides a temporal decomposition. As notionally indicated in Fig. 3.1, the update rate of the controllers at each level decreases for higher levels of the hierarchy. This allows each control level to effectively and efficiently determine state trajectories for dynamics with a corresponding timescale. Upper-level controllers utilize a slower update rate to better control slower dynamics in the system while lower-level controllers utilize a faster update rate to better control the faster dynamics. This matching of controller update rates with system dynamic timescales can provide significant control performance advantages, as discussed in the following Section.

## 3.2 Hierarchical Control Advantages

When compared to a centralized control approach, a hierarchical controller has two primary advantages relating to the temporal and functional decomposition of the power flow system. While the dynamics of the system occur over a wide range of timescales, a centralized MPC controller only has a single time step $\Delta T$ and a single number of discrete steps in the prediction horizon $N_p$. The computational cost of the controller is directly affected by $N_p$ and thus $N_p$ is chosen based on the computational resources available. Therefore, $\Delta T$ becomes the primary decision variable when designing a centralized controller for a multi-timescale system.

Assuming $N_p$ is fixed, Fig. 3.2 demonstrates the effects of $\Delta T$. Consider a simple system consisting of a slow dynamic (e.g. the temperature of the fuel in a fuel tank, blue line in Fig. 3.2) and a fast dynamic (e.g. the temperature of a fuel cooled electrical load, pink line in Fig. 3.2). Assume that a disturbance (red line in Fig. 3.2) affects the system which consists of a large pulse, which is known ahead of time, along with some small, high-frequency variations which are unknown. A MPC controller with a large $\Delta T$, and thus a long prediction horizon $\left(N_p \cdot \Delta T\right)$, observes the upcoming large pulse disturbance and can begin to precool the fuel as shown in Fig. 3.2a. This precooling prevents the fuel temperature from reaching its maximum safe temperature (dashed line in Fig. 3.2). However, since the controller updates slowly, it cannot reject the high-frequency disturbance that causes the fast dynamic state to deviate significantly from the desired value, which may be highly undesirable due to the additional thermal fatigue placed on the electrical components. Alternatively, Fig. 3.2b shows the result of a controller with a small $\Delta T$. Now the controller is able to reject the high frequency disturbance but since the prediction horizon is short, the fuel tank is not sufficiently precooled, which results in a constraint violation due to the large pulse disturbance. A centralized controller with a small $\Delta T$ and a very large $N_p$ could potentially provide effective control in the presence of both types of disturbances. However, the excessive computational cost could render this approach infeasible in many applications, especially in vehicle systems where all controller computation is performed onboard using limited computation resources.

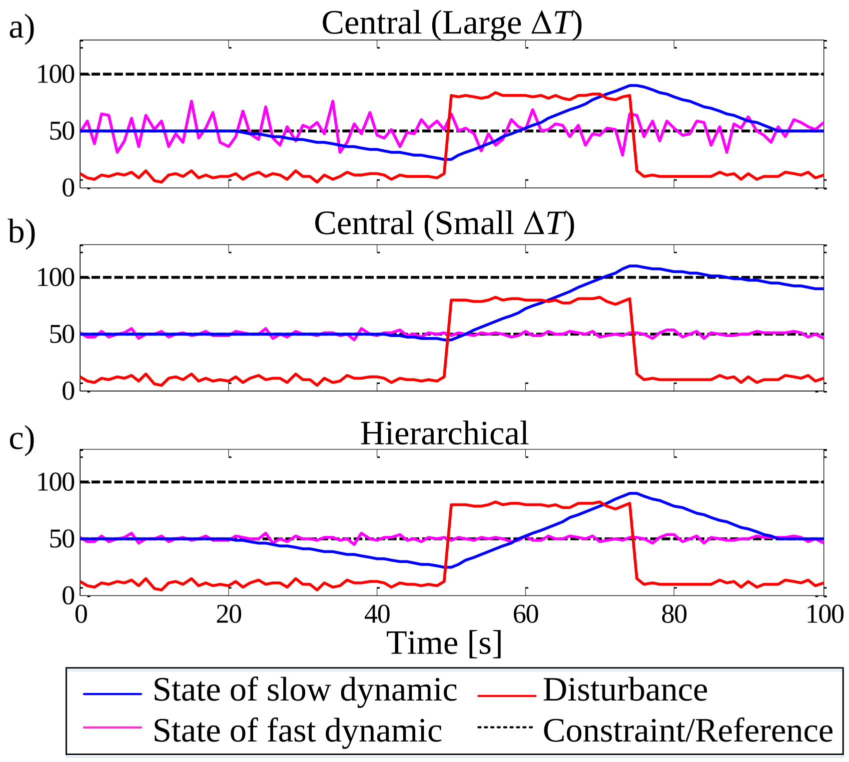


**Figure 3.2 The effects of $\Delta T$ on a centralized controller with regard to large known disturbances and small high-frequency unknown disturbances compared to a hierarchical control approach.**

A hierarchical control approach, however, uses the multiple levels of control in order to predict far into the future using large $\Delta T$ for the upper-level controllers as well as respond quickly to unknown disturbances using small $\Delta T$ for lower-level controllers. Thus $N_p$ can be relatively small at each level of the hierarchy, reducing overall computational cost. Fig. 3.2c shows how the hierarchical controller is able to combine the fast dynamic regulation performance of the small $\Delta T$ centralized control with the precooling and constraint satisfaction of the large $\Delta T$ centralized controller.

The second primary advantage of a hierarchical controller relates to the functional partitioning of the system. As with decentralized and distributed controllers, no one controller in the hierarchy has a model of all the dynamics of the system. A centralized controller, which utilizes a model of the entire system, can make control decisions based on the known coupling throughout the system. Lack of knowledge of this coupling is what limits the performance and forces the iterative or conservative nature of many decentralized and distributed control approaches [37]. However, the proposed hierarchical control approach has the advantage of directly accounting for the coupling in the plant. Fig. 3.3 demonstrates how the coupling between two subsystems A and B is directly addressed in the proposed hierarchical framework. Power flows from state $x_1$ in subsystem A and enters subsystem B through state $x_2$. This power flow may be a function of $x_1$, $x_2$, and an actuation input $u$. Assume the input is determined by the controller for subsystem A. This power flow creates a coupling between the two subsystems. In decentralized control, the state $x_2$ in subsystem B would be treated as an unknown disturbance affecting subsystem A and the power flow would be an unknown disturbance affecting subsystem B. In the proposed hierarchical control framework, however, the power flow and the state $x_2$ are decision variables of the system-level controller. The desired value of $x_2$ is sent as a predicted disturbance to the controller for subsystem A and the desired power flow is sent as a predicted disturbance to the controller for subsystem B. To ensure consistency, the subsystem A controller is designed to track the desired power flow using the actuator input $u$ and the subsystem $B$ controller is designed to track the desired value for $x_2$ using the actuators in subsystem $B$ (not shown in Fig. 3.3). In this way, the hierarchical control framework, while decomposing the power flow system into systems and subsystems, is still able to directly consider the coupling between these systems and subsystems, resulting in significantly improved control performance compared to a decentralized approach.

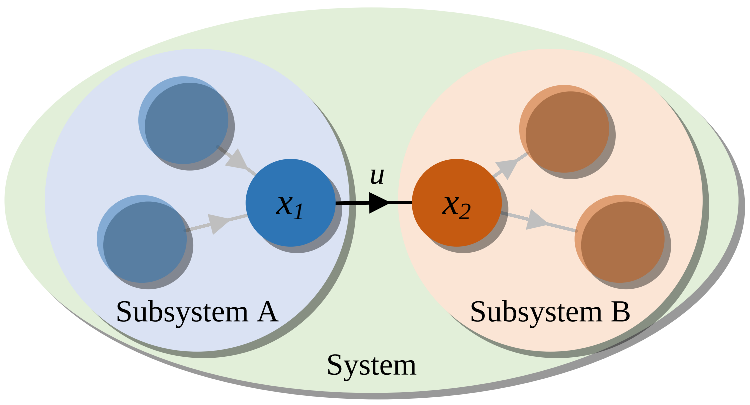


**Figure 3.3 A system with two interconnected subsystems used to demonstrate the ability of a hierarchical control framework to directly account for the coupling between systems and subsystems.**

The following Sections summarize the hierarchical control framework and then formalize the procedure by first decomposing and modeling a system as a graph and then detailing the hierarchical control framework and MPC controller development.

## 3.3 Hierarchical Control Development Procedure

The following algorithm summarizes the proposed procedure for modeling and hierarchical control of a multi-energy domain, multi-timescale power flow system. For notational simplicity and clarity, and without loss of generality, a three-level control hierarchy is assumed for a vehicle composed of systems and subsystems with slow, medium and fast dynamics.

*Hierarchical Control Development Algorithm*

1) Model the power flow system dynamics as a graph based on the procedure introduced in Chapter 2.
2) Partition the graph into systems and subsystems and slow, medium, and fast dynamics.
3) Construct new graphs to capture timescale relevant dynamics at the subsystem, system, and vehicle level.
4) Identify necessary information communication throughout the hierarchy.
5) Formulate the individual MPC controllers at each level.

The details of each step in this process are presented in Sections 3.4-3.7 and applied for a numerical example in Section 3.8.

## 3.4 Graph-based System Model

Following the generic graph-based modeling framework presented in Chapter 2, the following specific formulation is used for this Chapter. A bilinear power flow relationship is assumed where

$$P_j = \left(u_j + a_j\right)\left(b_j x_j^{tail} + c_j x_j^{head} + d_j\right), \ \forall e_j \in E, \tag{3.1}$$

where $a_j, b_j, c_j, d_j$ are parameters for each edge of the system. Each input $u_j$ and each dynamic state $x_i$ has upper and lower bounds of the form $\underline{u}_j \le u_j \le \overline{u}_j$ and $\underline{x}_i \le x_i \le \overline{x}_i$. Bilinear power flow relationships are often found in power flow systems [38], [39], such as thermal systems where heat flow $Q \propto \dot{m}\left(T_1 - T_2\right)$ is proportional to a mass flow rate $\dot{m}$ and a temperature difference between a source temperature $T_1$ and a sink temperature $T_2$. The power flow for the entire system is represented as the vector

$$P = \left(u + a\right) \cdot \left( M_{b,c}^T \begin{bmatrix} x \\ x^t \end{bmatrix} + d \right) \tag{3.2}$$

where $u = \left[u_j\right]$, $a = \left[a_j\right]$, and $d = \left[d_j\right]$ for $e_j \in E$, $x \in \mathbb{R}^{N_v}$ are the states, $x^t \in \mathbb{R}^{N_t}$ are the sink values, and $M_{b,c} = \left[m_{i,j}\right] \in \mathbb{R}^{\left(N_v + N_t\right) \times N_e}$ is a weighted incidence matrix where

$$m_{i,j} = \begin{Bmatrix} b_j & \text{if } v_i \text{ is the tail of } e_j \\ c_j & \text{if } v_i \text{ is the head of } e_j \\ 0 & \text{else} \end{Bmatrix}. \tag{3.3}$$

Due to the bilinearity in (3.2), the system dynamics cannot be represented as a linear state space equation. However, to keep the control optimization problem formulated in Section 3.7 linear, a convex relaxation can be used where the vector $P$ serves as the decision variable, reducing the system to a system of integrators represented by (2.9), rewritten here as

$$C\dot{x} = -\overline{M}P + DP^{in}. \tag{3.4}$$

In the control formulation, (3.2) is incorporated as the set of linear constraints

$$\left(\underline{u}+a\right)\cdot\left(M_{b,c}^{T}\begin{bmatrix} x \\ x^{t} \end{bmatrix}+d\right)\le P\le\left(\bar{u}+a\right)\cdot\left(M_{b,c}^{T}\begin{bmatrix} x \\ x^{t} \end{bmatrix}+d\right), \tag{3.5}$$

where $\underline{u}=\left[\underline{u}_i\right]$, $\bar{u}=\left[\bar{u}_i\right]$ for $e_j\in\left[E\right]$, which ensures that the power flows determined by the controller can be realized with a set of inputs $\underline{u}\le u\le\bar{u}$. Thus the bilinear plant dynamics can be represented as integrator dynamics with linear constraints used by the controller, and the nonlinearity is captured by calculating the inputs to the system as

$$u_j=P_j\cdot\left(b_j x_j^{tail}+c_j x_j^{head}+d_j\right)^{-1}-a_j,\ \ \forall e_j\in E. \tag{3.6}$$

This plant representation is used to develop the centralized MPC controller used for comparison purposes in Section 3.8. To develop the hierarchical control framework, this centralized plant model must be partitioned to develop graphs used by each controller in the hierarchy.

## 3.5 System Decomposition

In order to develop a hierarchical controller, the power flow system, represented as an oriented graph $\mathcal{G}$, using the framework presented in Chapter 2, must be decomposed temporally and functionally. To demonstrate this decomposition, the example system shown in Fig. 3.4 is used throughout the remainder of this Chapter. The graph for this system has $N_v=12$ vertices, $N_e=18$ edges, $N_s=2$ source power flows, and $N_t=2$ sinks. Based on the capacitances of the vertices, vertices with slow, medium, and fast dynamics are indicated by different colors. The overall vehicle system is decomposed into 2 systems, each containing 2 subsystems. Based on the three timescales and the number of systems and subsystems, the corresponding three-level hierarchical controller is shown in Fig. 3.5.

In general, temporal partitioning is based on the timescale separation of the dynamics in the plant, represented by the magnitude of the vertex capacitances $C_i$, $i\in\left[1,N_v\right]$, as discussed in Section 2.4. For the three timescale example in this Chapter, it is convenient to refer to the dynamics as fast, medium, or slow. Note that when determining this temporal decomposition, it may be necessary to normalize the capacitances based on the magnitude of the corresponding state, especially when multiple energy domains are modeled. For example, the capacitance

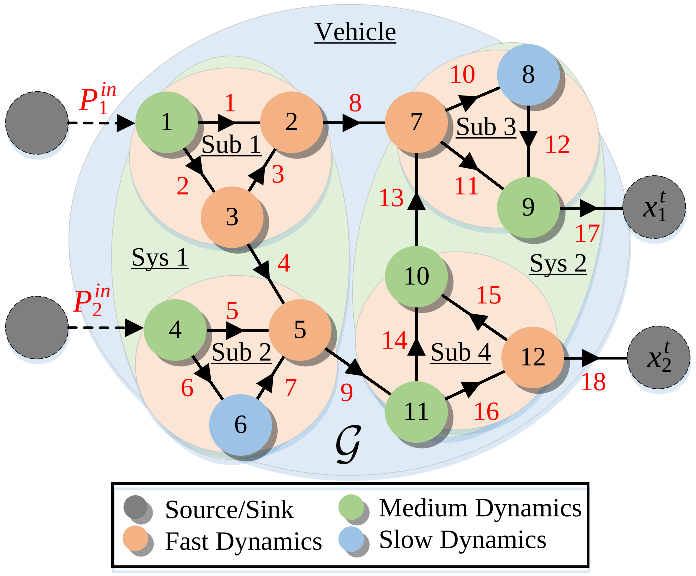


**Figure 3.4 Example system graph used to demonstrate the hierarchical control development and performance.**

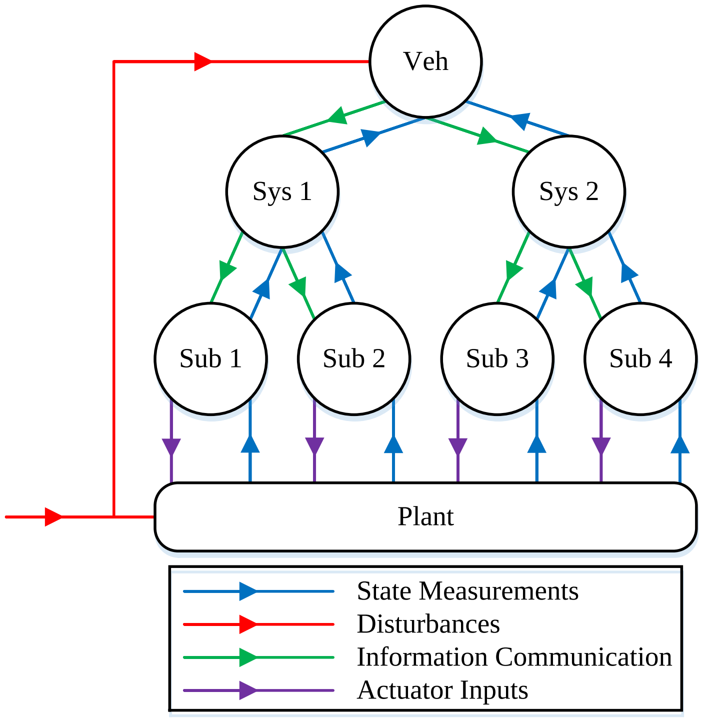


**Figure 3.5 Example three-level hierarchy, and corresponding information flow, with a single vehicle-level controller, two system-level controllers, and four subsystem-level controllers.**

representing the inertia of an engine with the state corresponding to the angular velocity should be normalized when compared to the capacitance representing the thermal capacitance of a heat exchanger with the state corresponding to average temperature since the engine speed might change by thousands of RPM compared to heat exchanger temperature which might only change tens of degrees.

The plant must also be partitioned spatially. Often this spatial partitioning is intuitive and can be derived from functionality, physical location, or energy domain. However, for plants where the partitioning is unclear, [40] presents several algorithms for optimal partitioning of systems for decentralized and distributed control. In general, for a control hierarchy with $N$ levels, the $i^{th}$ level of the hierarchy has $n_i$ controllers where $n_1 = 1$ and $n_i \leq n_{i+1}$. As discussed in Section 8.2, future work should establish graph-based temporal and spatial partitioning procedures specifically for hierarchical control where partitioning optimizes the tradeoff between control performance and computational cost. System partitioning in this dissertation is performed manually to demonstrate hierarchical controller development and is not optimized to maximize performance.

For the $N = 3$ level example system from Fig. 3.4, the system is decomposed into the $n_N = n_3 = 4$ subsystems $G_i^{sub}$ shown in Fig. 3.6. For these individual subsystem graphs, light gray vertices represent virtual sources and sinks where power flow is exchanged with a neighboring subsystem. Thus $G_i^{sub}$ includes all edges of $G$ which are incident to the dynamic vertices of the subsystem, but no more.

These subsystem graphs are used to create the models for the MPC controllers at the bottom level of the hierarchy in Fig. 3.5, following the procedure outlined in Section 3.7. The controllers at the upper levels of the hierarchy use a reduced system model, as discussed in the following Section, to minimize computational cost and improve the scalability of the control approach.

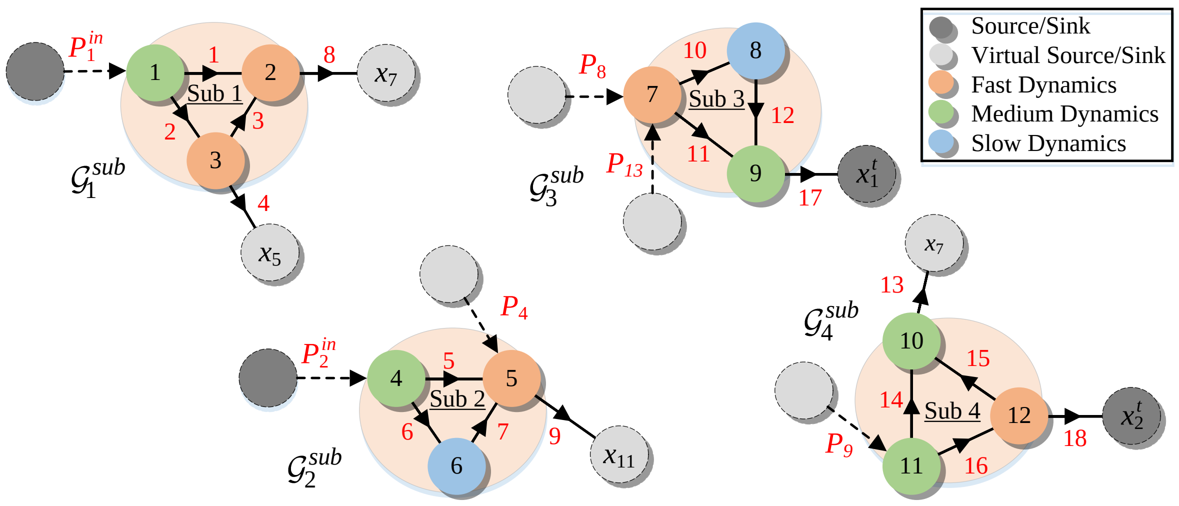


**Figure 3.6 Individual subsystem graph representations for the example system.**

## 3.6 Graph-based Model Reduction

For graph-based model reduction, the general idea is to use the $n_i$ subsystem reduced graphs at the $i^{th}$ level to generate the reduced graphs at the $i-1$ level. However, the $N^{th}$ level controllers do not use a reduced model as discussed in the previous Section, and thus this model reduction is only performed for levels $N-1$ through 1.

Let the $j^{th}$ subsystem at the $i-1$ level be an aggregation of neighboring subsystems at the $i^{th}$ level. For the example from Fig. 3.4, the graph for system 1, $G_1^{sys}$, is derived from reduced graphs for subsystems 1 and 2, while the graph for system 2, $G_2^{sys}$, is derived from reduced graphs for subsystems 3 and 4. For each subsystem, the graph condensation is a two-step process where 1) the fast dynamic vertices are converted into algebraic vertices and 2) neighboring algebraic vertices are combined into a single vertex. Algebraic vertices $v_i$ have no capacitance and thus must satisfy

$$0 = \sum_{e_j \in E_i^{in}} P_j - \sum_{e_k \in E_i^{out}} P_k. \tag{3.7}$$

Once condensed, the reduced graphs for subsystems 1 and 2 and the reduced graphs for subsystems 3 and 4 are combined to create the graphs for system 1 and 2, $G_1^{sys}$ and $G_2^{sys}$, respectively. It is important to note that if there are adjacent algebraic vertices in the resulting system graph, these vertices should not be combined, since the power flow between these vertices is critical to the coordination of the constitutive subsystems. As a result of this condensation, the system graphs only include vertices with slow and medium dynamics, as well as algebraic vertices.

Fig. 3.7 shows the resulting system graphs $G_1^{sys}$ and $G_2^{sys}$. The two fast dynamic vertices of $G_1^{sub}$ are converted into algebraic vertices (represented as white vertices) and condensed since $v_2$ and $v_3$ are neighbors through $e_3$. The fast dynamic vertex $v_5$ in $G_2^{sub}$ is also converted to an algebraic vertex. The resulting subsystem graphs were combined to produce $G_1^{sys}$. As mentioned previously, the algebraic vertex $v_{2,3}$ is not combined with algebraic vertex $v_5$ since the controller for system 1 must determine the desired power flow along $e_4$ in order to address the coupling between subsystems 1 and 2 based on the previous discussion using Fig. 3.3. The same procedure is followed to generate $G_2^{sys}$, where $v_7$ and $v_{12}$ are converted to algebraic vertices; however, no vertices are combined since there were no neighboring fast dynamic vertices in either subsystem 3 or 4.

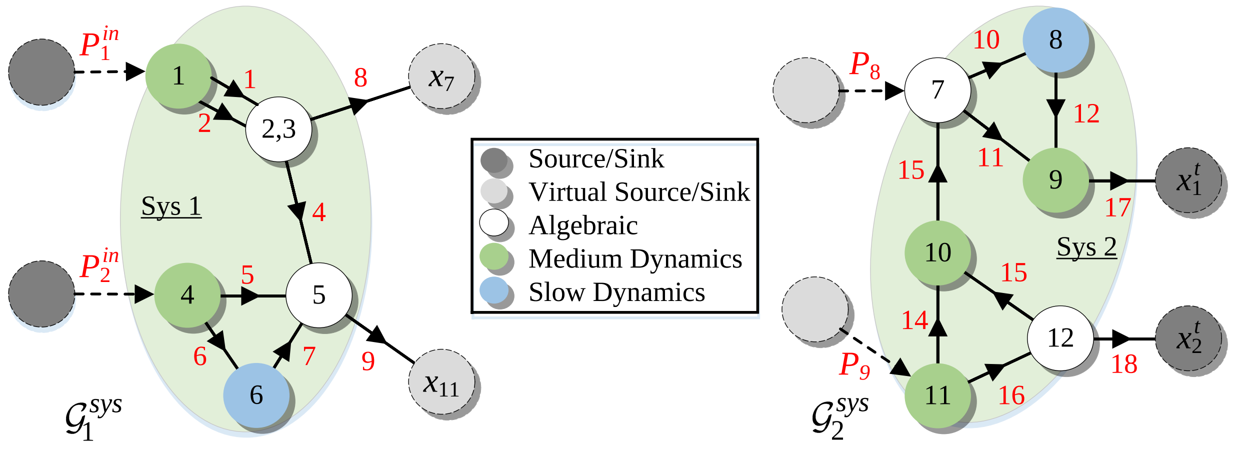


**Figure 3.7 System graph representations for the example system.**

This procedure of condensing is repeated to develop the vehicle graph $G^{veh}$ from the 2 system graphs $G_i^{sys}$. Now the medium dynamic vertices are converted into algebraic vertices and neighboring algebraic vertices are combined into a single vertex. Once condensed, the 2 system graphs are combined to create the vehicle graph. The vehicle graph only includes vertices with slow dynamics and algebraic vertices.

Fig. 3.8 shows the resulting graph $G^{veh}$. The medium dynamic vertices of $G_1^{sys}$ and $G_2^{sys}$ were converted to algebraic vertices and combined with neighboring algebraic vertices. Based on the structure of $G_1^{sys}$ and $G_2^{sys}$ all algebraic vertices were combined into a single algebraic vertex for each system. The two condensed system graphs were combined resulting in two algebraic vertices and the two slow dynamic vertices $v_6$ and $v_8$. Once again, the two algebraic vertices are not combined because the vehicle-level controller must determine the desired power flow along $e_8$ and $e_9$ to address the coupling between systems 1 and 2.

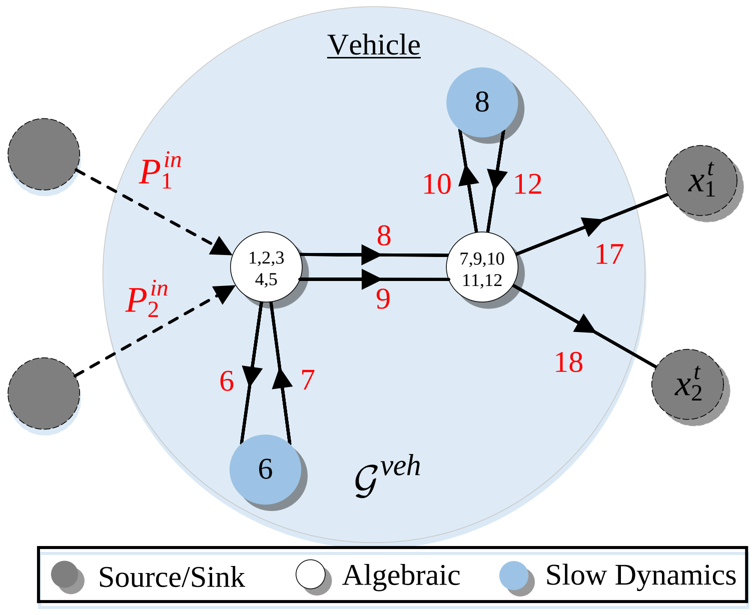


**Figure 3.8 Vehicle graph representation for the example system.**

The sets and parameters identified for the entire system in Chapter 2 can be defined for each graph $G^{veh}$, $G_i^{sys}$, and $G_i^{sub}$, where a superscript is used to denote the set or parameters for a particular graph (e.g. $x^z$ is used to denote the vector of dynamic states for graph $G^z$ in the controller development below, where for the example system $z \in \{veh, sys_1, sys_2, sub_1, sub_2, sub_3, sub_4\}$).

## 3.7 Controller Development

Fig. 3.5 shows the three-level hierarchical control architecture used to control the example system from Fig. 3.4. In general, the vehicle-level receives preview information for upcoming disturbances. For the example system this refers to the predicted values for $P_1^{in}$, $P_2^{in}$, $x_1^t$, and $x_2^t$. The vehicle-level controller then uses the dynamic representation of the system derived from $G^{veh}$ to determine the desired states for the slow dynamic vertices. For the example system this corresponds to $x_6$ and $x_8$. The vehicle-level also determines the desired power flows for any edge that connects two systems as well as the desired states for the tail and head vertices for these edges. For the example system, these are the power flows along $e_8$ and $e_9$ along with $x_2$, $x_7$, $x_5$, and $x_{11}$. The system-level controllers attempt to track these desired values. If a desired power flow is exiting the system, the system controller tries to track the desired power flow using knowledge of the desired value of the head vertex state sent from the vehicle-level controller. If a desired power flow is entering the system, this power flow is treated as a known disturbance to the system and the system controller tries to achieve the desired head vertex states sent from the vehicle-level controller. For the example system, the system 1 controller tries to achieve the desired power flows along $e_8$ and $e_9$ using knowledge of the desired values $x_7$ and $x_{11}$. The system 2 controller tries to achieve the desired values for $x_7$ and $x_{11}$ using knowledge of the desired power flow along $e_8$ and $e_9$. This approach helps to ensure consistency among the actions of the system-level controllers. The system-level also determines the desired values for the medium dynamic states, the desired power flow along edges connecting subsystems, and the tail and head vertex states for these edges. This process continues for the subsystem-level controllers.

With the information communication architecture and desired control behavior defined, MPC controllers are used at each level of the hierarchy to achieve this coordination between systems and subsystems. One of the key features of the proposed hierarchical controller in this Chapter is that each MPC controller in the hierarchy has the same general form. This significantly simplifies the control design procedure and allows the hierarchical approach to be readily scaled to larger systems with more levels of control. Controller $z$ solves the constrained quadratic program

$$\begin{aligned}
&\min_{\boldsymbol{P}^z} \sum_{k=0}^{N_p-1} \sum_{i=1}^{6} \lambda_i J_i(k) \\
&s.t. \quad g_i(k) \le 0 \quad \forall i \in [1,5], k \in \left[1, N_p - 1\right], \\
&\qquad h_i(k) = 0 \quad \forall i \in [1,4], k \in \left[1, N_p - 1\right], \\
&\qquad x^z(0) = x_0^z,
\end{aligned} \tag{3.8}$$

where

$$\begin{aligned}
&J_1(k) = \left\| x^z(k) - x_{des}^z(k) \right\|_2^2, \qquad J_2(k) = \left\| P^z(k) \right\|_2^2, \\
&J_3(k) = \left\| x^z(k) - x_{track}^z(k) \right\|_2^2, \quad J_4(k) = \left\| P^z(k) - P_{track}^z(k) \right\|_2^2, \\
&J_5(k) = \left\| P^z(k) - P^z(k-1) \right\|_2^2, \quad J_6(k) = \left\| s^z(k) \right\|_2^2,
\end{aligned} \tag{3.9}$$

and

$$\begin{aligned}
&g_1(k) = \left( \underline{u}^z + a^z \right) \cdot \left( \left( M_{b,c}^z \right)^T \begin{bmatrix} x^z(k) \\ x^{t,z}(k) \end{bmatrix} + d^z \right) - P^z(k), \\
&g_2(k) = P^z(k) - \left( \overline{u}^z + a^z \right) \cdot \left( \left( M_{b,c}^z \right)^T \begin{bmatrix} x^z(k) \\ x^{t,z}(k) \end{bmatrix} + d^z \right), \\
&g_3(k) = \underline{x}^z - x^z(k) - s^z(k), \qquad g_4(k) = x^z(k) - \overline{x}^z - s^z(k), \\
&g_5(k) = -s^z(k), \\
&h_1(k) = C^z \left( x^z(k+1) - x^z(k) \right) - \Delta t^z M^z P^z(k) + \Delta t^z D^z P^{in,z}(k), \\
&h_2(k) = F^z P^z(k) \qquad h_3(k) = P^{in,z}(k) - P_{preview}^{in,z}(k), \\
&h_4(k) = x^{t,z}(k) - x_{preview}^{t,z}(k).
\end{aligned} \tag{3.10}$$

The lifted vector $\boldsymbol{P}^z = \left[ P^z(k) \right]$ contains the decision variables corresponding to the power flows along the edges of the graph at every time step $k$ over the prediction horizon $N_p$. The objective function includes six cost terms each defined at every time step of the prediction horizon as well as five inequality constraints and four equality constraints. The initial state vector $x^z(0)$ equals the measured state of the system $x_0^z$. The six terms of the cost function, weighted by $\lambda_i$, penalize

1) tracking of desired stated values,
2) power flow along edges,
3) tracking of desired state values sent from the controller directly above in the hierarchy,
4) tracking of desired power flows also sent from the controller directly above in the hierarchy,
5) changes in power flows in time, and
6) the slack variables used to ensure feasibility of the optimization problem when state constraint violation is unavoidable.

The five inequality constraints bound

1) the power flows from below based on the minimum achievable inputs,
2) the power flows from above based on the maximum achievable inputs,
3) the states from below,
4) the states from above, and
5) the slack variables to be positive.

Finally, the two equality constraints provide

1) the discretized system dynamics with a time step of $\Delta t^z$ based on the corresponding graph for the particular controller,
2) the algebraic relationships between power flows in the system,
3) preview for the source disturbances provided by the controller directly above in the hierarchy, and
4) preview of the sink disturbances also provided by the controller directly above in the hierarchy.

While each controller has the same generic form, the exact controller formulation depends on the level of the controller in the hierarchy. Since there is no controller above the vehicle-level, the vehicle-level controller does not have $J_3$ or $J_4$ and does not receive the disturbance preview information in $h_3$ and $h_4$ from a controller. If preview information is available, this information must be provided directly to the vehicle-level controller as shown in Fig. 3.5. In addition to the MPC controller, the controllers at the lowest level in the hierarchy, the subsystem controllers in this Chapter, must also calculate the control inputs to be sent directly to the system. Eq. (3.6) is used to calculate the input signals based on the desired power flows at the first time step $P^z(1)$ and the measured states $x_0^z$.

## 3.8 Numerical Example

The following simulation results compare the control performance achieved by centralized, decentralized, and hierarchical control approaches. The example system from Fig. 3.4 is used with the parameters listed at the end of the Chapter. All controllers have a prediction horizon of $N_p = 20$ steps. The centralized controller is evaluated using three different update rates of $\Delta t = 1$, 10, and 100 seconds corresponding to the update rates of the subsystem, system, and vehicle-level controllers in the hierarchical controller, $\Delta t^{sub} = 1$, $\Delta t^{sys} = 10$, and $\Delta t^{veh} = 100$, respectively. The decentralized controller has an update rate of $\Delta t = 1$ second and consists of the four subsystem-level controllers at the lowest level of the hierarchy without the system- and vehicle-level controllers. For brevity, the following figures only show the centralized control performance corresponding to an update rate of 10 seconds. Also, for clarity, only some of the vertex state trajectories are highlighted in Figs. 3.9 and 3.11 as indicated in the legends.

First, the convergence properties of each control approach are evaluated. Fig. 3.9 shows the convergence performance of the centralized, decentralized, and hierarchical controllers where each vertex in the example system from Fig. 3.4 has an initial state of either 25 or 75 and all states are to be regulated at 50. As expected, the centralized controller, Fig. 3.9a, provides the fastest convergence utilizing complete knowledge of the coupling in the plant. The convergence results for the centralized controllers with update rates of 1 and 100 seconds are very similar to that shown in Fig. 3.9a. The decentralized controller, with no knowledge of the coupling between subsystems, converges much more slowly as shown in Fig. 3.9b. The hierarchical controller, Fig. 3.9c, while not as effective as the centralized approach, performs significantly better than the decentralized controller, demonstrating how the system and vehicle-level controllers effectively coordinate the actions of the subsystems. The spikes in $x_5$ and $x_7$ (cyan and black lines in Fig. 3.9c) are due to an update in the desired power flow determined by the vehicle-level controller. These states are affected most significantly because they are in the set of fast dynamic vertices and are directly affected by the power flow out of other subsystems.

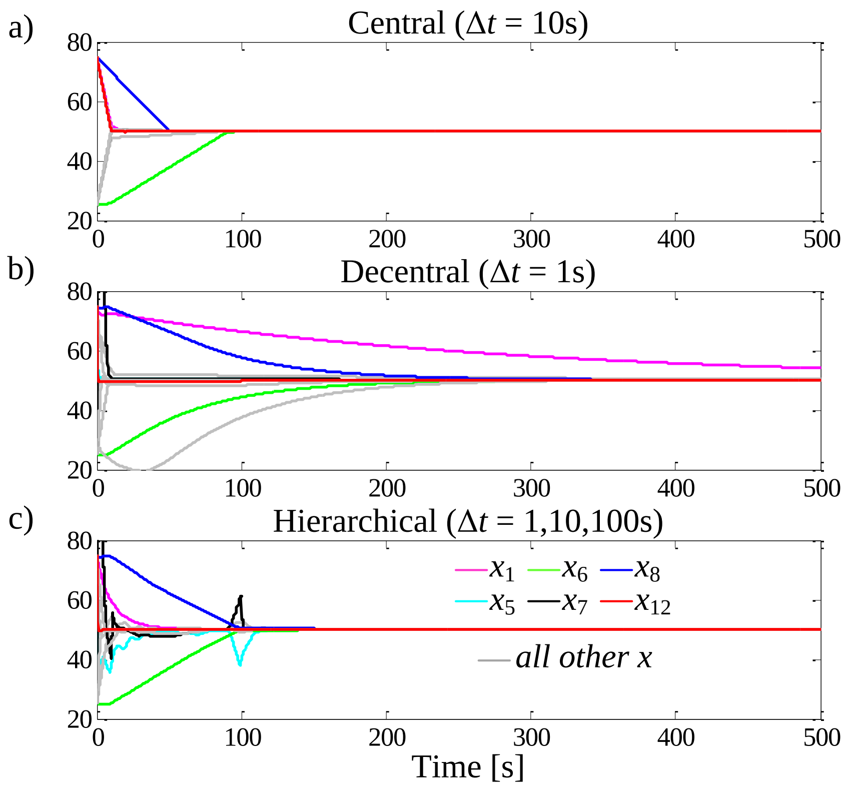


**Figure 3.9 Convergence results for centralized, decentralized, and hierarchical controllers.**

Next, the controller performance is evaluated for a scenario with time-varying source and sink disturbances. Fig. 3.10 shows the disturbance profiles for $P_1^{in}$, $P_2^{in}$, $x_1^t$, and $x_2^t$. Based on the power flow relationship for $e_{17}$, the change in $x_1^t$ results in a change in the maximum power flow to this sink from 50W to 75W. This disturbance is followed by an increase in inlet power $P_1^{in}$ from 50W to 75W. The inlet power flow $P_2^{in}$ and the sink state $x_2^t$ are constant at 50. These disturbances are previewed by the centralized controller and the vehicle-level controller of the hierarchy. Each source and sink also has small unmeasured high-frequency deviations that are not included in the preview information. Fig. 3.11 shows the control results for the three different

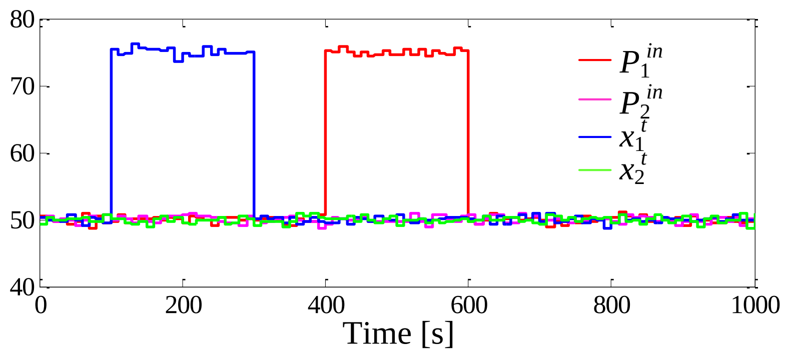


**Figure 3.10 Disturbance profiles.**

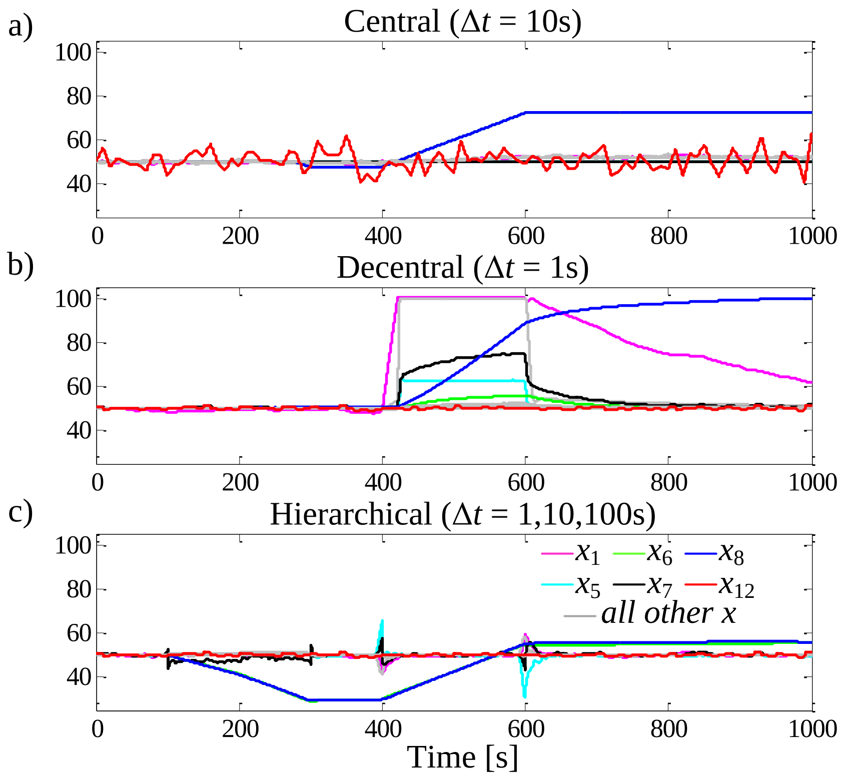


**Figure 3.11 Disturbance rejection results for centralized, decentralized, and hierarchical controllers.**

control approaches. Starting with the hierarchical control approach in Fig. 3.11c, the controller uses the long prediction horizon of the vehicle-level controller to "precool" the slow dynamic states ($x_6$ and $x_8$) from 100 to 300 seconds (shown by the blue and green lines, green is covered by blue). When the large power load enters the system from 400 to 600 seconds, these states increase and settle near the desired value of 50. Also, note that $x_{12}$ (red line) is well regulated to the desired value of 50. The centralized controller with an update rate of $\Delta t = 10$, Fig. 3.11a, does not have as long of a prediction horizon as the vehicle-level controller and thus does not effectively use the extra sink capacity during 100 to 300 seconds resulting in minimal precooling of $x_6$ and $x_8$. The large power load then causes these states to rise well above the desired value of 50. Additionally, since the centralized controller has a larger update rate than the subsystem-level controllers, the centralized controller is not able to reject the unmeasured disturbances as effectively, resulting in the large variations in $x_{12}$ (red line in Fig. 3.11a). Finally, as shown in Fig. 3.11b, the decentralized controller, while able to reject the unmeasured disturbances due to its fast update rate, is unable to precool $x_6$ and $x_8$ and is also unable to regulate $x_1$, $x_5$, and $x_7$ (the magenta, cyan, and black lines in Fig. 3.11b) due to its lack of disturbance preview and subsystem coupling knowledge. The relative tracking performance for each controller is compared in Fig. 3.12, where the height of each bar $h_i$ is calculated as

$$h_i = \sum_{k=0}^{1000} \left\| x(k) - 50 \right\|_2^2, \tag{3.11}$$

and then normalized compared to the hierarchical control approach. Thus a height of zero on this plot corresponds to perfect tracking. In Fig. 3.12, each bar is subdivided corresponding to the tracking error for $x_6$ and $x_8$ (blue area), $x_{12}$ (green area), and the states of the remaining vertices (red area). In general, the centralized controllers perform significantly better than the decentralized approach but have about twice the tracking error as the hierarchical approach. The majority of the tracking error for the centralized controller with an update rate of $\Delta t = 1$ second comes from the inability to precool $x_6$ and $x_8$ (blue area), while the majority of the tracking error for the centralized controller with an update rate of $\Delta t = 100$ seconds comes from the inability to regulate $x_{12}$ due to the unmeasured high-frequency disturbances (green area). The majority of the tracking error for the hierarchical control approach comes from the strategic precooling of $x_6$ and $x_8$. Due to the actuator constraints of the system and the magnitude of the disturbances, zero tracking error is infeasible for this scenario. Thus the hierarchical controller determines the appropriate precooling to minimize the total tracking error before and after the large power load.

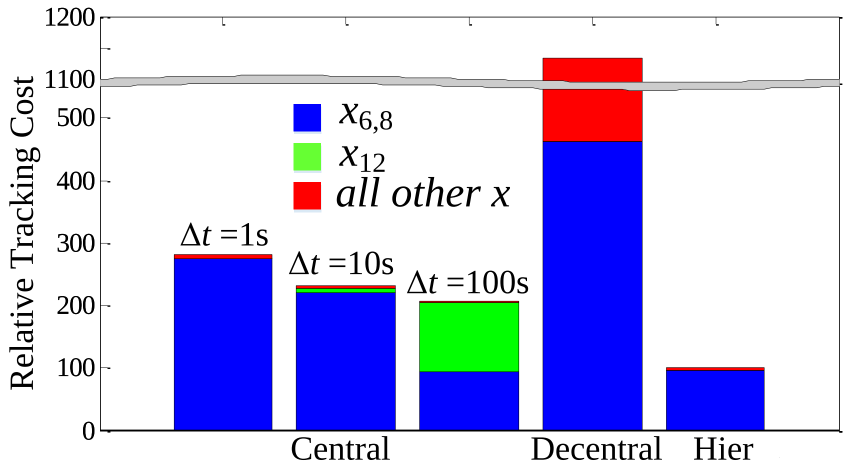


**Figure 3.12 Comparison of reference tracking error for centralized, decentralized, and hierarchical controllers.**

The hierarchical controller achieves this performance increase in a computationally efficient way. Fig. 3.13 shows the time required to solve the optimization problem at each time step throughout the simulation for the centralized controller with $\Delta t = 10$ and each of the individual controllers within the hierarchical controller. The MPC optimization problems are formulated using the YALMIP Toolbox [41] and solved using the Gurobi optimization suite [42] run on a desktop computer with a 3.10 GHz Intel Xeon E31225 processor and 8 GB of RAM. By decomposing the control decisions for the system among coordinated controllers, the hierarchy reduces the computational cost of each controller by about 50% of the centralized controller. However, since there are more individual controllers in the hierarchy, the overall computational cost is higher. This would not necessarily be true for a larger system with more states and decision variables. Additionally, the hierarchical controller can take advantage of parallel processing while the centralized controller cannot. Future work should investigate how the increase in computational cost of hierarchical controllers compares that that of centralized controllers.

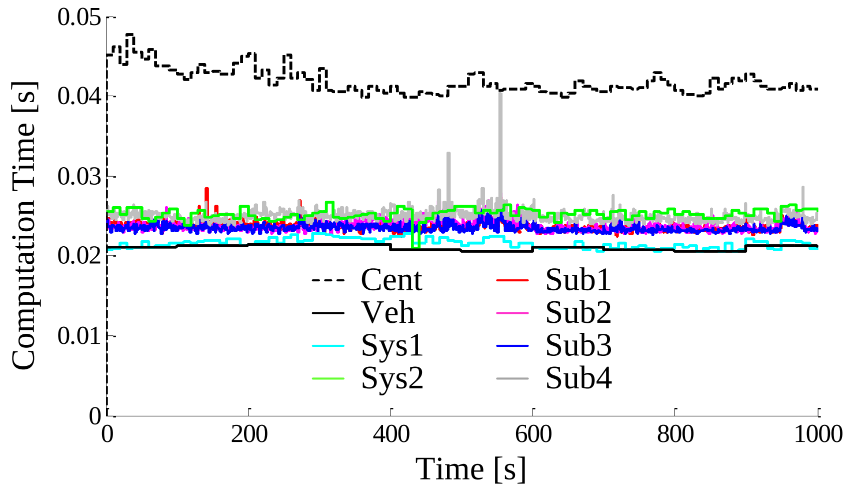


**Figure 3.13 Comparison of controller computation times for the centralized controller with $\Delta t = 10$ and the hierarchical controllers.**

## 3.9 Chapter Summary

A hierarchical control approach is well suited for controlling the complex multi-timescale power flow systems in vehicles. Vehicle-wide control performance is achieved through coordination among systems and subsystems at each timescale. This Chapter presented a generic hierarchical control development procedure and demonstrated the efficacy of the approach on a simulated example system. The following two Chapters further the development of hierarchical control by analyzing the theoretical properties of stability and robust feasibility.

*Example System Parameters*

The following table contains the parameters for the example system in Fig. 3.4.

| $x_i$ | $C_i$ | $e_i$ | $a_i$ | $b_i$ | $c_i$ | $d_i$ | $\underline{u}_i$ | $\overline{u}_i$ |
|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| 2 | 10 | 2 | 0 | 0 | 0 | 1 | 0 | 50 |
| 3 | 1 | 3 | 0 | 0 | 0 | 1 | 0 | 50 |
| 4 | 1 | 4 | 0 | 0 | 0 | 1 | 0 | 50 |
| 5 | 0 | 5 | 0 | 0 | 0 | 1 | 0 | 50 |
| 6 | 10 | 6 | 1 | 1 | 0 | 0 | 0 | 0 |
| 7 | 1 | 7 | 0 | 0 | 0 | 1 | 0 | 50 |
| 8 | 100 | 8 | 0 | 0 | 0 | 1 | 0 | 50 |
| 9 | 1 | 9 | 0 | 0 | 0 | 1 | 0 | 50 |
| 10 | 100 | 10 | 0 | 0 | 0 | 1 | 0 | 100 |
| 11 | 10 | 11 | 0 | 0 | 0 | 1 | 0 | 50 |
| 12 | 0 | 12 | 0 | 0 | 0 | 1 | 0 | 50 |
| 13 | 10 | 13 | 0 | 0 | 0 | 1 | 0 | 50 |
| 14 | 10 | 14 | 0 | 0 | 1 | 0 | 0 | 1 |
| 15 | 1 | 15 | 0 | 0 | 0 | 1 | 0 | 50 |
| 16 | 0 | 16 | 0 | 0 | 0 | 1 | 0 | 100 |
|  |  | 17 | 0 | 0 | 0 | 1 | 0 | 50 |
|  |  | 18 | 0 | 0 | 0 | 1 | 0 | 50 |
|  |  | 19 | 0 | 0 | 0 | 1 | 0 | 50 |
|  |  | 20 | 0 | 0 | 1 | 0 | 0 | 1 |

$\underline{x}_i = 0,\ \overline{x}_i = 100 \quad \forall i \in [1,16]$

# Chapter 4
# Passivity-based Stability

## 4.1 Motivation

For energy management onboard vehicle systems, controllers are tasks with maximizing performance through optimizing the generation, storage, distribution, and utilization of energy. As these controllers are designed more aggressively, guaranteeing stability of the closed-loop system becomes vital to safe and reliable operation. However, assessing closed-loop stability of a control hierarchy remains difficult due to the complex structure of interaction among individual controllers in the hierarchy. Despite the performance advantages of MPC, the general lack of closed-form control solution compounds this difficulty. Through analysis of the specific structure of graph-based power flow systems, this Chapter provides a formulation for augmenting each MPC controller at the lowest level of a hierarchical controller, using a local passivity constraint, which guarantees closed-loop stability for the overall system.

## 4.2 Background

When modeling power flow systems as a graph, the governing energy conservation laws suggest an inherent feature of these systems: *passivity*. The notion of passivity in system modeling and control originated from the physical principles of energy conservation and dissipation in electrical and mechanical systems [43] and has become a widely used and highly general methodology in nonlinear system analysis and control [44]–[46]. Thus, passivity-based control has been applied to a variety of power flow systems in centralized [30], [47], [48] and decentralized control architectures [49].

Due to the numerous benefits of MPC, many centralized, passivity-based MPC formulations have been developed in the literature [50]–[54]. However, due to the complexity of many systems, including power flow systems, a centralized control approach may be computationally impractical and may not provide sufficient robustness to faults in the system. Thus, several distributed passivity-based MPC formulations have also been established [18], [55]. In these approaches, along with the system analysis in [56], [57], stability is assessed with a global, system-wide matrix condition that accounts for the subsystem interconnection topology and the gain of the coupling between subsystems. While this may be practical for some systems, the need to analyze global properties of the system is limiting and, as will be shown, unnecessary for guaranteeing closed-loop stability of graph-based power flow systems.

The aim of this Chapter is to present a purely decentralized and easily implementable method for augmenting existing decentralized and hierarchical control frameworks that guarantees stability of the overall closed-loop system. The relative simplicity of the approach is enabled by exploiting the structure of power flow systems represented as graphs. The proposed approach identifies a set of inputs and outputs that render each subsystem passive. Neighboring subsystems form a negative feedback connection, establishing passivity of the overall system. While the approach relies on a graph-based representation of the system, a nonlinear, affine in control, power flow representation provides applicability to a wide class of systems. Actuator input and state constraints are considered, with slack variables on the state constraints to avoid infeasibility issues. Through the addition of a nonlinear constraint to each controller, the proposed approach provides simple implementation and reduced conservatism compared to standard passivity-based approaches.

## 4.3 Nonlinear Graph System Model

Following the general graph-modeling framework from Chapter 2, consider a power flow system composed of $N_{sub}$ interconnected subsystems $\mathbf{S}_i$, $i \in [1, N_{sub}]$. Each subsystem is represented by an oriented graph $\mathcal{G}_i = (V_i, E_i)$ of order $N_{v,i}$ with set of vertices $V_i = \{v_{i,k}\}$, $k \in [1, N_{v,i}]$ and of size $N_{e,i}$ with set of edges $E_i = \{e_{i,j}\}$, $j \in [1, N_{e,i}]$. Each oriented edge $e_{i,j} \in E_i$ represents a path for power flow in $\mathbf{S}_i$, where positive power $P_{i,j}$ flows from the tail vertex $v_{i,j}^{tail}$ to the head vertex $v_{i,j}^{head}$. Each vertex $v_{i,k} \in V_i$ has an associated state $x_{i,k}$ that represents the amount of energy stored in that vertex. Thus the dynamic for the state of each $v_{i,k}$ satisfies the energy conservation equation

$$C_{i,k}\dot{x}_{i,k} = \sum_{e_{i,j} \in E_{i,k}^{in}} P_{i,j} - \sum_{e_{i,j} \in E_{i,k}^{out}} P_{i,j}, \tag{4.1}$$

where $C_{i,k} > 0$ is the energy storage capacitance of vertex $v_{i,k}$ and $E_{i,k}^{in} = \left\{ e_{i,j} \mid v_{i,j}^{head} = v_{i,k} \right\}$ and $E_{i,k}^{out} = \left\{ e_{i,j} \mid v_{i,j}^{tail} = v_{i,k} \right\}$ are the sets of edges oriented into and out of the vertex $v_{i,k}$.

**Assumption 4.1**

*The power flow $P_{i,j}$ along edge $e_{i,j}$ is defined as*

$$P_{i,j} = f_{i,j}\left(x_{i,j}^{tail}, x_{i,j}^{head}\right) + g_{i,j}\left(x_{i,j}^{tail}, x_{i,j}^{head}\right)u_{i,j}, \tag{4.2}$$

*where $x_{i,j}^{tail}$ and $x_{i,j}^{head}$ are the states of the tail and head vertices $v_{i,j}^{tail}$ and $v_{i,j}^{head}$, $u_{i,j}$ is an associated actuator input, and $f_{i,j}, g_{i,j} : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$. Additionally, $f_{i,j}$ is Lipschitz, twice continuously differentiable, and $f_{i,j}(0,0) = 0$ while $g_{i,j}$ is continuous, $g_{i,j}(0,0) = 0$, and the intersection of the zero sets of $g_{i,j}$ is the origin $\bigcap_j \mathcal{N}\left(g_{i,j}\left(x_{i,j}^{tail}, x_{i,j}^{head}\right)\right) = \{0\}$.*

Fig. 4.1 shows a graph of an example subsystem $\mathbf{S}_i$ used to identify key components. For this example subsystem, there are three paths for power to enter or exit the subsystem. For the two dashed edges oriented into the subsystem, the power flow along these edges, denoted $P_{i,1}^{in}$ and $P_{i,2}^{in}$, is treated as a disturbance to the subsystem and these edges are not included in $\mathcal{G}_i$. The third path is represented by an edge oriented out of the subsystem, labeled $P_{i,1}^{out}$. Power flow along this type of edge follows the relationship from (4.2), where now $x_{i,j}^{head}$ is a sink vertex state $x_{i,1}^{t}$. These sink states are *not* states of $\mathbf{S}_i$ and thus are disturbances to the subsystem, representing the surrounding environment. Finally, as indicated in Fig. 4.1, each subsystem has a subset $x_i^{in}$ of the states $x_i$ that represent the states directly affected by the inlet power flows $P_i^{in}$.

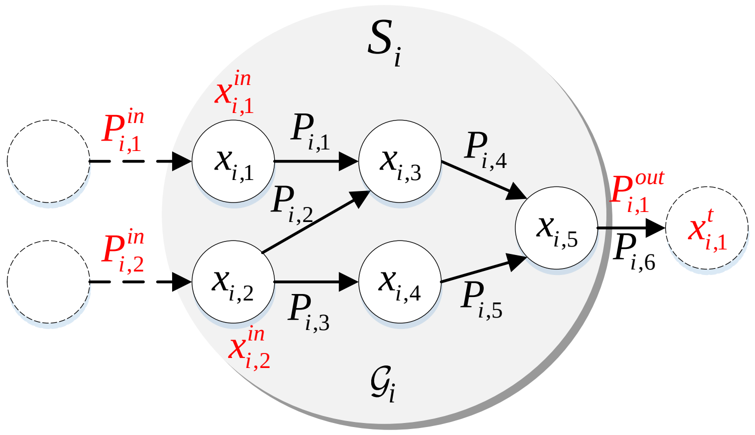


**Figure 4.1 Notional subsystem exemplifying the graph-based power flow representation with key power flows and states highlighted in red. Dashed lines indicate elements that serve as disturbances to the subsystem.**

Let $M_i = \left[ m_{i,jk} \right]$ be the incidence matrix of graph $\mathcal{G}_i$ [36] where

$$m_{i,jk} = \begin{cases} +1 & \text{if } v_{i,j} \text{ is the tail of } e_{i,k} \\ -1 & \text{if } v_{i,j} \text{ is the head of } e_{i,k} \\ 0 & \text{else} \end{cases}. \tag{4.3}$$

Then, based on (4.1), the subsystem dynamics are

$$\begin{bmatrix} C_i \dot{x}_i \\ \dot{x}_i^t \end{bmatrix} = -M_i P_i + \begin{bmatrix} D_i \\ 0 \end{bmatrix} P_i^{in}, \tag{4.4}$$

where $x_i$ are the states of the dynamic vertices, $x_i^t$ are the states of the sink vertices, $C_i = diag\left(\left[C_{i,k}\right]\right)$ is a diagonal matrix of the capacitances of the dynamic vertices, $P_i$ are the power flows along the edges of $\mathcal{G}_i$, $P_i^{in}$ are the source power flows entering $\mathbf{S}_i$, and $D_i = \left[ d_{i,jk} \right]$ is a matrix where

$$d_{i,jk} = \begin{Bmatrix} 1 & \text{if } v_{i,j} \text{ is the head of } P_{i,k}^{in} \\ 0 & \text{else} \end{Bmatrix}. \tag{4.5}$$

Since $x_i^t$ are disturbances to the system, not states, $M_i$ is partitioned as $M_i = \begin{bmatrix} \bar{M}_i \\ \underline{M}_i \end{bmatrix}$, with $\bar{M}_i \in \mathbb{R}^{N_{v,i} \times N_{e,i}}$ and $\underline{M}_i \in \mathbb{R}^{N_{t,i} \times N_{e,i}}$, resulting in

$$C_i \dot{x}_i = -\bar{M}_i P_i + D_i P_i^{in}. \tag{4.6}$$

From (4.2), the vector of power flows in $\mathbf{S}_i$ is

$$P_i = F_i\left(x_i, x_i^t\right) + G_i\left(x_i, x_i^t\right) u_i, \tag{4.7}$$

where $F_i\left(x_i, x_i^t\right) = \left[ f_{i,j}\left(x_{i,j}^{tail}, x_{i,j}^{head}\right) \right]$ and $G_i\left(x_i, x_i^t\right) = diag\left(\left[ g_{i,j}\left(x_{i,j}^{tail}, x_{i,j}^{head}\right) \right]\right)$. Thus the dynamics for $\mathbf{S}_i$ are

$$C_i \dot{x}_i = -\bar{M}_i F_i\left(x_i, x_i^t\right) - \bar{M}_i G_i\left(x_i, x_i^t\right) u_i + D_i P_i^{in}. \tag{4.8}$$

**Assumption 4.2**

*Each subsystem of the form*

$$\begin{bmatrix} C_i & 0 \\ 0 & I \end{bmatrix} \dot{\bar{x}}_i = -M_i F_i\left(\bar{x}_i\right), \quad \bar{x}_i = \begin{bmatrix} x_i \\ x_i^t \end{bmatrix}, \tag{4.9}$$

*admits an equilibrium $\bar{x}_i^*$ for a set of nominal inputs $u_i^*$ and disturbances $P_i^{in,*}$, $x_i^{t,*}$ and such an equilibrium is locally stable in the sense that the Jacobian matrix*

$$A_i = -\left.\frac{\partial M_i F_i\left(\bar{x}_i\right)}{\partial \bar{x}_i}\right|_{\bar{x}_i = \bar{x}_i^*}, \tag{4.10}$$

*has eigenvalues such that* $\operatorname{Re} \lambda_k\left(A_i\right) \le 0 \; \forall k$.

**Remark 4.1**

*It is assumed that* $\bar{x}_i^*$, $u_i^*$, $P_i^{in,*}$, $x_i^{t,*}=0$. *If otherwise, the subsystem states, inputs, and disturbances can be shifted such that* $\bar{x}_i^*$, $u_i^*$, $P_i^{in,*}$, $x_i^{t,*}=0$. *Note that if* $\mathrm{Re}\,\lambda_k\left(A_i\right)=0$ *for some* $k$, *linearization fails to assess the stability of the system and a center manifold analysis may be employed* [44].

**Remark 4.2**

*From the notation introduced in* [33], *if each* $f_{i,j}$ *is restricted to be a* g-type *flow, with* $f_{i,j}\left(x_{i,j}^{tail},x_{i,j}^{head}\right)=f_{i,j}\left(x_{i,j}^{tail}-x_{i,j}^{head}\right)$, *or a* h-type *flow, with* $f_{i,j}\left(x_{i,j}^{tail},x_{i,j}^{head}\right)=f_{i,j}\left(x_{i,j}^{tail}\right)$, *and is smooth with positive derivative, stabilizability of the open-loop subsystem can be assessed based on the external connectivity of* $\mathcal{G}_i$.

The overall power flow system $\mathbf{S}$, with graph $\mathcal{G}$, is composed of $N$ interconnected subsystems $\mathbf{S}_i$, $i\in[1,N]$. Following the same procedure used to define the dynamics of each subsystem, the overall system dynamics are

$$C\dot{x}=-\bar{M}F\left(x,x^t\right)-\bar{M}G\left(x,x^t\right)u+DP^{in}, \tag{4.11}$$

where $x=[x_i]$ and $u=[u_i]$ are the states and inputs of the entire system. The inlet power flows $P^{in}\subset\left\{P_1^{in},...,P_N^{in}\right\}$ are the power flows into system $\mathbf{S}$, i.e. power flows that do not come from neighboring subsystems. These power flows directly affect the states $x^{in}\subset\left\{x_1^{in},...,x_N^{in}\right\}$. The sink states for the system $x^t\subset\left\{x_1^t,...,x_N^t\right\}$ are the sink states from the individual subsystems that do not correspond to states of a neighboring subsystem. The power flows to these sink states are denoted $P^{out}\subset\left\{P_1^{out},...,P_N^{out}\right\}$.

**Definition 4.1**

*A* $u,v$-path *on* $\mathcal{G}=(V,E)$ *is a sequence of edges (regardless of orientation) connecting two distinct vertices* $u,v\in V$*, not including any sink vertices as intermediate vertices. A graph* $\mathcal{G}$ *is* connected *if it has a* $u,v$-path *for each* $u,v\in V$.

**Assumption 4.3**

*The graph* $\mathcal{G}$ *is connected.*

If $\mathcal{G}$ is not connected, the individual components of $\mathcal{G}$ are to be analyzed independently.

**Remark 4.3**

*While the dynamics of the overall system are defined in* (4.11)*, one of the key advantages of the proposed approach is that all assumptions and analysis are local to each subsystem. Development of the controller and assessing the closed-loop stability does not require analysis of the entire system. This can be advantageous when the size of the system prevents any type of centralized design or analysis and when analyzing plug-and-play systems* [58]*, where subsystems may go on- and offline during operation.*

## 4.4 Main Results

### 4.4.1 Passivity of Subsystems

**Definition 4.2** [44]

*The system* $H$ *with* $\dot{x}=f(x,u)$, $y=h(x,u)$ *where* $f:\mathbb{R}^n\times\mathbb{R}^p\to\mathbb{R}^n$ *is locally Lipschitz,* $h:\mathbb{R}^n\times\mathbb{R}^p\to\mathbb{R}^p$ *is continuous,* $f(0,0)=0$*, and* $h(0,0)=0$ *is* passive *if there exists a continuously differentiable positive semidefinite function* $V(x)$ *such that*

$$u^T y \geq \dot{V} = \frac{\partial V}{\partial x} f(x,u), \quad (x,u)\in\mathbb{R}^n\times\mathbb{R}^p. \tag{4.12}$$

*If* $u^T y \geq \dot{V}$ *for only a neighborhood of the origin,* $H$ *is* locally passive.

Fig. 4.2 shows the interconnection of subsystem $\mathbf{S}_i$ with "upstream" and "downstream" subsystems $\mathbf{S}_{i-1}$ and $\mathbf{S}_{i+1}$. The set of $N_i^s$ power flows into $\mathbf{S}_i$ from $\mathbf{S}_{i-1}$ is denoted $P_i^{in} \in \mathbb{R}^{N_i^s}$ and directly affects a corresponding set of states in $\mathbf{S}_i$, denoted as $x_i^{in} \in \mathbb{R}^{N_i^s}$. The set of $N_i^t$ power flows out of $\mathbf{S}_i$ into $\mathbf{S}_{i+1}$ is denoted $P_i^{out} \in \mathbb{R}^{N_i^t}$ and directly affects a corresponding set of states in $\mathbf{S}_{i+1}$, denoted as $x_i^t \in \mathbb{R}^{N_i^t}$. Note that, for the particular subsystem interactions shown in Fig. 4.2, $P_i^{out} = P_{i+1}^{in}$ and $x_i^t = x_{i+1}^t$. The interconnection between these three subsystems is shown as a set of negative feedback connections in Fig. 4.3. For subsystem $\mathbf{S}_i$, the actuator inputs and passivity outputs are denoted $u_i$ and $y_i$, respectively, where $y_i$ is a function of subsystem states $x_i$ and neighboring states $x_i^t$ and is strategically chosen below.

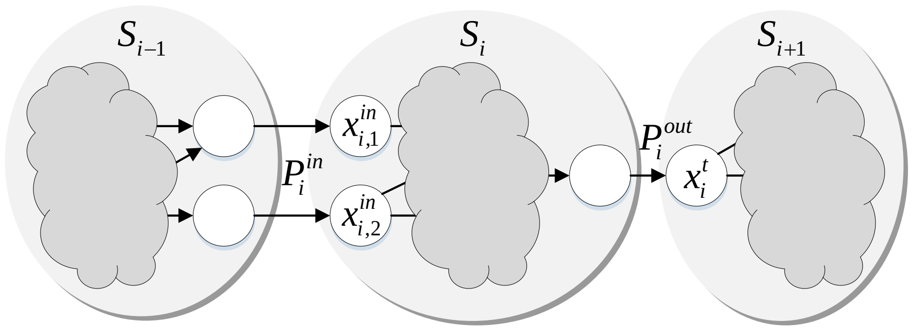


**Figure 4.2 Notional interconnection between three subsystems demonstrating the key interactions and relevant variables.**

**Theorem 4.1**

*A subsystem* $\mathbf{S}_i$, *represented by* (4.8), *is locally passive from* $\bar{u}_i$ *to* $\bar{y}_i$ *with*

$$\bar{u}_i = \begin{bmatrix} P_i^{in} \\ u_i \\ -x_i^t \end{bmatrix}, \quad \bar{y}_i = \begin{bmatrix} x_i^{in} \\ y_i \\ P_i^{out} \end{bmatrix}, \tag{4.13}$$

*and*

$$y_i = -G_i(\bar{x}_i) M_i^T \bar{x}_i. \tag{4.14}$$

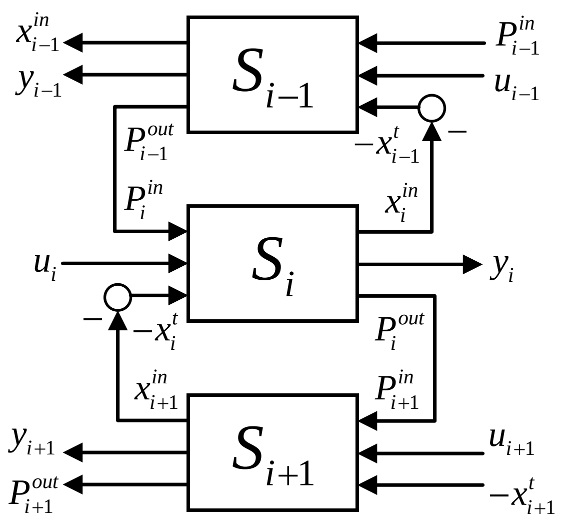


**Figure 4.3 Block diagram for the subsystems from Fig. 4.2.**

**Proof.**

Consider the storage function $V_i = \frac{1}{2} x_i^T C_i x_i$. Taking the derivative and using (4.6) yields

$$\dot{V}_i = x_i^T C_i \dot{x}_i = -x_i^T \bar{M}_i P_i + x_i^T D_i P_i^{in}. \tag{4.15}$$

Noting that $x_i^{in} = D_i^T x_i$, (4.15) simplifies to

$$\dot{V}_i = \left(x_i^{in}\right)^T P_i^{in} - x_i^T \bar{M}_i P_i. \tag{4.16}$$

Adding and subtracting $\left(x_i^t\right)^T P_i^{out}$, with $P_i^{out} = -\underline{M}_i P_i$, results in

$$\dot{V}_i = \left(x_i^{in}\right)^T P_i^{in} + \left(-x_i^t\right)^T P_i^{out} - x_i^T \bar{M}_i P_i + \left(x_i^t\right)^T \underline{M}_i P_i. \tag{4.17}$$

Using (4.7) and the definition for $y_i$ in (4.14), $\dot{V}_i$ reduces to

$$\begin{aligned} \dot{V}_i &= \left(x_i^{in}\right)^T P_i^{in} + \left(-x_i^t\right)^T P_i^{out} + y_i^T u_i - x_i^T \bar{M}_i F_i\left(\bar{x}_i\right), \\ &= \bar{y}_i^T \bar{u}_i - \bar{x}_i^T M_i F_i\left(\bar{x}_i\right). \end{aligned} \tag{4.18}$$

By Assumption 4.2, there exists $\tilde{A}_i \preceq 0$ such that

$$-\bar{x}_i^T M_i F_i\left(\bar{x}_i\right) \le \bar{x}_i^T \tilde{A}_i \bar{x}_i \le 0 \tag{4.19}$$

within a neighborhood of the equilibrium. Thus $\dot{V}_i \le \bar{y}_i^T \bar{u}_i$ within this neighborhood, proving the theorem. □

Establishing the connection between passivity and stability requires the system to be zero-state detectable (ZSD) [45].

**Definition 4.3** [45]

*The system $H$ with zero input is $\dot{x} = f\left(x,0\right)$, $y = h\left(x,0\right)$, and $Z \subset \mathbb{R}^n$ is its largest positively invariant set contained in $\left\{x \in \mathbb{R}^n \mid y = h\left(x,0\right) = 0\right\}$. The system $H$ is* zero-state detectable *(ZSD) if $x = 0$ is asymptotically stable conditionally to $Z$. If $Z = 0$, $H$ is* zero-state observable *(ZSO).*

**Lemma 4.1**

*The system* (4.8) *with outputs $\bar{y}_i$ is ZSO.*

**Proof.**

Based on the definition of $M_i$ in (4.3), the output vector is $y_i = \left[y_{i,j}\right]$, where

$$y_{i,j} = -g_{i,j}\left(x_{i,j}^{tail}, x_{i,j}^{head}\right)\left(x_{i,j}^{tail} - x_{i,j}^{head}\right), \tag{4.20}$$

For ZSO, $y_i = 0$ only if $x_i = 0$. Based on Assumption 4.1, there exists $j$ such that $g_{i,j}\left(x_{i,j}^{tail}, x_{i,j}^{head}\right) \ne 0$ if $x_{i,j}^{tail}, x_{i,j}^{head} \ne 0$. Combined with the fact that $x_{i,j}^{tail} - x_{i,j}^{head} = 0\ \forall j$ only at the equilibrium $\bar{x}_i^* = 0$, there exists $j$ such that $y_{i,j} \ne 0$ if $x_i \ne 0$, which proves the system is ZSO, and thus also ZSD. □

**Remark 4.4**

*With the $j^{th}$ output for $\mathbf{S}_i$ defined in* (4.20)*, the passivity outputs reflect the physical structure of the system. For edge $e_{i,j}$, the corresponding passivity output $y_{i,j}$ is a function of the difference* *between the tail and head vertex states $x_{i,j}^{tail}$ and $x_{i,j}^{head}$ and the nonlinear gain $g_{i,j}$ between the input $u_{i,j}$ and power flow $P_{i,j}$. While physical meaning of this output depends on the particular power flow system, in general, the passivity output for each edge represents the disparity between the neighboring vertex states weighted by the corresponding input control authority.*

## 4.4.2 Passivity of the System

With the passivity of each subsystem $\mathbf{S}_i$ established in Theorem 4.1, the structure of the interconnections between subsystems, as shown in Figs. 4.2 and 4.3, is used to assess the passivity of the overall open-loop system.

**Theorem 4.2**

*Given a system $\mathbf{S}$ composed of $N$ interacting subsystems $\mathbf{S}_i$, if each subsystem is passive from inputs $\bar{u}_i$ to outputs $\bar{y}_i$, then the overall open-loop system is passive from the inputs $\bar{u}$ to the outputs $\bar{y}$ with*

$$\bar{u} = \begin{bmatrix} P^{in} \\ u \\ -x^t \end{bmatrix}, \quad \bar{y} = \begin{bmatrix} x^{in} \\ y \\ P^{out} \end{bmatrix}, \tag{4.21}$$

*where $u = [u_i]$ and $y = [y_i]$ are all of the actuator inputs and passivity outputs for the system.*

**Proof.**

The following proves the theorem by induction on the number of subsystems in $\mathbf{S}$. Let $\mathbf{S}(N)$ denote a system with $N$ subsystems. For the base case, $N = 1$, Theorem 4.1 establishes that a system comprised of a single subsystem $S_i$ is passive from $\bar{u}_i$ to $\bar{y}_i$. For the induction step, let $\mathbf{S}(N+1)$ be the union of a system $\mathbf{S}'(N)$ with $N$ subsystems and a single subsystem $\mathbf{S}_r$, where $\mathcal{G}'$ is a subgraph of $\mathcal{G}$. By the induction hypothesis, $\mathbf{S}'$ is passive from $\bar{u}'$ to $\bar{y}'$ and by Theorem 4.1, $\mathbf{S}_r$ is passive from $\bar{u}_r$ to $\bar{y}_r$. Since, by Assumption 4.3, $\mathcal{G}$ is a connected graph, there exists at least one edge for power to flow between $\mathbf{S}'$ and $\mathbf{S}_r$. As shown in Fig. 4.3, each power flow between neighboring subsystems forms a negative feedback connection. Therefore, **S** is a system formed by the negative feedback of two passive systems and is itself passive. □

**Example 4.1**

*To demonstrate the result of Theorem* 4.2*, and the construction of vectors* $P^{in}$, $x^{in}$, $P^{out}$*, and* $x^{t}$*, consider the three interconnected subsystems shown in Figs.* 4.2 *and* 4.3*. Each subsystem has a storage function where*

$$\dot{V}_k \le \bar{y}_k^T \bar{u}_k \quad \forall k \in \{i-1, i, i+1\}. \tag{4.22}$$

*The storage function for the system is*

$$V = V_{i-1} + V_i + V_{i+1}, \tag{4.23}$$

*and*

$$\begin{aligned}\dot{V} = &\left(P_{i-1}^{in}\right)^T x_{i-1}^{in} + u_{i-1}^T y_{i-1} + \left(-x_{i-1}^{t}\right) P_{i-1}^{out} + \\ &\left(P_{i}^{in}\right)^T x_{i}^{in} + u_{i}^T y_{i} + \left(-x_{i}^{t}\right) P_{i}^{out} + \\ &\left(P_{i+1}^{in}\right)^T x_{i+1}^{in} + u_{i+1}^T y_{i+1} + \left(-x_{i+1}^{t}\right) P_{i+1}^{out}.\end{aligned} \tag{4.24}$$

*Noting that* $P_{i-1}^{out} = P_i^{in}$, $P_i^{out} = P_{i+1}^{in}$, $x_i^{in} = x_{i-1}^{t}$*, and* $x_i^{t} = x_{i+1}^{in}$,

$$\begin{aligned}\dot{V} &= \left(P_{i-1}^{in}\right)^T x_{i-1}^{in} + u_{i-1}^T y_{i-1} + u_i^T y_i + u_{i+1}^T y_{i+1} + \left(-x_{i+1}^{t}\right) P_{i+1}^{out}, \\ &= \left(P^{in}\right)^T x^{in} + u^T y + \left(-x^{t}\right) P^{out}, \\ &= \bar{u}^T \bar{y},\end{aligned} \tag{4.25}$$

*where* $P^{in} = P_{i-1}^{in}$, $x^{in} = x_{i-1}^{in}$, $P^{out} = P_{i+1}^{out}$ *, and* $x^{t} = x_{i+1}^{t}$.

**Remark 4.5**

*Note that the cascaded structure shown in Figs.* 4.2 *and* 4.3 *is used purely for notational simplicity. The fact that the negative feedback connections, on which the proof of Theorem* 4.2 *is* *based, are formed on a per edge basis allows the stability of a system with any subsystem interconnection structure to be established from the passivity of each subsystem.*

### 4.4.3 Decentralized Closed-loop Stability

The structured, passivity-preserving coupling between subsystems used to prove passivity of the open-loop system also allows for the independent design of MPC controllers for each subsystem. As shown in Fig. 4.4, each controller can be treated as another subsystem $\mathbf{C}_i$ in negative feedback with subsystem $\mathbf{S}_i$ through the inputs $u_i$ and passivity outputs $y_i$. Thus, passivity of the closed-loop system under decentralized control is achieved by enforcing passivity in each controller individually.

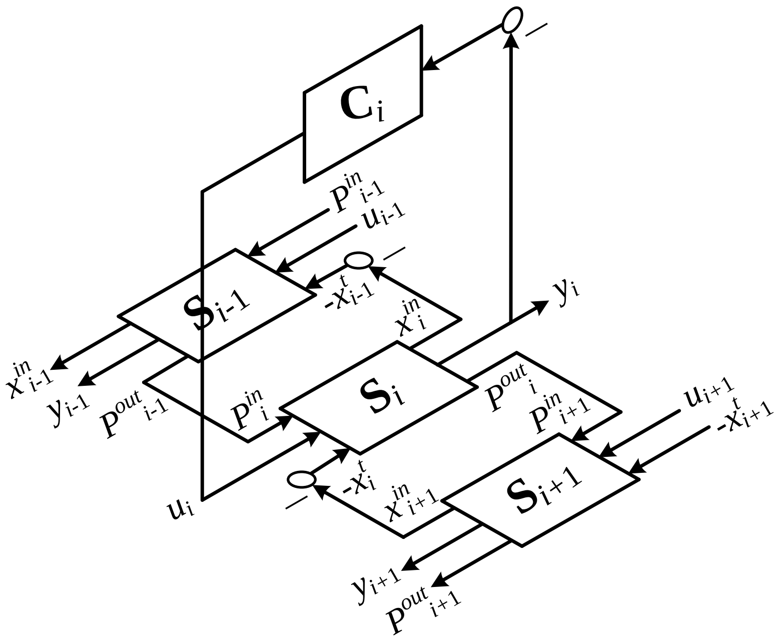


**Figure 4.4 Block diagram showing the negative feedback connection between the subsystem $\mathrm{S}_i$ and the controller $\mathrm{C}_i$. For clarity of presentation, controllers $\mathrm{C}_{i-1}$ and $\mathrm{C}_{i+1}$ for subsystems $\mathrm{S}_{i-1}$ and $\mathrm{S}_{i+1}$ are omitted from the diagram.**

Each controller $\mathbf{C}_i$ solves the following augmented nonlinear MPC optimization problem

$$\min_{u_i(\cdot)} \quad \int_0^T \ell\left(x_i(\tau), u_i(\tau), r_i(\tau), s_i(\tau)\right) d\tau \tag{4.26a}$$

$$s.t. \qquad C_i \dot{x}_i = -\bar{M}_i P_i + D_i P_i^{in}, \tag{4.26b}$$

$$P_i = F_i\left(x_i, x_i^t\right) + G_i\left(x_i, x_i^t\right) u_i, \tag{4.26c}$$

$$x_i(0) = x_{i,0}, \tag{4.26d}$$

$$y_i = -G(\bar{x}_i) M_i^T \bar{x}_i, \tag{4.26e}$$

$$x_i^{\min} - s_i(\tau) \le x_i(\tau) \le x_i^{\max} + s_i(\tau), \tag{4.26f}$$

$$u_i(\tau) \in \mathbb{U}_i, \quad \tau \in [0, T], \tag{4.26g}$$

$$\dot{z}_i = u_i^T y_i, \quad z_i(\tau) \le \beta_i, \quad \tau \in [0, T], \tag{4.26h}$$

where the stage cost $\ell(\cdot)$ is a positive definite function, $r_i(\tau)$ is a set of references to be tracked, $s_i(\tau)$ are slack variables to ensure feasibility of the state constraints, and $\mathbb{U}_i = \left\{u_i \mid u_{i,j}^{\min} \le u_{i,j} \le u_{i,j}^{\max} \ \ \forall j\right\}$ with $u_{i,j}^{\min} < 0 < u_{i,j}^{\max}$. Similar to [50], $z_i \in \mathbb{R}$ represents the accumulation of passivity. When $u_i^T y_i < 0$, the excess passivity is stored by decreasing $z_i$. This stored passivity can be depleted by allowing the system to operate with a deficiency of passivity for a finite amount of time, at which point $z_i = \beta_i$, where $\beta_i$ is a predetermined constant, and the controller is required to enforce passivity once again. This integral form of passivity reduces the conservatism associated with the more conventional passivity-based MPC found in [52]. The following theorem shows how this passivity constraint guarantees stability of the closed-loop system.

**Theorem 4.3**

*Given the system* $\mathbf{S}$ *composed of* $N$ *passive subsystems* $\mathbf{S}_i$*, if each MPC controller* $\mathbf{C}_i$*,* $i \in [1, N]$ *is augmented with the passivity constraint, as in (4.26h), then the overall closed-loop system remains stable.*

**Proof.**

With the result from Theorem 4.2, the interconnection of the subsystems preserves passivity. Thus, when proving stability of the overall closed-loop system, it is sufficient to show that passivity is preserved for each subsystem $\mathbf{S}_i$ in negative feedback with the MPC controller $\mathbf{C}_i$, as shown in Fig. 4.4. As in [48], the proof consists of demonstrating feasibility of (4.26) and using this feasibility to show stability. From Theorem 2.28 in [45], when there is no throughput, i.e. $y = h(x)$, the feedback $u = -y$ achieves asymptotic stability if and only if the system is zero state detectable (ZSD). With ZSO established in Lemma 4.1, and thus also ZSD, it holds that $u_i = -y_i$ stabilizes (4.8). This property also holds true for $u_i = -\alpha_i y_i$, where $\alpha_i > 0$. For any $y_i$ there exists $\alpha_i > 0$ such that $u_i = -\alpha_i y_i \in \mathbb{U}_i$. Since $u_i = -\alpha_i y_i$ is a stabilizing candidate control law with $\dot{z}_i = u_i^T y_i = -\alpha_i y_i^T y_i \le 0$, the constraint $z_i \le \beta_i$ is always feasible.

To prove stability, let $V_{C_i}(z_i) = z_i + \beta_i$ be a storage function for the controller $\mathbf{C}_i$, as is done in [50], where $V_{C_i}(z_i) \ge 0$ since $z_i \le \beta_i$. Thus the storage function for the closed-loop subsystem is $V_i + V_{C_i}$ where

$$\begin{aligned}\dot{V}_i + \dot{V}_{C_i} &= \bar{y}_i^T \bar{u}_i - \dot{z}_i, \\ &= \bar{y}_i^T \bar{u}_i - u_i^T y_i, \\ &= \begin{bmatrix} \left(x_i^{in}\right)^T & \left(P_i^{out}\right)^T \end{bmatrix} \begin{bmatrix} P_i^{in} \\ -x_i^t \end{bmatrix}. \end{aligned} \tag{4.27}$$

Thus the closed-loop subsystem is passive with respect to the inputs and outputs that couple subsystem $\mathbf{S}_i$ to neighboring subsystems. With each closed-loop subsystem preserving passivity, the overall closed-loop system remains passive and stable. □

**Remark 4.6**

*The passivity constraint* (4.26h) *is interpreted as a sector condition* [44] *as follows. The constraint* $z_i(\tau) \le \beta_i$ *limits the time spent in the sector* $u_i^T y_i > 0$. *Based on the definition of* $y_{i,j}$ *for each edge from* (4.20), *this sector restriction limits operation that would cause a positive* *power flow from* $x_{i,j}^{tail}$ *to* $x_{i,j}^{head}$ *when* $x_{i,j}^{head} - x_{i,j}^{tail} > 0$ *thus preventing this difference from growing further. Thus the controller must eventually make control decisions that prevents the difference between neighboring states from growing and, as proved in Theorem* 4.3, *such a control input always exists.*

**Remark 4.7**

*This approach differs from many of the approaches in literature* [18], [55], [56], [59] *which assess the stability of a system of subsystems based on stability criteria of a global matrix which captures the passivity/dissipativity properties of each subsystem, the network topology of the subsystems, and the gains of the coupling between subsystems. For example, in* [55] *input-feedforward output-feedback passivity of each subsystem is quantified using parameters* $\nu_i$ *and* $\rho_i$ *where* $\dot{V} \le u_i^T y_i - \nu_i u_i^T u_i - \rho_i y_i^T y_i$. *Then passivity and stability of the entire system is established by evaluating the quasi-dominance of a matrix comprised of these* $\nu_i$ *and* $\rho_i$. *Thus determining stability requires analysis of the global system. The proposed approach leverages the specific structure of the coupling between subsystems, allowing passivity and stability of the system to be assessed locally for each subsystem without the need to analyze any global properties of the system.*

**Remark 4.8**

*This set of decentralized controllers can be thought of as a stability-assurance control layer similar to the supervisory stability layer (SSL) from* [60]. *Regardless of information or references sent to the individual decentralized controller, the passivity-based stability constraint will remain feasible and prevent the system from going unstable.*

### 4.4.4 Hierarchical Closed-loop Stability

The decentralized control layer shown in Fig. 4.4 serves to guarantee stability of the overall system but may lead to unacceptable control performance due to the unknown effects of coupling between subsystems in the form of power flow from one subsystem to another. Thus, hierarchical control can be used to improve the overall control performance of the system in the form of additional control levels above the decentralized level, as shown in Fig. 4.5. The upper-level controllers $\mathbf{C}_{1,1}$, $\mathbf{C}_{2,1-2}$ are designed to account for the coupling between subsystems and send references to the low-level controllers $\mathbf{C}_{3,1-4}$ to achieve better coordination among subsystems.

Due to the stability guarantee of the low-level decentralized controllers, the upper-level controllers can be designed with only performance in mind and do not have to be augmented to achieve stability. This provides the control design engineer a large degree of flexibility in the formulation of the structure and individual controllers at the upper-levels of the hierarchy.

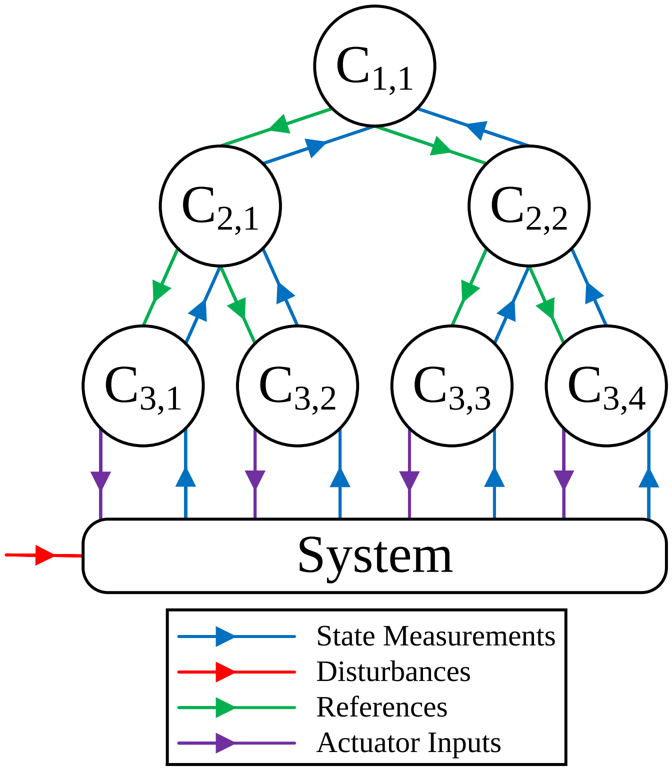


**Figure 4.5 Example hierarchical control structure used to improve control performance via coordination among subsystems. Only controllers $C_{3,1-4}$ require passivity constraints, forming a stability-assurance layer.**

## 4.5 Numerical Example

The efficacy of the decentralized, passivity-based stability constraints is demonstrated with the following numerical example. Fig. 4.6 shows the graph of a fluid tank system, which has the same structure as the example system from Chapter 3. Each vertex $v_i$ corresponds to a

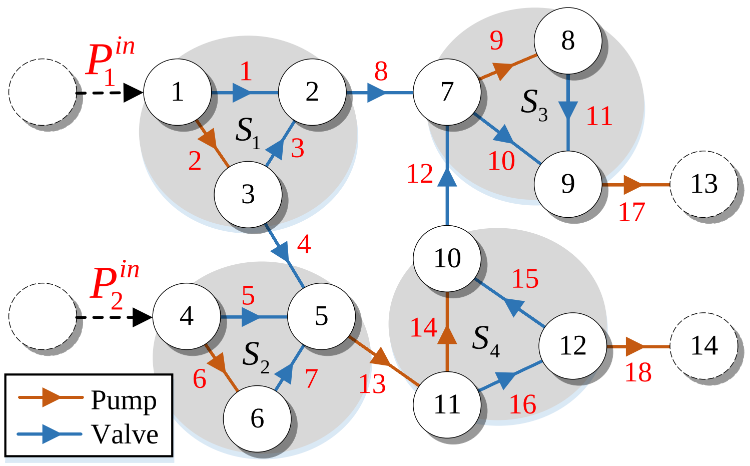


**Figure 4.6 Graph for example fluid tank system with four subsystems.**

fluid tank and has a state $x_i$ which represents the height of the fluid in the tank in meters. For this hydraulic system, conservation of energy for each vertex from (4.1) corresponds to conservation of mass for each tank,

$$\rho A_i \dot{x}_i = \sum_{j \in E_i^{in}} \dot{m}_j - \sum_{j \in E_i^{out}} \dot{m}_j, \tag{4.28}$$

where $\rho = 1000\,kg/m^3$ is the density of the fluid and $A_i = \pi d_i^2/4$ is the cross-sectional area of the tanks, all with diameter $d_i = 0.1m$. The power flows from (4.2) corresponding to mass flow rate between the tanks. For flows between tanks controlled by pumps, as indicated in Fig. 4.6, the mass flow rate along edge $e_j$ is

$$\dot{m}_j = Disp \cdot \omega_j - k_{leak}\left(x_j^{head} - x_j^{tail}\right), \tag{4.29}$$

where $Disp = 1kg/rev$ is the displacement of the pump, $\omega_j$ is the variable pump speed in revolutions per second, and $k_{leak} = 0.005\,kg/(m \cdot s)$ is a leakage coefficient. For flows between tanks controlled by valves, the mass flow rate along edge $e_j$ is

$$\dot{m}_j = C_D^{\max}\left(x_j^{tail} - x_j^{head}\right)a_j, \tag{4.30}$$

where $C_D^{\max} = 0.5 kg/(m \cdot s \cdot \%)$ is the maximum discharge coefficient and $a_j$ is the variable valve aperture in percent open. Based on these definitions for mass flow rate along each edge, the conditions in Assumption 4.1 hold and the open-loop system is stable, satisfying Assumption 4.2.

The overall system is composed of four dynamically coupled subsystems, as indicated in Fig. 4.6. As in Chapter 3, the four subsystem graphs are formulated and shown in Fig. 4.7. From these graphs and (4.28)-(4.30), the corresponding subsystem dynamics from (4.8) can be derived. Additionally, the passivity output $y_i$ for each subsystem can be determined as defined in (4.14). For a pump edge $e_{i,j}$ in subsystem $\mathbf{S}_i$, the corresponding passivity output is

$$y_{i,j} = -Disp \cdot \left(x_{i,j}^{tail} - x_{i,j}^{head}\right). \tag{4.31}$$

Similarly, for a valve edge, the corresponding passivity output is

$$y_{i,j} = -C_D^{\max}\left(x_{i,j}^{tail} - x_{i,j}^{head}\right)\left(x_{i,j}^{tail} - x_{i,j}^{head}\right). \tag{4.32}$$

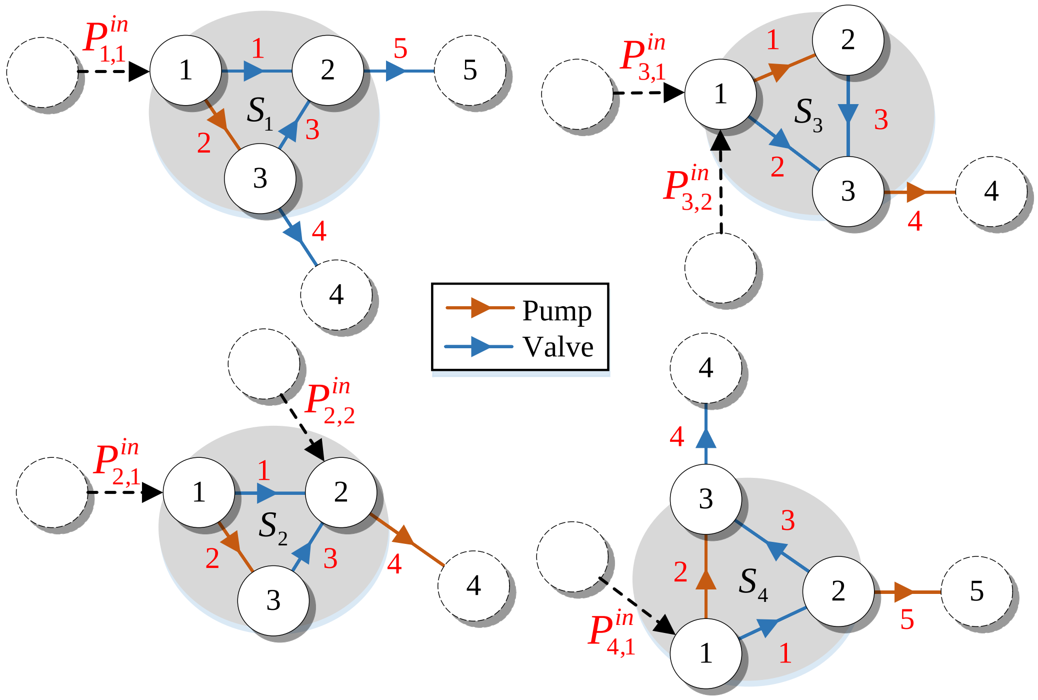


**Figure 4.7 Decomposition of the example system graph into four subsystem graphs used to develop the four decentralized MPC controllers.**

For each subsystem $\mathbf{S}_i$, a MPC-based controller $\mathbf{C}_i$ is designed based on the optimization problem from (4.26), where

$$\ell\left(x_i(\tau), u_i(\tau), r_i(\tau)\right) = \|x_i - r_i\|_2^2 + 0.01\|u_i\|_2^2, \tag{4.33}$$

$T = 2\text{sec}$, $\mathbb{U}_i = \left\{u_i \mid u_{i,j} \in \mathbb{R}\ \forall j\right\}$, and $\beta_i = 0$. This optimization problem is discretized with $\Delta t = 1\text{sec}$ and solved with an update rate of 1 Hz. YALMIP [41] and IPOPT [61] are used to formulate and solve this optimization problem for each controller.

To demonstrate the role of the passivity-based stability constraints as a stability-assurance layer within a hierarchical control framework, as discussed in Remark 4.8, an upper-level controller $\mathbf{C}_0$ is also designed, which sends references $r_i$ to be tracked by each subsystem. The structure of this control hierarchy is shown in Fig. 4.8. While the hierarchical control design procedure from Chapter 3 could be applied for the design of $\mathbf{C}_0$, for this Chapter, $\mathbf{C}_0$ has been designed using a linearization of the system about the initial state $x(0)$ and has the form $\mathbf{C}_0 : r = -Zx$. Following the same procedure, a small change in controller design parameters resulted in two different $Z$ matrices, $Z_{stable}$ and $Z_{unstable}$, which as their name suggests, resulted in stable and unstable closed-loop systems, as seen in Fig. 4.9. It is important to note that the two matrices are very similar with $Z_{stable} \approx Z_{unstable}$. In fact, using the normalized distance between the two matrices, defined as

$$\Delta Z = 100\frac{\|Z_{stable} - Z_{unstable}\|_2}{\|Z_{stable}\|_2}, \tag{4.34}$$

the similarity of the two matrices can be quantified as $\Delta Z = 1.8\%$. The similarity of these matrices and the corresponding disparity of their closed-loop behavior highlights a key challenge when developing hierarchical controllers in practice. A priori assessment of the overall closed-loop stability of the system can be very difficult due to the interaction of multiple control loops and multiple subsystems. This is especially true with MPC-based controllers which, in general, lack a closed-form control law.

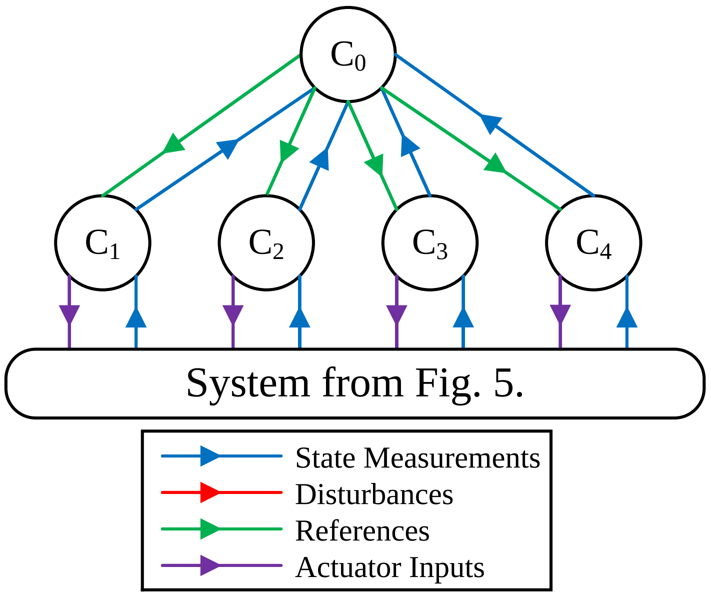


**Figure 4.8 Simple control hierarchy for the example system where $\mathbf{C}_{1-4}$ are passivity-constrained decentralized MPC controllers and $\mathbf{C}_0$ is a centralized reference generator (signal coloring is the same as Fig. 4.5).**

The stabilizing effect of the decentralized passivity constraints is shown in Fig. 4.9 for representative states $x_2$ and $x_{10}$ in subsystems $\mathbf{S}_1$ and $\mathbf{S}_4$, respectively. For both designs of $\mathbf{C}_0$, the closed-loop system remains stable and converges to the equilibrium. Note that more aggressive transient behavior could be achieved by increasing the value of $\beta_i$. For the current simulation results $\beta_i = 0$ and the accumulated passivity $z_i$ is shown for $\mathbf{S}_1$ in Fig. 4.10 for the nominal and passivity-constrained control formulations. From the accumulated passivity, $z_1$, for the nominal MPC with $Z_{stable}$ (red trace in the first subplot of Fig. 4.10), it is clear that even the stable system response was not instantaneously passive, $\dot{z}_1 = u_1^T y_1 > 0$, for the majority of the transient. This highlights the well-known potential conservatism associated with a passivity-based stability approach. This conservatism can be reduced by increasing the value of $\beta_i$ and allowing the system to violate passivity longer. Thus $\beta_i$ should be designed based on the application specific trade-off between the benefit of aggressive control and the cost of potentially, yet temporarily, following an unstable trajectory.

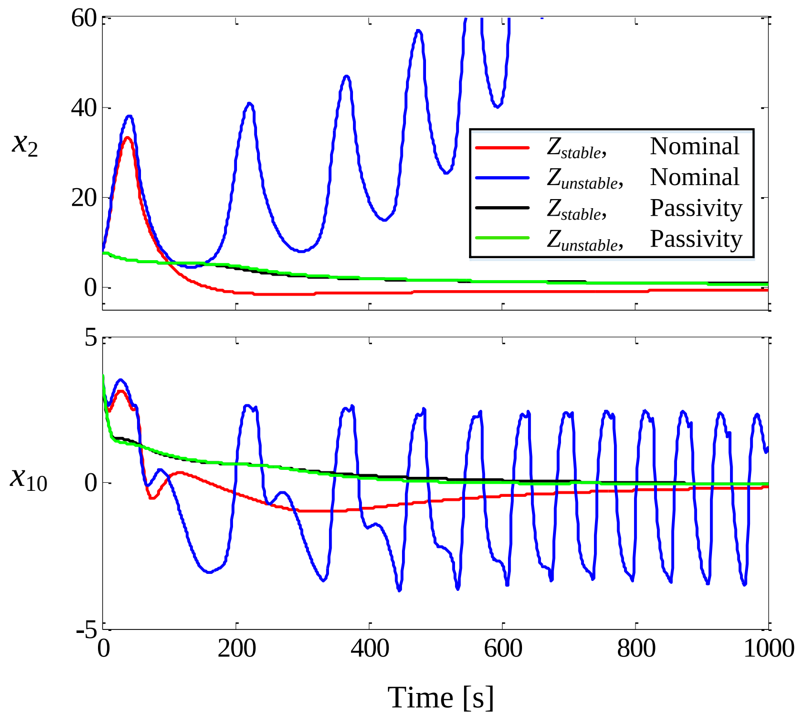


**Figure 4.9 Representative state trajectories for $x_2$ and $x_{10}$ with stable and unstable reference generator formulations and for the nominal and passivity-constrained MPC designs.**

## 4.6 Chapter Summary

This Chapter presented a purely decentralized procedure for augmenting existing model predictive control formulations with a passivity-based constraint to guarantee closed-loop stability of a power flow system. By establishing passivity of individual subsystems and analyzing the structure of the interactions between subsystems, a stability guarantee for the overall closed-loop system was achieved through simple, local augmentations to each controller in the form of passivity constraints. While the control formulation in this Chapter used slack variables on the state constraints to avoid infeasibility of the optimization problem, the following Chapter presents a hierarchical control formulation for linear graph-based power flow systems that guarantees constraint satisfaction in the presence of model and disturbance signal uncertainty.

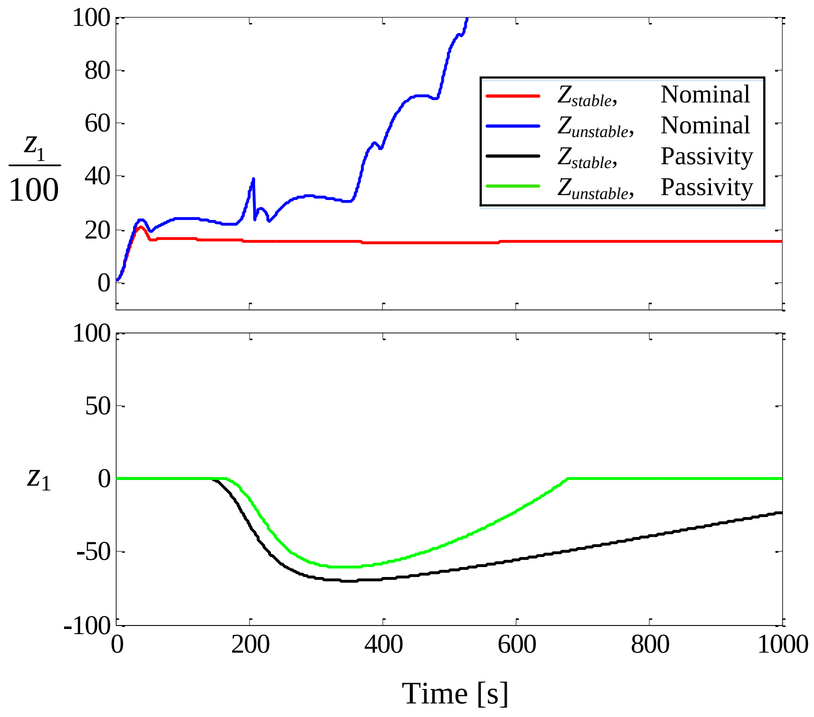


**Figure 4.10 Trajectories for $z_1$ for subsystem $S_1$ with the stable and unstable reference generator formulations and for the nominal and passivity-constrained MPC designs. Note that these trajectories are plotted separately due to the disparity in the magnitudes and sign of the trajectories for the nominal and passivity-constrained scenarios.**

# Chapter 5
# Robust Feasibility

## 5.1 Motivation

With increasing performance demands, the power flow systems onboard vehicles are required to safely function at the limit of their operating envelop. To maximize the capability of the vehicle, systems must operate very closely to their actuator and state constraints without exceeding these bounds. From a controls perspective, guaranteeing that system operation will satisfy these constraints is critical for practical implementation. However, even when using MPC to predict the future trajectories of the system to anticipate and avoid possible constraint violations, the presence of model and disturbance signal uncertainty makes providing such guarantees very difficult in practice. While a hierarchical control framework provides numerous advantages in terms of control performance, the decentralization of control decisions and the complexity of controller interactions make establishing constraint satisfaction guarantees even more challenging.

## 5.2 Background

Building on a number of robust centralized [62]–[65] and distributed [66]–[69] MPC formulations, several robust hierarchical MPC formulations have been developed in the literature. In [70], a two-level hierarchical control approach is presented with a slow higher-level and fast lower-level controller. The lower-level controller bounds deviations between the control decisions made at each level and the higher-level controller is made robust to these deviations using a min-max robust MPC formulation. This approach is extended in [71] by allowing the lower-level of control to consist of $m$ controllers for systems with decoupled actuator dynamics.

The goal of the upper-level controller is to determine which actuators to enable along with their desired control inputs, while the lower-level controllers determine the actual control inputs that account for the dynamics of the actuators at a faster timescale. This work is further formalized in [72]. Additional approaches are presented in [73]–[75] where two-level hierarchical controllers are developed that act similar to reference governors, using dynamic actuators to satisfy system constraints with guaranteed stability.

In each of these efforts, a two-level hierarchical framework is developed to handle the timescale separation between the system and actuator dynamics. However, in practice many systems have more than two timescales and an $N$-level hierarchical controller would be more effective in controlling each timescale. While [76] presents a more generic mathematical formulation for $N$-level hierarchical MPC, theoretical properties like robust stability and feasibility are not established and the authors state that "much work is still needed."

The goal of this Chapter is to modify the generic hierarchical control formulation from Chapter 3 into a specific formulation that can maintain robust feasibility of actuator and state constraints in the presence of model and disturbance signal uncertainty. The main features of the proposed approach are:

1) the control hierarchy for a system with $N_{sub}$ subsystems has $N$ levels, with $n_i$ controllers at the $i^{th}$ level ($i \in [1, N]$, $n_1 = 1$, $n_N = N_{sub}$),
2) the formulation guarantees state and actuator constraint satisfaction in the presence of both model and disturbance signal uncertainty,
3) model reduction is employed to reduce computational costs of the upper-level controllers, and
4) all constraints are simple and numerically efficient to calculate offline and implement online. With these benefits, the proposed approach relies on several assumptions about the system and control formulation that are discussed throughout the Chapter.

## 5.3 Linear Graph System Model

### 5.3.1 System Dynamics

Following the graph-modeling framework from Chapter 2, consider the power flow system $\mathbf{S}$ represented by an oriented graph $\mathcal{G} = (V, E)$ of order $N_v$ with set of vertices $V = \{v_i\}$, $i \in [1, N_v]$ and of size $N_e$ with set of edges $E = [e_j]$, $j \in [1, N_e]$. Each oriented edge $e_j \in E$ represents a path for power flow in $\mathbf{S}$, where positive power $P_j$ flows from the tail vertex $v_j^{tail}$ to the head vertex $v_j^{head}$. Each vertex $v_i \in V$ has an associated state $x_i$ that represents the amount of energy stored in that vertex. Thus the dynamic for the state of each $v_i$ satisfies the discrete-time energy conservation equation

$$C_i \frac{x_i^+ - x_i}{\Delta t} = \sum_{e_j \in E_i^{in}} P_j - \sum_{e_j \in E_i^{out}} P_j, \tag{5.1}$$

where $\Delta t$ is the time step and $C_i > 0$ is the energy storage capacitance of vertex $v_i$ while $E_i^{in} = \{e_j \mid v_j^{head} = v_i\}$ and $E_i^{out} = \{e_j \mid v_j^{tail} = v_i\}$ are the sets of edges oriented into and out of the vertex $v_i$.

**Assumption 5.1**

*The power flow $P_j$ along edge $e_j$ is defined as*

$$P_j = a_j x_j^{tail} - b_j x_j^{head} + c_j u_j + \Delta P_j, \tag{5.2}$$

*where $x_j^{tail}$ and $x_j^{head}$ are the states of the tail and head vertices $v_j^{tail}$ and $v_j^{head}$, $u_j$ is an associated actuator input, $(a_j, b_j) \geq 0$, $c_j \neq 0$, and $|\Delta P_j| \leq \Delta P_j^{\max}$.*

**Remark 5.1**

*While a more generic power flow relationship is considered in (4.2), the set operations used for constraint tightening in this Chapter rely on a linear system model and thus a linear power flow* *relationship. However, to allow the results of this Chapter to be applicable to a wider class of systems, $\Delta P_j$ in (5.2) is treated as an unknown, yet bounded, disturbance. This disturbance represents both model uncertainty and bounded linearization error when using (5.2) to approximate nonlinear power flow relationships.*

Following the same graph representation used in Chapter 4, the system $\mathbf{S}$ has states $x \in \mathbb{R}^{N_v}$ that each satisfy (5.1) and power flows $P \in \mathbb{R}^{N_e}$ that each satisfy (5.2). The disturbances to $\mathbf{S}$ capture how power enters and exits the system, with inlet power flows $P^{in} \in \mathbb{R}^{N_s}$ and sink states $x^t \in \mathbb{R}^{N_t}$. As indicated by dashed lines in Fig. 5.1, the inlet power flow edges are not included in $\mathcal{G}$. Also indicated by dashed lines in Fig. 5.1, the sink states are *not* states of $\mathbf{S}$, but the sink vertices and the edges connecting $\mathbf{S}$ to the sink vertices *are* included in $\mathcal{G}$. Power flows along this type of edge, denoted $P^{out} \in \mathbb{R}^{N_t}$, each follow the relationship from (5.2). Finally, each system has a subset $x^{in} \in \mathbb{R}^{N_s}$ of the states $x$ that represents the states directly affected by the inlet power flows $P^{in}$. Note that Figs. 4.1 and 5.1 are nearly identical, with Fig. 4.1 showing the subsystem-based notation used in Chapter 4 while Fig. 5.1 shows the system-based notation used in this Chapter.

Let $M = \left[m_{i,j}\right] \in \mathbb{R}^{(N_v+N_t)\times N_e}$ be the incidence matrix of graph $\mathcal{G}$ [36] where

$$m_{i,j} = \begin{cases} +1 & \text{if } v_i \text{ is the tail of } e_j \\ -1 & \text{if } v_i \text{ is the head of } e_j \\ 0 & \text{else} \end{cases}. \tag{5.3}$$

Then, based on (5.1), the system dynamics are

$$\begin{bmatrix} C\left(x^+ - x\right) \\ \left(x^t\right)^+ - x^t \end{bmatrix} = -\Delta t M P + \begin{bmatrix} D \\ 0 \end{bmatrix} P^{in}, \tag{5.4}$$

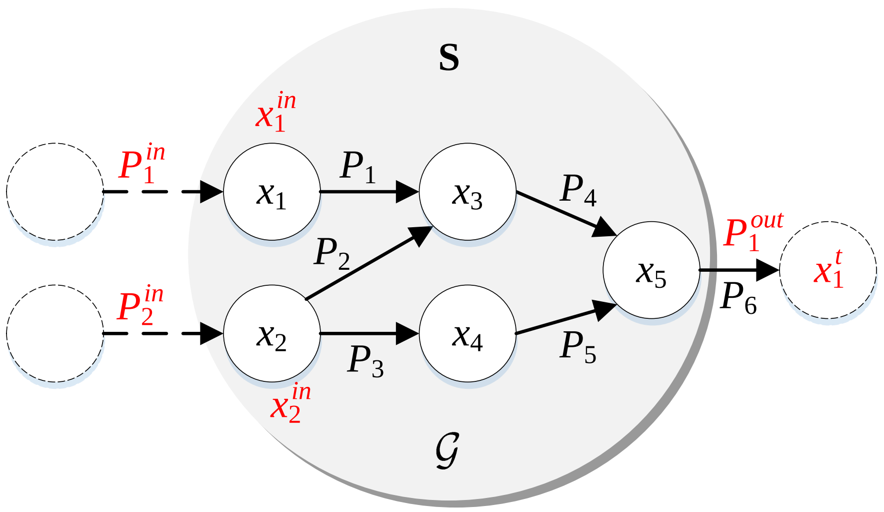


**Figure 5.1 Notional system exemplifying the graph-based power flow representation with key power flows and states highlighted in red. Dashed lines indicate elements that serve as disturbances to the system.**

where $C = diag\left(\left[C_i\right]\right)$ is a diagonal matrix of the vertex capacitances and $D = \left[d_{i,j}\right] \in \mathbb{R}^{N_v \times N_s}$ where

$$d_{i,jk} = \begin{cases} 1 & \text{if } v_i \text{ is the head of } P_j^{in} \\ 0 & \text{else} \end{cases}. \tag{5.5}$$

Since $x^t$ are disturbances to the system, not states, $M$ is partitioned as $M = \begin{bmatrix} \bar{M} \\ \underline{M} \end{bmatrix}$, with $\bar{M} \in \mathbb{R}^{N_v \times N_e}$ and $\underline{M} \in \mathbb{R}^{N_t \times N_e}$, resulting in

$$C\left(x^+ - x\right) = -\Delta t \bar{M} P + D P^{in}. \tag{5.6}$$

From (5.2), the vector of power flows in $\mathbf{S}$ is

$$\begin{aligned} P &= M_{a,b}^T \begin{bmatrix} x \\ x^t \end{bmatrix} + \beta u + \Delta P, \\ &= \bar{M}_{a,b}^T x + \underline{M}_{a,b}^T x^t + \beta u + \Delta P, \end{aligned} \tag{5.7}$$

where $M_{a,b} = \left[m_{i,j}\right] \in \mathbb{R}^{(N_v + N_t) \times N_e}$ is a weighted incidence matrix with

$$m_{i,j} = \left\{ \begin{array}{cc} a_j & \text{if } v_i \text{ is the tail of } e_j \\ -b_j & \text{if } v_i \text{ is the head of } e_j \\ 0 & \text{else} \end{array} \right\} \tag{5.8}$$

$\beta = diag\left(\left[c_j\right]\right)$, $u = \left[u_j\right]$, and $\Delta P = \left[\Delta P_j\right]$. Thus the dynamics of $\mathbf{S}$ are given by

$$\mathbf{S}: \; x^+ = Ax + B\beta u + V_1 P^{in} + V_2 x^t + V_3 \Delta P, \tag{5.9}$$

where $A = I - \Delta t C^{-1} \bar{M} \bar{M}_{a,b}^T$, $B = -\Delta t C^{-1} \bar{M}$, $V_1 = \Delta t C^{-1} D$, $V_2 = -\Delta t C^{-1} \bar{M} \underline{M}_{a,b}^T$, and $V_3 = -\Delta t C^{-1} \bar{M}$.

### 5.3.2 Dynamic Timescales

Let the state vector be subdivided as $x = \begin{bmatrix} \boldsymbol{x}_1^T & \boldsymbol{x}_2^T & \dots & \boldsymbol{x}_N^T \end{bmatrix}^T$, where $\boldsymbol{x}_i \in \mathbb{R}^{N_v^i}$ denotes a vector of states with the $i^{th}$ timescale where $\sum_{i=1}^N N_v^i = N_v$ and $C_j < C_i < C_k$ for $x_j \in \boldsymbol{x}_j$, $x_i \in \boldsymbol{x}_i$, $x_k \in \boldsymbol{x}_k$, and $j > i > k$. Note that the number of levels of the hierarchy, $N$, matches the number of timescales of the system.

### 5.3.3 Local Constraints

The system is subject to box state and actuator input constraints

$$x \in \mathcal{X} \subset \mathbb{R}^{N_v}, \quad u \in \mathcal{U} \subset \mathbb{R}^{N_e}, \tag{5.10}$$

where $\mathcal{X} = \left\{ x \in \mathbb{R}^{N_v} \mid \underline{x} \le x \le \bar{x} \right\}$, $\mathcal{U} = \left\{ u \in \mathbb{R}^{N_e} \mid \underline{u} \le u \le \bar{u} \right\}$, and each set contains the origin.

### 5.3.4 Nominal System

While (5.9) represents the true system behavior, the nominal state trajectories, inputs, and disturbances used by the hierarchical controller are denoted as $\hat{x}$, $\hat{u}$, $\hat{P}^{in}$, and $\hat{x}^t$, resulting in the nominal system

$$\hat{\mathbf{S}}: \; \hat{x}^+ = A\hat{x} + B\beta\hat{u} + V_1 \hat{P}^{in} + V_2 \hat{x}^t, \tag{5.11}$$

Thus the unknown disturbances to the system, to which the hierarchical controller must be robust, are $\Delta P^{in} = P^{in} - \hat{P}^{in}$, $\Delta x^t = x^t - \hat{x}^t$, and $\Delta P$. These disturbances are assumed to be bounded with $\Delta P^{in} \in \Delta \mathcal{P}^{in}$, $\Delta x^t \in \Delta \mathcal{X}^t$, and $\Delta P \in \Delta \mathcal{P}$. As with $\mathcal{X}$ and $\mathcal{U}$, these disturbance sets are assumed to contain the origin and are defined by box constraints.

## 5.3.5 Control Objective

The control objective is to satisfy all state and input constraints from (5.10) while minimizing the finite-horizon, system-wide cost function

$$J(k) = \sum_{k=0}^{N_{op}} \ell\Big(x(k), u(k), \Lambda^s(k), \Lambda^t(k)\Big), \tag{5.12}$$

where $N_{op}$ is the operational duration of the system in time steps and $\ell(\cdot)$ is a generic running cost. To minimize deviation from $\hat{P}^{in}_{des}$ and $\hat{x}^t_{des}$, (5.12) is designed to heavily penalize $\left\|\Lambda^s(k) - I\right\|_2^2$ and $\left\|\Lambda^t(k) - I\right\|_2^2$.

## 5.3.6 Feedback Integralization

To significantly simplify the hierarchical control formulation, the control law

$$u = \beta^{-1}\left(\bar{P} - M_{a,b}^T \begin{bmatrix} x \\ x^t \end{bmatrix}\right) \tag{5.13}$$

is implemented to convert the linear system (5.9) into the integrator system

$$\mathbf{S}_0: \; x^+ = x + B\bar{P} + V_1 P^{in} + V_3 \Delta P, \tag{5.14}$$

where $\bar{P}$ is the desired power flow vector. The corresponding nominal control law

$$\hat{u} = \beta^{-1}\left(\hat{P} - M_{a,b}^T \begin{bmatrix} \hat{x} \\ \hat{x}^t \end{bmatrix}\right) \tag{5.15}$$

converts the nominal linear system (5.11) into the integrator system

$$\hat{\mathbf{S}}_0 : \ \hat{x}^+ = \hat{x} + B\hat{P} + V_1\hat{P}^{in}, \tag{5.16}$$

where $\hat{P}$ is the nominal power flow vector.

**Remark 5.2**

*The feedback control law from* (5.15), *referred to as* feedback integralization, *forces nominal state trajectories to evolve piecewise linearly. As discussed in the following Section, the controllers at the upper levels of the hierarchy use slow update rates designed to match the slow dynamics under control at that level. When converting a discrete-time model from a fast update rate to a slow update rate, as described in* [70], *no additional model error is introduced by this conversion, but information about the state trajectory between the slow updates is lost. By converting the linear system to an integrator system,* (5.15) *ensures that the intersample state trajectory is bounded by state values at the neighboring slow time steps, i.e. no over/undershoot between slow time steps occurs. This is key in guaranteeing that the state trajectories determined by the upper-level controllers are feasible for tracking by the lower-level controllers. Fig.* 5.2 *demonstrates this notion and the benefit of the feedback integralization. Finally,* (5.13) *can be implemented for each edge independently using* $u_j = \frac{1}{c_j}\left(\bar{P}_j - a_j x_j^{tail} + b_j x_j^{head}\right)$.

## 5.4 Hierarchical Control Structure

### 5.4.1 Subsystem Interconnections

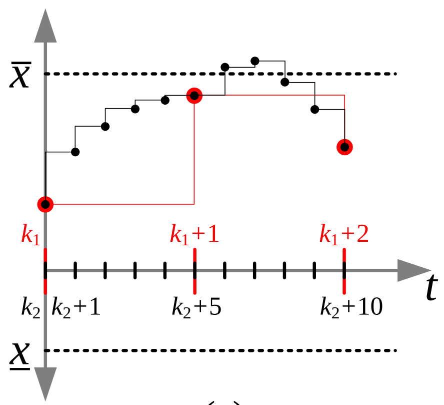
Let $\mathbf{S}$ be decomposed into $N_{sub}$ non-overlapping subsystems $\mathbf{S}_i$, $i \in [i, N_{sub}]$.

**Definition 5.1**

*A* $u,v-path$ *in* $\mathcal{G}$ *is a sequence of oriented edges connecting two distinct vertices* $u,v \in V$, *not including any sink vertices as intermediate vertices.*

**Assumption 5.2**

*If* $v_i \in \mathbf{S}_i$ *and* $v_j \in \mathbf{S}_j, i \neq j$ *and there exists a* $u,v-path$ *in* $\mathcal{G}$, *then there does not exist a* $v,u-path$ *in* $\mathcal{G}$.



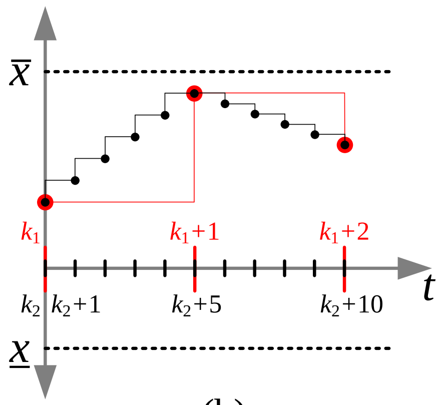


**Figure 5.2 (a) The upper-level controller plans a feasible state trajectory at the slow time step $k_1$ but the lower-level controller is unable to track this trajectory at the faster time step $k_2$ without violating state constraints. (b) Using the feedback integralization control law from (5.15), the system follows piecewise linear state trajectories; thus any trajectory that is feasible at the slower time step $k_1$ is also feasible at the faster time step $k_2$.**

**Remark 5.3**

*Assumption* 5.2 *prevents cyclical connections between subsystems, simplifying the process for identifying* $\Delta\mathcal{P}^{in}$ *for power flows coming from neighboring subsystems. An example of an acyclic graph of subsystems is shown in Fig.* 5.3. *If systems with cyclically connected subsystems are of interest, a more centralized approach for calculating* $\Delta\mathcal{P}^{in}$ *can be adopted from* [66].

## 5.4.2 Control Structure

To control the $N$ dynamic timescales, a hierarchical control framework with $N$ levels is proposed where one of the main functions of controllers at the $i^{th}$ level is to control states at the $i^{th}$ timescale. At the lowest level of the hierarchy, each subsystem $\mathbf{S}_j$ has a corresponding controller $\mathbf{C}^{[N,j]}$ that uses a nominal subsystem model, denoted $\hat{\mathbf{S}}^{[N,j]}$. These controllers have a time step of $\Delta t_N = \Delta t$ and use the time index $k_N$. The controllers at levels $N-1$ through 1 of the hierarchy coordinate control decisions between these subsystems and have slower update rates to more effectively control the slower timescale dynamics of the system. Thus the $j^{th}$

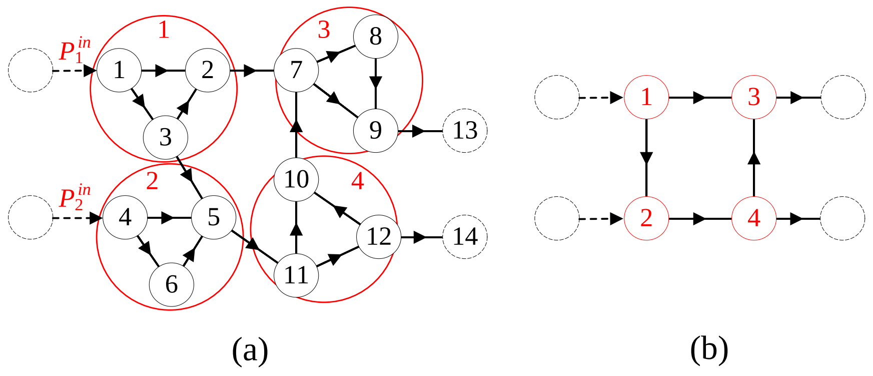


**Figure 5.3 (a) An example system graph decomposed into subsystems. (b) The interconnection of these subsystems is acyclic.**

controller at the $i^{th}$ level, $\mathbf{C}^{[i,j]}$, has a time step of $\Delta t_i$, where $\Delta t_i = \nu_{i-1}\Delta t_{i-1}$ and $\nu_{i-1}$ is a positive integer, with a discrete time index $k_i$. The $i^{th}$ level of the hierarchy has $n_i$ controllers where $n_N = N_{sub}$, $n_1 = 1$, and $n_i \le n_{i+1}$.

Fig. 5.4 provides an example of a 3-level hierarchy that demonstrates the proposed formulation.

### 5.4.3 Nominal Subsystems for Level *N* Controllers

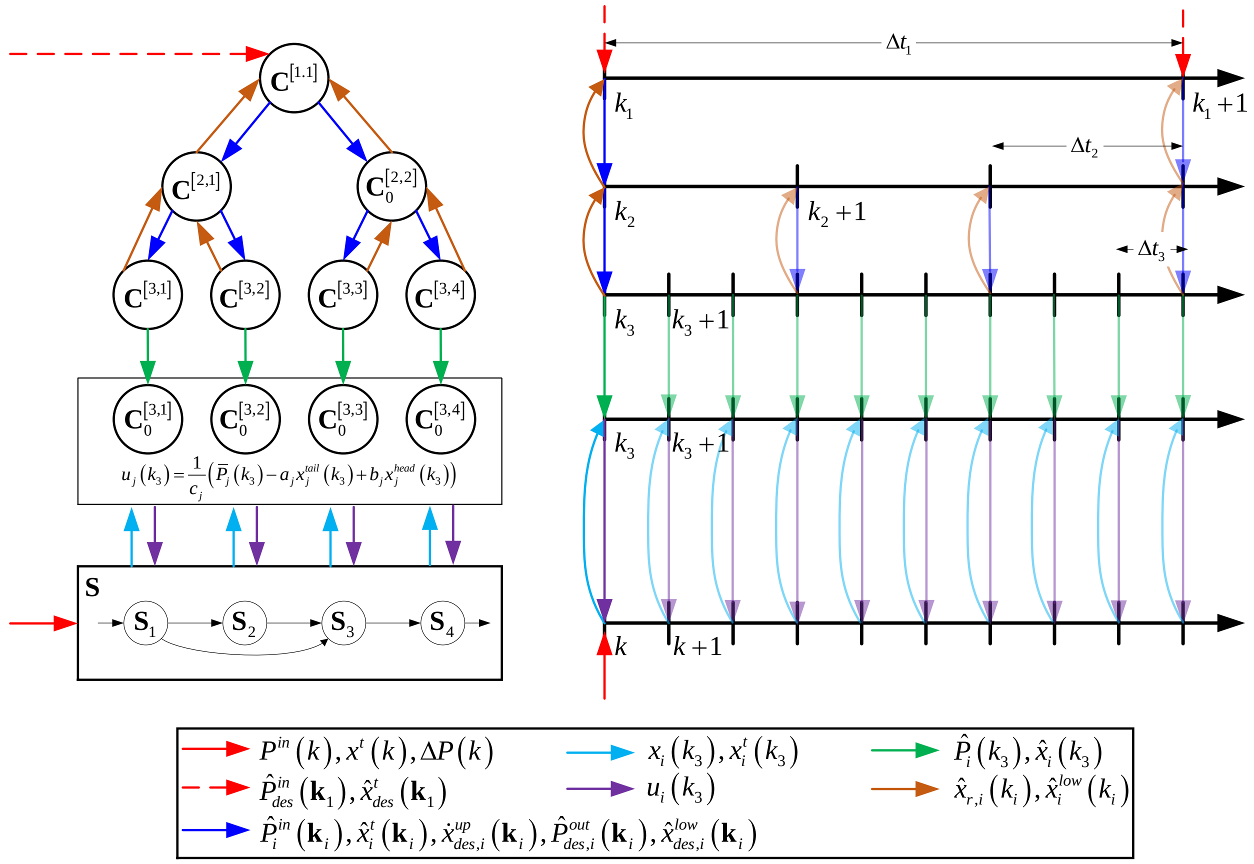
The nominal subsystem model for $\mathbf{S}_j$ follows the same model development used for the entire system model in Section 5.3 where

$$\hat{\mathbf{S}}_0^{[N,j]}:\ \hat{x}^+ = \hat{x} + B\hat{P} + V_1\hat{P}^{in}. \tag{5.17}$$

Note the abuse of notation used for improved readability, where $\hat{x}$, $\hat{P}$, $\hat{P}^{in}$, $B$, and $V_1$ are all specific to $\mathbf{S}_j$ and occur at time index $k_N$.

### 5.4.4 Nominal Reduced Subsystems for Level *i* Controllers, *i* ∈ [1,*N*- 1]

An agglomerative, or bottom-up, clustering scheme [77] is used to form the subsystems at the $i^{th}$ level based on the subsystems at level $i+1$. At the $i^{th}$ level, the $j^{th}$ subsystem $\mathbf{S}^{[i,j]}$, $j \in [1, n_i]$, consists of subsystems $\mathbf{S}^{[N,l]}$, $l \in I_{i,j}$, where $I_{i,j}$ denotes the set of constitutive



$\mathbf{C}^{[1,1]}$
$\mathbf{C}^{[2,1]}$
$\mathbf{C}_0^{[2,2]}$
$\mathbf{C}^{[3,1]}$
$\mathbf{C}^{[3,2]}$
$\mathbf{C}^{[3,3]}$
$\mathbf{C}^{[3,4]}$
$\mathbf{C}_0^{[3,1]}$
$\mathbf{C}_0^{[3,2]}$
$\mathbf{C}_0^{[3,3]}$
$\mathbf{C}_0^{[3,4]}$
$u_j(k_3)=\frac{1}{c_j}\left(\bar{P}_j(k_3)-a_j x_j^{tail}(k_3)+b_j x_j^{head}(k_3)\right)$
S
$\mathbf{S}_1$
$\mathbf{S}_2$
$\mathbf{S}_3$
$\mathbf{S}_4$
$\Delta t_1$
$\Delta t_2$
$\Delta t_3$
$k_1$
$k_1+1$
$k_2$
$k_2+1$
$k_3$
$k_3+1$
$k_3$
$k_3+1$
$k$
$k+1$
$P^{in}(k), x^t(k), \Delta P(k)$
$x_i(k_3), x_i^t(k_3)$
$\hat{P}_i(k_3), \hat{x}_i(k_3)$
$\hat{P}_{des}^{in}(\mathbf{k}_1), \hat{x}_{des}^t(\mathbf{k}_1)$
$u_i(k_3)$
$\hat{x}_{r,i}(k_i), \hat{x}_i^{low}(k_i)$
$\hat{P}_i^{in}(\mathbf{k}_i), \hat{x}_i^t(\mathbf{k}_i), \dot{x}_{des,i}^{up}(\mathbf{k}_i), \hat{P}_{des,i}^{out}(\mathbf{k}_i), \hat{x}_{des,i}^{low}(\mathbf{k}_i)$

**Figure 5.4 Example 3-level hierarchy where $N=3$, $N_{sub}=4$, $\nu_2=\nu_3=3$, $I_{2,1}=\{1,2\}$, $I_{2,2}=\{3,4\}$, and $I_{1,1}=\{1,2,3,4\}$. The notation $x(\mathbf{k}_i)$ refers to a sequence of $x$ values at time steps $\{k_i,\ldots,k_i+N_p^i\}$.**

Level $N$ subsystems. Since all subsystems at Level $N$ are included in the agglomerated subsystems at Level $i$, $\bigcup_{j=1}^{n_i} I_{i,j}=[1,N_{sub}]$. Additionally, a subsystem at Level $N$ can only be included in a single subsystem at Level $i$, thus $l\in I_{i,j}\Rightarrow l\notin I_{i,k}, k\neq j$. The nominal subsystem model $\hat{\mathbf{S}}_0^{[i,j]}$ is used to derive a reduced model $\hat{\mathbf{S}}_0^{r[i,j]}$ to be used by controller $\mathbf{C}^{[i,j]}$. Model reduction reduces the dimension of the state and power flow vectors as follows.

For subsystem $\hat{\mathbf{S}}_0^{[i,j]}$, the state vector is decomposed as

$$\hat{x} = \begin{bmatrix} \hat{x}_r \\ \hat{x}_f \end{bmatrix}, \ \hat{x}_r \in \mathbb{R}^{N_r}, \ \hat{x}_f \in \mathbb{R}^{N_f}. \tag{5.18}$$

In this work, the states of the reduced model are $\hat{x}_r = \left[ \hat{x}_{r,z} \right]$, where

(1) $\hat{x}_{r,z} \in \boldsymbol{x}_i$,

(2) $\hat{x}_{r,z} \in \hat{x}_r$ for $\mathbf{S}^{[i-1,k]}$, $j \in I_{i-1,k}$, and/or

(3) $\hat{x}_{r,z} \in x^{in}$ for $\mathbf{S}^{[i+1,l]}$, $l \in I_{i,j}$.

All remaining states of the nominal subsystem model are denoted $\hat{x}_f$ and are excluded from the reduced model.

**Remark 5.4**

*The reasoning for including these three types of states is as follows. The first set of states aligns with the overall principle of a hierarchical controller: controllers at the $i^{th}$ level determine the desired state trajectories for states with the corresponding timescale. The second set of states is included to achieve coordination between the control levels. The desired state trajectories for these states are determined at Level $i-1$ and are tracked by the controllers at Levels $i$ through $N$. Finally, the third set of states is used to achieve coordination between subsystems at Level $i+1$. Since the states of these vertices affect the power flow exiting neighboring subsystems, coordinating the values of these states is important. The desired trajectories for these states are determined by controllers at Level $i$ and then sent down as a desired state trajectory to be tracked by a controller at Level $i+1$ and as a known sink state disturbance to the controllers for the neighboring subsystems.*

The decomposed subsystem dynamics are

$$\begin{bmatrix} \hat{x}_r^+ \\ \hat{x}_f^+ \end{bmatrix} = \begin{bmatrix} \hat{x}_r \\ \hat{x}_f \end{bmatrix} + \begin{bmatrix} B_r \\ B_f \end{bmatrix} \hat{P} + \begin{bmatrix} V_r \\ V_f \end{bmatrix} \hat{P}^{in}. \tag{5.19}$$

In $\hat{\mathbf{S}}_0^{r[i,j]}$, it is assumed $\hat{x}_f\left(k_i + 1\right) = \hat{x}_f\left(k_i\right)$, resulting in

$$B_f \hat{P}(k_i) + V_f \hat{P}^{in}(k_i) = 0. \tag{5.20}$$

Thus (5.20) provides $N_f$ constraints that can be used to reduce the decision vector $\hat{P} \in \mathbb{R}^{N_e}$ to a reduced decision vector $\hat{P}_r \in \mathbb{R}^{N_e - N_f}$ where

$$\hat{P}(k_i) = T\hat{P}_r(k_i) + Y\hat{P}^{in}(k_i). \tag{5.21}$$

Calculation of $T \in \mathbb{R}^{N_e \times (N_e - N_f)}$ and $Y \in \mathbb{R}^{N_e \times N_s}$ is detailed in the Appendix at the end of this Chapter. The reduced subsystem is

$$\hat{\mathbf{S}}_0^{r[i,j]}: \; \hat{x}_r^+ = \hat{x}_r + \hat{B}_r \hat{P}_r + \hat{V}_r \hat{P}^{in}, \tag{5.22}$$

where $\hat{B}_r = B_r T$ and $\hat{V}_r = V_r + B_r Y$.

**Remark 5.5**

*While the graph-based model reduction approach presented in Section* 3.6 *can be used for developing the generic hierarchical controller in Chapter 3 and visually shows the reduced graphs, this Chapter utilizes the residualization-based approach* [78] *presented above that results in no modeling error between the full and reduced nominal integrator system dynamics.*

## 5.5 Level *N* Controller Formulation

Following a bottom-up control development approach, the subsystem models and controllers for Level $N$ are generated first.

### 5.5.1 Error Subsystem

With the nominal subsystem dynamics from (5.17) and the true subsystem dynamics as

$$\mathbf{S}_0^{[N,j]}: \; x^+ = x + B\bar{P} + V_1 P^{in} + V_3 \Delta P, \tag{5.23}$$

a candidate control law for $\hat{\mathbf{S}}_0^{[N,j]}$ is defined as

$$\mathbf{C}_0^{[N,j]}: \; \bar{P} = \hat{P} + K\left[x - \hat{x}\right]. \tag{5.24}$$

Letting $e(k_N)=x(k_N)-\hat{x}(k_N)$, from (5.17), (5.23), and (5.24), the error dynamic between the true and nominal subsystem is

$$e^+ = (I-BK)e + B\Delta P + V_1\Delta P^{in}. \tag{5.25}$$

Choosing $K=-B^\dagger$, (5.25) reduces to

$$e^+ = B\Delta P + V_1\Delta P^{in}. \tag{5.26}$$

Since $\Delta P(k)\in\Delta\mathcal{P}$ and $\Delta P^{in}(k)\in\Delta\mathcal{P}^{in}$, there exists a robust positively invariant (RPI) set $\mathcal{E}$ such that $e(k)\in\mathcal{E}\,\forall k$. From (5.26), $\mathcal{E}=B\Delta\mathcal{P}\oplus V_1\Delta\mathcal{P}^{in}$. Similarly $\bar{P}(k)-\hat{P}(k)\in K\mathcal{E}\ \forall k$.

**Remark 5.6**

*The candidate control law from* (5.24) *with* $K=-B^\dagger$ *requires the existence of the right inverse of* $B=-\Delta tC^{-1}\bar{M}$. *It is well established that the column sums of the incidence matrix* $M$ *are zero for a connected graph, resulting in linearly* dependent *rows of* $M$. *However, when a subsystem is externally connected, i.e.* $N_t^{[i,j]}>0$, $\underline{M}$ *contains nonzero values and the rows of* $\bar{M}$ *become linearly* independent. *When the rows of a matrix* $A$ *are independent,* $AA^\dagger=I$. *Thus, assuming* $N_t^{[i,j]}>0$ *for all subsystems, a controller with* $K=-B^\dagger$ *is valid and* $(I+BK)=0$.

**Remark 5.7**

*When the* $j^{th}$ *power flow into a subsystem,* $P_j^{in}$, *comes from the environment, i.e. external to the entire system, the bound on* $\Delta P_j^{in}=P_j^{in}-\hat{P}_j^{in}$ *is assumed to be a known property of the system and is used to define* $\Delta\mathcal{P}^{in}$ *for the subsystem. However, when* $P_j^{in}\subset P^{out}$ *of a neighboring subsystem, the bound on* $\Delta P_j^{in}$ *is not immediately known. However, using the acyclic assumption from Assumption* 5.2, *this bound equals the bound on* $\bar{P}-\hat{P}\in K\mathcal{E}$ *for the corresponding edge exiting that neighboring subsystem. Thus calculating* $\Delta\mathcal{P}^{in}$ *for each subsystem is a sequential process starting with the subsystem having inlet power flows exclusively from the environment.*

Using the example from Fig. 5.3, the subsystems $\mathbf{S}_1$ - $\mathbf{S}_4$ can be ordered as $\{\mathbf{S}_1, \mathbf{S}_2, \mathbf{S}_4, \mathbf{S}_3\}$ to calculate the bounds on $\Delta P_j^{in}$ for each subsystem since $\mathbf{S}_1$ only has inlet power flow from the environment, $\mathbf{S}_2$ only has inlet power flow from the environment and $\mathbf{S}_1$, and so on.

## 5.5.2 Constraint Tightening

Using the constraint tightening approach from [62], $\mathbf{C}^{[N,j]}$ constrains $\hat{x}(k_N) = \hat{\mathcal{X}} = \mathcal{X} \ominus \mathcal{E}$. Additionally, using (5.13), the constraint on actuator inputs is imposed as

$$u = \beta^{-1}\left(\bar{P} - M_{a,b}^{T}\begin{bmatrix} x \\ x^t \end{bmatrix}\right) \in \mathcal{U}. \tag{5.27}$$

From comparing (5.13) and (5.15),

$$u - \hat{u} = \beta^{-1}\left(\bar{P} - \hat{P} - M_{a,b}^{T}\begin{bmatrix} e \\ \Delta x^t \end{bmatrix}\right). \tag{5.28}$$

Due to the difference between the desired power flow $\bar{P}$ and the nominal power flow $\hat{P}$, where $\bar{P}(k_N) - \hat{P}(k_N) \in K\mathcal{E}$, $\mathbf{C}^{[N,j]}$ imposes tightened input constraints as

$$\hat{u} = \beta^{-1}\left(\hat{P} - M_{a,b}^{T}\begin{bmatrix} \hat{x} \\ \hat{x}^t \end{bmatrix}\right) \in \hat{\mathcal{U}}, \tag{5.29}$$

where

$$\hat{\mathcal{U}} = \mathcal{U} \ominus \beta^{-1}K\mathcal{E} \ominus \left(-\beta^{-1}\bar{M}_{a,b}^{T}\mathcal{E}\right) \ominus \left(-\beta^{-1}\underline{M}_{a,b}^{T}\Delta\mathcal{X}^t\right). \tag{5.30}$$

Note that when $\Delta x^t$ corresponds to a sink of the overall system, $\Delta\mathcal{X}^t$ is defined. However, when $\Delta x^t$ corresponds to a state of a neighboring subsystem, $\Delta\mathcal{X}^t$ is not directly known but can be calculated based on the error set $\mathcal{E}$ for that neighboring subsystem.

## 5.5.3 Level *N* MPC Problem

Each controller $\mathbf{C}^{[N,j]}$ solves the optimization problem $\mathbb{P}^{[N,j]}(k_N)$:

$$\min_{\hat{P}\left(k_N : k_N + N_p^N - 1\right)} J^{[N,j]}(k_N) \tag{5.31a}$$

subject to, for $h = \left[k_N : k_N + N_p^N - 1\right]$,

$$\hat{x}(h+1) = \hat{x}(h) + B\hat{P}(h) + V_1 \hat{P}^{in}(h), \tag{5.31b}$$

$$\hat{x}(h) \in \hat{\mathcal{X}}, \tag{5.31c}$$

$$\beta^{-1}\left(\hat{P}(h) - M_{a,b}^T \begin{bmatrix} \hat{x}(h) \\ \hat{x}^t(h) \end{bmatrix}\right) \in \hat{\mathcal{U}}, \tag{5.31d}$$

$$Z^{up} C^{-1}\left[-\bar{M}\hat{P}(h) + D\hat{P}^{in}(h)\right] = \dot{x}_{des}^{up}(h), \tag{5.31e}$$

$$\hat{P}^{out}(h) = \hat{P}_{des}^{out}(h), \tag{5.31f}$$

$$\hat{x}^{low}(h) - \hat{x}_{des}^{low}(k_N) \in \Delta\mathcal{X}_{low}, \tag{5.31g}$$

where $\hat{P}^{in}(h)$, $\hat{x}^t(h)$, $\dot{x}_{des}^{up}(h)$, $\hat{P}_{des}^{out}(h)$, and $\hat{x}_{des}^{low}(k_N)$ are communicated from controllers at Level $N-1$.

**Remark 5.8**

*Constraint* (5.31e) *uses the matrix* $Z^{up}$ *to constrain the power flows into and out of the vertices with states included in a controller at Level* $N-1$. *Constraint* (5.31f) *ensures that the power flows exiting a subsystem equal the power flows determined by the upper-level controller. Finally, constraint* (5.31g) *bounds the deviation between the trajectories of states not included in upper-level controllers and the assumed value* $\hat{x}_{des}^{low}$ *used by those upper-level controllers. Note that* $\hat{x}_{des}^{low}$ *is constant over the prediction horizon. The box constraint set* $\Delta\mathcal{X}_{low}$ *includes the* *origin and the size of the set determines the trade-off between conservativeness of upper-level controller and the freedom of lower-level controllers.*

## 5.6 Level *i* Controller Formulation (*i* ≠ *N*)

### 5.6.1 Constraint Tightening

No additional state constraint tightening is required and thus $\mathbf{C}^{[i,j]}, i \neq N$, constrain $\hat{x}_r(k_i) \in \hat{\mathcal{X}}_r$ where $\hat{\mathcal{X}}_r$ is formed directly from the box constraints of $\hat{\mathcal{X}}$.

The input constraints *do* need additional tightening to ensure robustness to $\hat{x}^{low}(k_i) - \hat{x}^{low}_{des}(k_i)$. Using the state decomposition from (5.18) and adding and subtracting $\hat{x}^{low}_{des}(k_i)$, (5.29) becomes

$$\beta^{-1}\left(\hat{P}(k_i) - M_{a,b}^T \begin{bmatrix} \hat{x}_r(k_i) \\ \hat{x}^{low}_{des}(k_i) \\ \hat{x}^t(k_i) \end{bmatrix}\right) - M_f^T\left[\hat{x}_f(k_i) - \hat{x}^{low}_{des}(k_i)\right] \in \hat{\mathcal{U}}, \tag{5.32}$$

where $M_f$ is the portion of $M_{a,b}$ corresponding to the fast states. Since (5.31g) bounds the difference between $\hat{x}_f$ and $\hat{x}^{low}_{des}$, the input constraints are further tightened such that

$$\beta^{-1}\left(\hat{P}(k_i) - M_{a,b}^T \begin{bmatrix} \hat{x}_r(k_i) \\ \hat{x}^{low}_{des}(k_i) \\ \hat{x}^t(k_i) \end{bmatrix}\right) \in \hat{\mathcal{U}}_r, \tag{5.33}$$

where $\hat{\mathcal{U}}_r = \hat{\mathcal{U}} \ominus \left(-M_f \Delta\tilde{\mathcal{X}}_{low}\right)$. Note that $\Delta\tilde{\mathcal{X}}_{low}$ is a set similar to $\Delta\mathcal{X}_{low}$ from (5.31g) but the box constraints are twice the magnitude of those in $\Delta\mathcal{X}_{low}$. This is due to the following. The Level 1 controller is constrained such that $\hat{x}_0^{low}(k_1) - \hat{x}^{low}_{des}(k_1) \in \Delta\mathcal{X}_{low}$, where $\hat{x}^{low}$ is the current values of $\hat{x}_f$, the states not included in the reduced model for the Level 1 controller, and $\hat{x}^{low}_{des}$ are the values of $\hat{x}_f$ chosen by the controller that remain constant over the prediction horizon.

The values $\hat{x}_{des}^{low}$ are communicated down the hierarchy, where each controller at Levels 2 through $N$ is constrained such that $\hat{x}^{low}(k_i) - \hat{x}_{des}^{low}(k_i) \in \Delta\mathcal{X}_{low}$ and $\hat{x}^{low}$ are the trajectories of the states in the lower-level controller model that were approximated by $\hat{x}_{des}^{low}$ for the controller at Level 1. This two-step process for constraining these fast state trajectories requires the constraint tightening using $\Delta\tilde{\mathcal{X}}_{low}$, instead of $\Delta\mathcal{X}_{low}$, as defined above.

To ensure feasibility using the control law from (5.27) it is necessary to impose the additional constraint

$$\beta^{-1}\left(\hat{P}(k_i) - M_{a,b}^{T}\begin{bmatrix}\hat{x}_r(k_i+1)\\ \hat{x}_{des}^{low}(k_i)\\ \hat{x}^{t}(k_i+1)\end{bmatrix}\right) \in \hat{\mathcal{U}}_r, \tag{5.34}$$

so that, as the states evolve between time steps, there is always an input $\hat{u}(k_N) \in \hat{\mathcal{U}}$ to achieve the desired power flow $\hat{P}(k_i)$ at the faster time steps $k_N$ between the slower time steps $k_i$ and $k_i+1$. Note that $\hat{x}_{des}^{low}(k_i) = \hat{x}_{des}^{low}(k_i+1)$ since $\hat{x}_{des}^{low}$ is assumed constant over the prediction horizon.

### 5.6.2 Level *i* MPC Problem (*i* ≠ *N*, *i* ≠ 1)

Each controller $\mathbf{C}^{[i,j]}$, $i \in [2, N-1]$, solves the optimization problem $\mathbb{P}^{[i,j]}(k_i)$:

$$\min_{\hat{P}_r\left(k_i:k_i+N_p^i-1\right)} J^{[i,j]}(k_i) \tag{5.35a}$$

subject to, for $h = \left[k_i : k_i + N_p^i - 1\right]$,

$$\hat{x}_r(h+1) = \hat{x}_r(h) + \hat{B}_r\hat{P}_r(h) + \hat{V}_r\hat{P}^{in}(h), \tag{5.35b}$$

$$\hat{x}_r(h) \in \hat{\mathcal{X}}_r, \tag{5.35c}$$

$$\hat{P}(h) = T\hat{P}_r(h) + Y\hat{P}^{in}(h), \tag{5.35d}$$

$$\beta^{-1}\left(\hat{P}(h)-M_{a,b}^{T}\begin{bmatrix}\hat{x}_r(h)\\ \hat{x}_{des}^{low}(k_i)\\ \hat{x}^{t}(h)\end{bmatrix}\right)\in\hat{\mathcal{U}}_r, \tag{5.35e}$$

$$\beta^{-1}\left(\hat{P}(h)-M_{a,b}^{T}\begin{bmatrix}\hat{x}_r(h+1)\\ \hat{x}_{des}^{low}(k_i)\\ \hat{x}^{t}(h+1)\end{bmatrix}\right)\in\hat{\mathcal{U}}_r, \tag{5.35f}$$

$$Z^{up}C^{-1}\left[-\bar{M}\hat{P}(h)+D\hat{P}^{in}(h)\right]=\dot{x}_{des}^{up}(h), \tag{5.35g}$$

$$\hat{P}^{out}(h)=\hat{P}_{des}^{out}(h), \tag{5.35h}$$

$$\hat{x}^{low}(h)-\hat{x}_{des}^{low}(k_i)\in\Delta\mathcal{X}_{low}, \tag{5.35i}$$

where $\hat{P}^{in}(h)$, $\hat{x}^{t}(h)$, $\dot{x}_{des}^{up}(h)$, $\hat{P}_{des}^{out}(h)$, and $\hat{x}_{des}^{low}(k_i)$ are communicated from controllers at Level $i-1$.

### 5.6.3 Level 1 MPC Problem

The single controller $\mathbf{C}^{[1,1]}$ solves the optimization problem $\mathbb{P}^{[1,1]}(k_1)$:

$$\min_{\substack{\hat{P}_r\left(k_1:k_1+N_p^1-1\right),\hat{x}_{des}^{low}(k_1)\\ \Lambda^{s}\left(k_1:k_1+N_p^1-1\right),\Lambda^{t}\left(k_1:k_1+N_p^1\right)}} J^{[1,1]}(k_1) \tag{5.36a}$$

subject to, for $h=\left[k_1:k_1+N_p^1-1\right]$,

$$\hat{x}_r(h+1)=\hat{x}_r(h)+\hat{B}_r\hat{P}_r(h)+\hat{V}_r\hat{P}^{in}(h), \tag{5.36b}$$

$$\hat{x}_r(h)\in\hat{\mathcal{X}}_r, \tag{5.36c}$$

$$\hat{P}(h)=T\hat{P}_r(h)+Y\hat{P}^{in}(h), \tag{5.36d}$$

$$\beta^{-1}\left(\hat{P}(h)-M_{a,b}^{T}\begin{bmatrix}\hat{x}_r(h)\\ \hat{x}_{des}^{low}(k_1)\\ \hat{x}^t(h)\end{bmatrix}\right)\in\hat{\mathcal{U}}_r, \tag{5.36e}$$

$$\beta^{-1}\left(\hat{P}(h)-M_{a,b}^{T}\begin{bmatrix}\hat{x}_r(h+1)\\ \hat{x}_{des}^{low}(k_1)\\ \hat{x}^t(h+1)\end{bmatrix}\right)\in\hat{\mathcal{U}}_r, \tag{5.36f}$$

$$\hat{P}^{in}(h)=\Lambda^s(h)\hat{P}_{des}^{in}(h),\quad \hat{x}^t(h)=\Lambda^t(h)\hat{x}_{des}^t(h), \tag{5.36g}$$

$$\hat{x}_0^{low}-\hat{x}_{des}^{low}(k_1)\in\Delta\mathcal{X}_{low}, \tag{5.36h}$$

$$\hat{x}_r\left(k_1+N_p^1\right)=\hat{x}_r\left(k_1+N_p^1-1\right), \tag{5.36i}$$

where $\hat{P}_{des}^{in}(h)$ and $\hat{x}_{des}^t(h)$ are provided directly to the controller and $\hat{x}_0^{low}$ is the current value of $\hat{x}^{low}$ communicated up the hierarchy from the controllers at Level $N$.

**Assumption 5.3**

*The desired nominal disturbances $\hat{P}_{des}^{in}$ and $\hat{x}_{des}^t$ are known over the entire prediction horizon of the hierarchical controller and are piecewise constant between updates of the controller at the highest level of the hierarchy.*

The controller has the ability to augment these values in order to maintain feasibility. Thus, from (5.36g), the nominal values are $\hat{P}^{in}(k)=\Lambda^s(k)\hat{P}_{des}^{in}(k)$ and $\hat{x}^t(k)=\Lambda^t(k)\hat{x}_{des}^t(k)$ where $\Lambda^s=diag\left(\left[\lambda_i^s\right]\right)$ and $\Lambda^t=diag\left(\left[\lambda_i^t\right]\right)$ are diagonal matrices with $\lambda_i^t,\lambda_i^s\in\mathbb{R}$.

**Remark 5.9**

*While this is a strong, and possibly limiting, assumption, the proposed framework can be readily extended to be robust to intersample changes in disturbances through additional constraint tightening.*

**Remark 5.10**

*Constraint* (5.36i) *helps with the proof of feasibility presented in Section 5.7 and guarantees that* $\Lambda^s = diag\left(\left[\lambda_i^s\right]\right)$ *and* $\Lambda^t = diag\left(\left[\lambda_i^t\right]\right)$ *are chosen such that there is a feasible solution that allows all states to remain constant at the end of the prediction horizon.*

## 5.7 Recursive Feasibility

**Assumption 5.4**

*There exists a neighborhood containing the origin,* $\mathcal{X}_0 \supset \{0\}$, *such that if* $x(k) \in \mathcal{X}_0$ *then* $\mathbb{P}^{[1,1]}(k_1)$ *is feasible with optimal solution*

$$\hat{\mathbf{P}}_r^*(k_1) = \left\{\hat{P}_r^*(k_1), \ldots, \hat{P}_r^*\left(k_1 + N_p^1 - 1\right)\right\} \tag{5.37}$$

*and associated nominal state trajectory*

$$\hat{\mathbf{x}}_r^*(k_1) = \left\{\hat{x}_r^*(k_1), \ldots, \hat{x}_r^*\left(k_1 + N_p^1\right)\right\}. \tag{5.38}$$

**Theorem 5.1**

*If* $x(k) \in \mathcal{X}_0$, *then the solution*

$$\hat{\mathbf{P}}_r^*(k_1+1) = \left\{\hat{P}_r^*(k_1+1), \ldots, \hat{P}_r^*\left(k_1 + N_p^1 - 1\right), \hat{P}_r^*\left(k_1 + N_p^1 - 1\right)\right\} \tag{5.39}$$

*with*

$$\hat{\mathbf{x}}_r^*(k_1+1) = \left\{\hat{x}_r^*(k_1+1), \ldots, \hat{x}_r^*\left(k_1 + N_p^1\right), \hat{x}_r^*\left(k_1 + N_p^1\right)\right\} \tag{5.40}$$

*is feasible for* $\mathbb{P}^{[1,1]}(k_1+1)$. *Furthermore, the feasibility of* $\mathbb{P}^{[1,1]}(k_1)$ *guarantees the feasibility of* $\mathbb{P}^{[i,j]}(k_i)$ *for any lower-level controller.*

**Proof.**

The proof is outlined as follows. First, it is established that the states and power flows of the system remain close to the nominal states and power flows determined by the hierarchical controller. This allows the hierarchy to only require knowledge of the nominal states and power flows; the true system state is not used by any MPC controller in the hierarchy. The feasibility of lower-level controllers follows directly from the feasibility of $\mathbb{P}^{[1,1]}(k_1)$. Feasibility of $\mathbb{P}^{[1,1]}(k_1+1)$ follows from the constraints imposed by each lower-level controller and the existence of a solution where all states remain constant at the end of the prediction horizon.

By implementing the feedback integralization control law (5.13) and the candidate control law (5.24), the error dynamic between the true and nominal states follows (5.26). Thus at any timestep, $e(k_N)\in\mathcal{E}$ and $\bar{P}(k_N)-\hat{P}(k_N)\in K\mathcal{E}$ and the proposed constraint tightening from Section 5.5.2 guarantees that $\hat{x}(k_N)\in\hat{\mathcal{X}}\Rightarrow x(k)\in\mathcal{X}$ and $\hat{u}(k_N)\in\hat{\mathcal{U}}\Rightarrow u(k)\in\mathcal{U}$.

From Assumption 5.4, there exists $x(k)\in\mathcal{X}_0$ such that $\mathbb{P}^{[1,1]}(k_1)$ is feasible. The optimal *reduced* power flow solution $\hat{\mathbf{P}}_r^*(k_1)$ is related to the optimal *unreduced* power flow solution $\hat{\mathbf{P}}^*(k_1)$ via (5.21). From the construction of the reduced state and input constraints, $\hat{x}_r(k_1)\in\hat{\mathcal{X}}_r\Rightarrow\hat{x}_r(k_1)\in\hat{\mathcal{X}}$ and $\hat{u}(k_1)\in\hat{\mathcal{U}}_r\Rightarrow\hat{u}(k_1)\in\hat{\mathcal{U}}$, and thus, if $\hat{\mathbf{P}}^*(k_1)$ is feasible for $\mathbb{P}^{[1,1]}(k_1)$, these power flows form a feasible solution for all $\mathbf{C}^{[i,j]}$, $i\in[2,N]$. This solution provides perfect tracking of $\hat{x}_r$ and constant trajectories for $\hat{x}_f$ of $\hat{\mathbf{S}}_0^{[1,1]}$.

If all controllers $\mathbf{C}^{[i,j]}$, $i\in[2,N]$ are feasible, the desired state trajectory $\hat{\mathbf{x}}_r^*(k_1)$ for $\mathbf{C}^{[1,1]}$ is tracked perfectly based on the constraints (5.31e) and (5.35g). Thus $\hat{x}_r^*(k_1+1\,|\,k_1+1)=\hat{x}_r^*(k_1+1\,|\,k_1)$. If $\hat{\mathbf{x}}_r^*(k_1)$ is a feasible solution for $\mathbf{C}^{[1,1]}$, then $\hat{x}_r^*\left(k_1+N_p^1\right)=\hat{x}_r^*\left(k_1+N_p^1-1\right)$ from constraint (5.36i). This implies that $\hat{B}_r\hat{P}_r^*\left(k_1+N_p^1-1\right)+\hat{V}_r\hat{P}^{in}\left(k_1+N_p^1-1\right)=0$. Since $\hat{\mathbf{x}}_r^*(k_1+1)$ assumes

$\hat{x}_r^*\left(k_1+N_p^1+1\right)=\hat{x}_r^*\left(k_1+N_p^1\right)$, it is desired that $\hat{B}_r\hat{P}_r^*\left(k_1+N_p^1\right)+\hat{V}_r\hat{P}^{in}\left(k_1+N_p^1\right)=0$ which is always feasible with $\hat{P}_r^*\left(k_1+N_p^1\right)=\hat{P}_r^*\left(k_1+N_p^1-1\right)$ and use of $\Lambda^s\left(k_1+N_p^1\right)$.

Additionally, if all controllers $\mathbf{C}^{[i,j]}, i\in[2,N]$ are feasible, then $\hat{x}^{low}(k_1+1)-\hat{x}_{des}^{low}(k_1)\in\Delta\mathcal{X}_{low}$. Thus, a feasible solution at time step $k_1+1$ is $\hat{x}_{des}^{low}(k_1+1)=\hat{x}_{des}^{low}(k_1)$. Since $\hat{x}_r^*\left(k_1+N_p^1+1\right)=\hat{x}_r^*\left(k_1+N_p^1\right)$, $\hat{P}_r^*\left(k_1+N_p^1\right)=\hat{P}_r^*\left(k_1+N_p^1-1\right)$, and $\hat{x}_{des}^{low}(k_1+1)=\hat{x}_{des}^{low}(k_1)$, feasibility of $\mathbb{P}^{[1,1]}(k_1+1)$ only depends on the feasibility of (5.36f) with regard to $\hat{x}^t\left(k_1+N_p^1+1\right)$. Using $\Lambda^t\left(k_1+N_p^1+1\right)$, (5.36f) is always feasible. Thus, with $\mathbb{P}^{[1,1]}(k_1)$ feasible, all lower-level controllers are feasible and $\mathbb{P}^{[1,1]}(k_1+1)$ is feasible, proving the theorem. □

## 5.8 Numerical Example

The efficacy of the proposed robust hierarchical control framework is demonstrated with the following numerical example that, as shown in Fig. 5.5, has the same structure as the example systems from Chapters 3 and 4 but is not intented to represent a particular physical system. The vehicle consists of two systems, each with two subsystems, and thus, the control hierarchy has the same general structure as those shown in Figs. 5.4, 3.5, and 4.5. The individual subsystem and system graphs are shown in Figs. 5.6 and 5.7. The Matlab/Simulink code used to implement the robust control hierarchy for this example system is provided in the Appendix.

For each edge $e_j\in E$, the parameters defining the power flow relationship from (5.2) are $a_j=b_j=c_j=1$. The vertex capacitances from (5.1) are $C_1=C_2=1000$, $C_3=...=C_7=100$, and $C_8=...=\mathrm{C}_{12}=10$. The state and input constraints from (5.10) are defined such that $-1\le x_i\le 1, \forall v_i\in V$ and $-1\le u_j\le 1, \forall e_j\in E$. The disturbance power flow from (5.2) is defined with $\Delta P_j^{\max}=0.1$. The input power flow disturbance set $\Delta\mathcal{P}^{in}$ is defined with $-0.1\le\Delta P_i^{in}=P_i^{in}-\hat{P}_i^{in}\le 0.1$ and the sink state disturbance set $\Delta\mathcal{X}^t$ is defined with

$-0.1 \le \Delta x^t = x^t - \hat{x}^t \le 0.1$. Finally, for the additional constraint tightening in (5.33), $\Delta\tilde{\mathcal{X}}_{low}$ is defined with $-0.1 \le \hat{x}_0^{low} - \hat{x}_{des}^{low} \le 0.1$, and thus $\Delta\mathcal{X}_{low}$ is defined with $-0.05 \le \hat{x}_0^{low} - \hat{x}_{des}^{low} \le 0.05$.

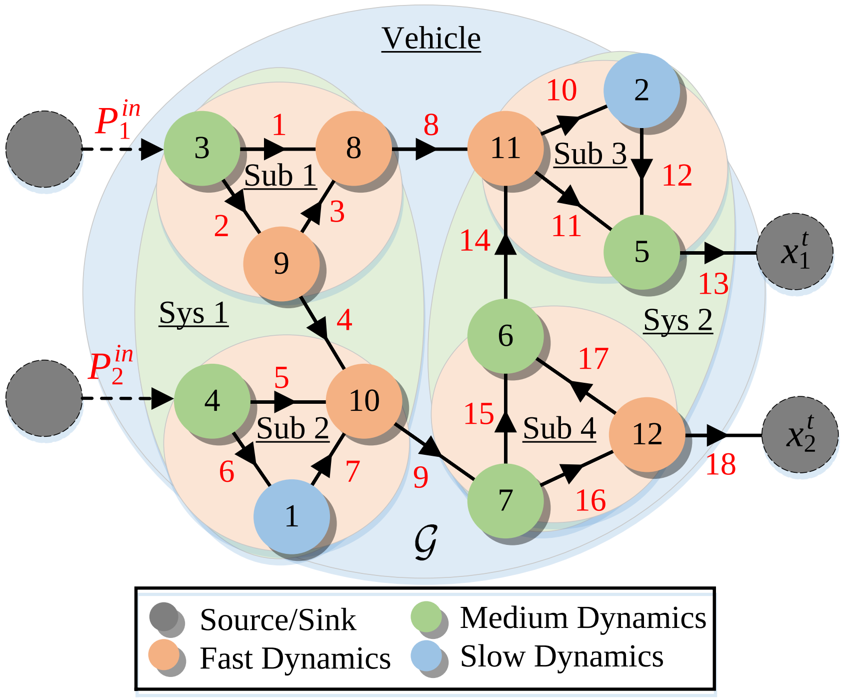


**Figure 5.5 Example system graph for numerical example.**

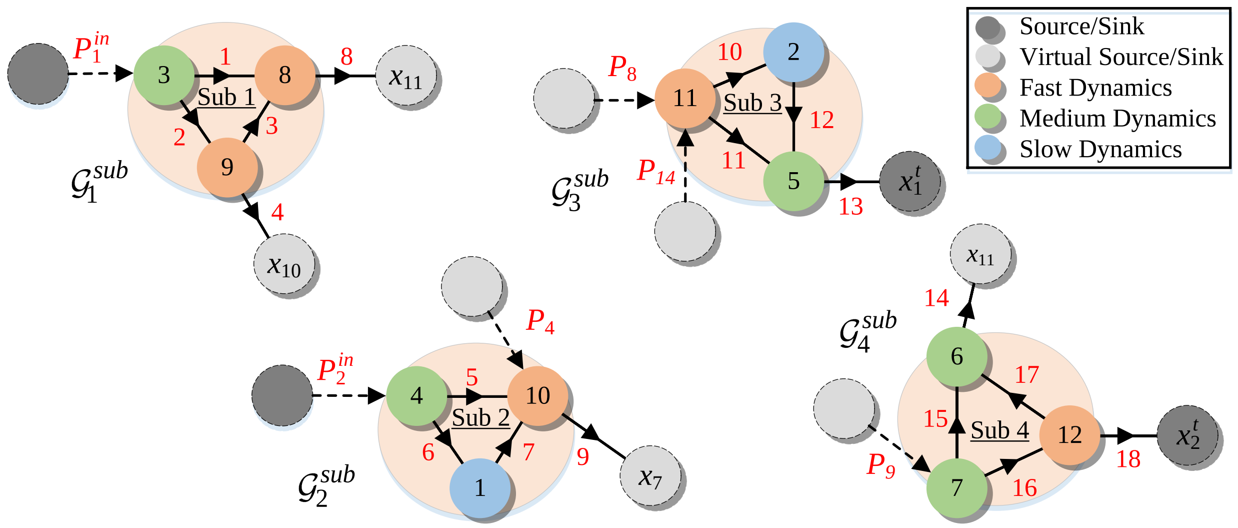


**Figure 5.6 Subsystem graphs for numerical example.**

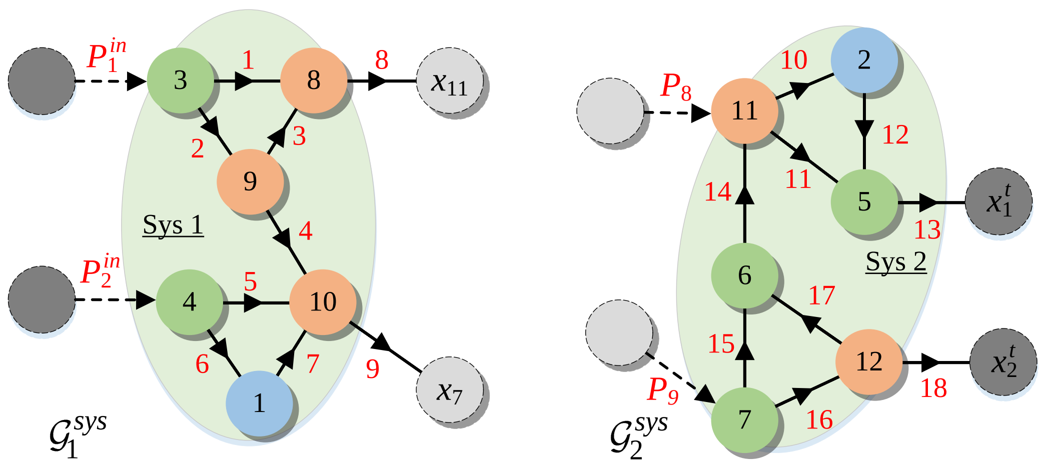


**Figure 5.7 System graphs for numerical example.**

The controller time steps are chosen to be $\Delta t_1 = 100$, $\Delta t_2 = 10$, and $\Delta t_2 = \Delta t = 1$ with prediction horizons $N_p^1 = N_p^2 = N_p^3 = 5$. The cost function from (5.12) is defined with

$$\ell\left(x(k),u(k),\Lambda^s(k),\Lambda^t(k)\right) = \left\|u(k) - \underline{u}(k)\right\|_2^2 + 10^6\left\|\Lambda^s(k) - I\right\|_2^2 + 10^6\left\|\Lambda^t(k) - I\right\|_2^2. \quad (5.41)$$

For the model reduction from (5.18), the state decompositions for System 1, $\hat{\mathbf{S}}_0^{[2,1]}$, is

$$\hat{x}_r = \begin{bmatrix} x_1 & x_3 & x_4 & x_{10} \end{bmatrix}^T, \; \hat{x}_f = \begin{bmatrix} x_8 & x_9 \end{bmatrix}^T, \quad (5.42)$$

for System 2, $\hat{\mathbf{S}}_0^{[2,2]}$, is

$$\hat{x}_r = \begin{bmatrix} x_2 & x_5 & x_6 & x_7 & x_{10} \end{bmatrix}^T, \; \hat{x}_f = \begin{bmatrix} 12 \end{bmatrix}, \quad (5.43)$$

and for the Vehicle, $\hat{\mathbf{S}}_0^{[1,1]}$, is

$$\hat{x}_r = \begin{bmatrix} x_1 & x_2 & x_7 & x_{11} \end{bmatrix}^T, \; \hat{x}_f = \begin{bmatrix} x_3 & x_4 & x_5 & x_6 & x_8 & x_9 & x_{10} & x_{12} \end{bmatrix}^T. \quad (5.44)$$

All set computation is performed with the Multi-Parametric Toolbox 3.0 [79]. The MPC optimization problems are formulated using the YALMIP Toolbox [41] and solved using the Gurobi optimization suite [42].

Fig. 5.8 shows a simple disturbance profile for $\hat{P}_{des}^{in}$ used to demonstrate some of the key features of the robust hierarchical controller. The actual "throttled" inlet power flows $\hat{P}^{in}$ are also shown in Fig. 5.8 based on the maximum feasible inlet power flow determined by $\mathbf{C}^{[1,1]}$ using $\Lambda^s(k_1)$ in (5.36g). Note, that the sink states are held at $\hat{x}_{des}^t = \hat{x}^t = 0$. Uniformly distributed random signals of maximum amplitude are applied for $\Delta P$, $\Delta P^{in}$, and $\Delta x^t$.

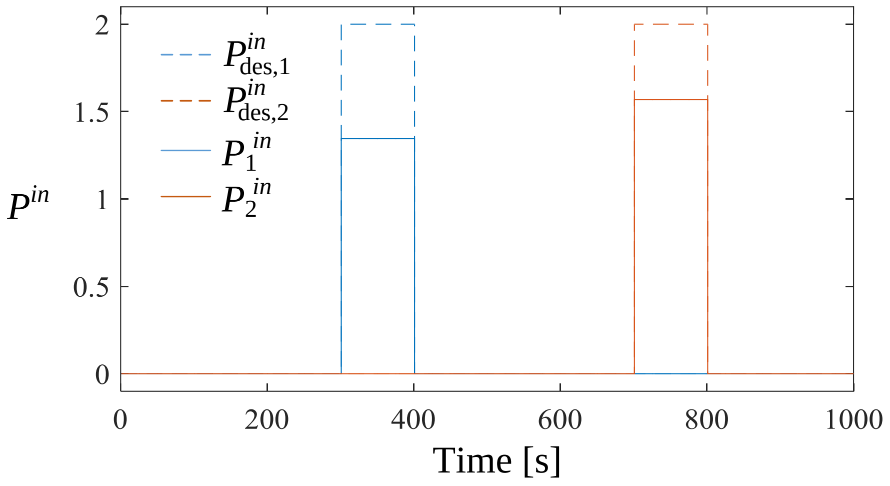


**Figure 5.8 Disturbance profile for numerical example with desired and actual inlet power flows.**

The following figures show the robustness properties of the hierarchical controller. Fig. 5.9 demonstrates the constraint tightening for actuator inputs $u_8$ and $u_{16}$. The actual inputs are bounded by $-1 \le u_i \le 1$ which form the input constraint set $\mathcal{U}$. The notation $\mathcal{U}_{8,16}$ is used to denote the projection of $\mathcal{U}$ into the 8th and 16th coordinates. The tightened inputs used by subsystem controllers at Level 3 are shown by $\hat{\mathcal{U}}_{8,16}$ and the further tighten inputs used by the vehicle controller at Level 1 are shown by $\hat{\mathcal{U}}_{r,8,16}$. Note that the nominal inputs must satisfy $\hat{u} \in \hat{\mathcal{U}}$ but the subsystem controller are capable of inputs such that $\hat{u} \notin \hat{U}_r$. This is a result of the additional constraint tightening employed but the controllers at Levels 1 through $N-1$. Based on (5.28), the difference between $\hat{u}$ and $u$ is bounded and based on the constraint tightening, if $\hat{u} \in \hat{\mathcal{U}}$ then $u \in \mathcal{U}$, as shown in the figure.

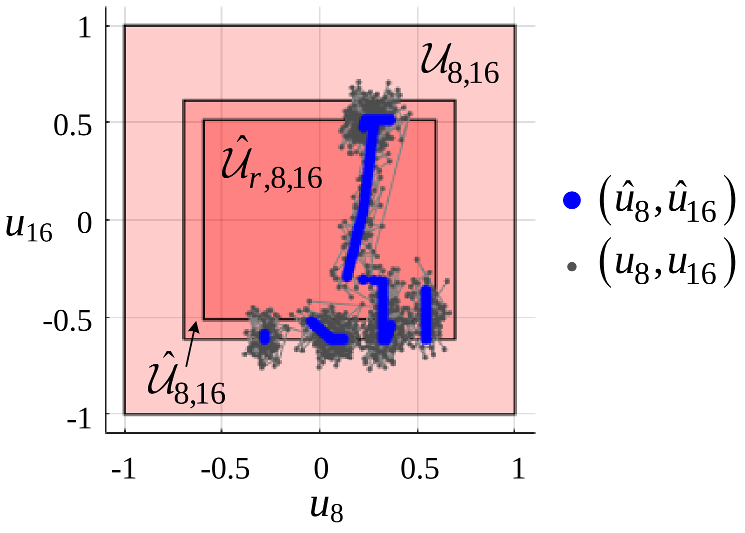


**Figure 5.9 Nominal and actual inputs for edges $e_8$ and $e_{16}$ with the nominal input constraint sets $\mathcal{U}_{8,16}$, the tightened input constraint set $\hat{\mathcal{U}}_{8,16}$ used by the subsystem controllers at Level 3 and, and the tighten input constraint set $\hat{\mathcal{U}}_{r,8,16}$ used by the vehicle controller at Level 1.**

Fig. 5.10 demonstrates the bounded difference between the nominal state trajectory $\hat{x}_{11}$ and the actual state trajectory $x_{11}$ for vertex $v_{11}$. Based on the error dynamic from (5.26), $x_{11} \in \hat{x}_{11} \oplus \mathcal{E}_{11}$, where $\mathcal{E}_{11}$ is the projection of the error set $\mathcal{E}$ onto the 11th coordinate, as shown in the figure.

Similarly, Fig. 5.11 demonstrates the bounded difference between the nominal power flow trajectory $\hat{P}_4$ and the desired power flow trajectory $\bar{P}_4$ along edge $e_4$. Based on the error dynamic from (5.26) using the candidate control law from (5.24), $\bar{P}_4 \in \hat{P}_4 \oplus (K\mathcal{E})_4$, where $K$ and $\mathcal{E}$ are the candidate controller and error sets for Subsystem 1, as shown in the figure.

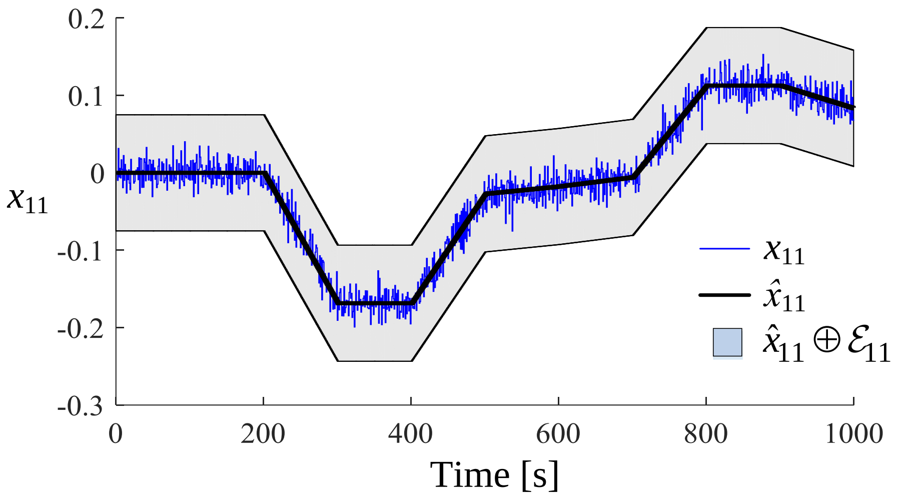


**Figure 5.10 Nominal and actual state trajectories for vertex $v_{11}$ with error set showing the bounds on the deviation between $x_{11}$ and $\hat{x}_{11}$ for which the hierarchical controller is robust.**

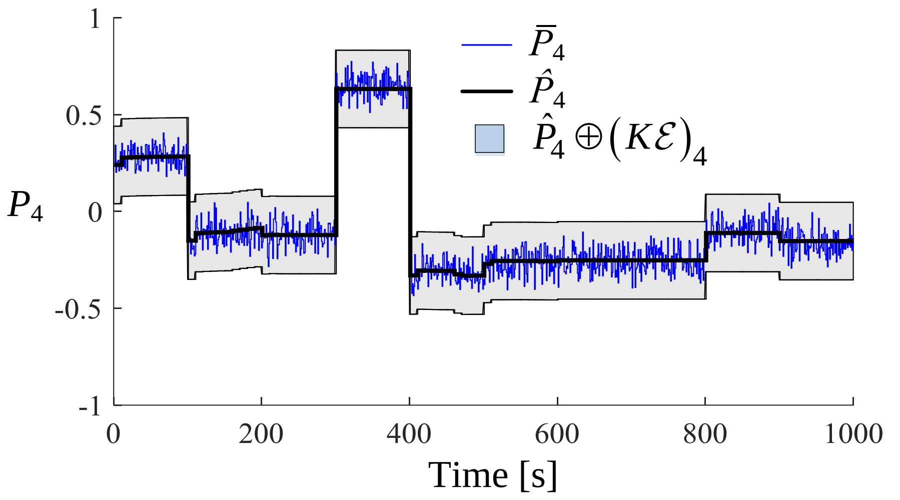


**Figure 5.11 Nominal and desired power flow trajectories along edge $e_4$ with error set showing the bounds on the deviation between $\bar{P}_4$ and $\hat{P}_4$ for which the hierarchical controller is robust.**

The bounded difference between the nominal and desired power flows $\hat{P}$ and $\bar{P}$ is used to formulate the input power uncertainty set $\Delta\mathcal{P}^{in}$ for subsystems that have inlet power flows from neighboring subsystems. Fig. 5.12 demonstrates the formation of $\Delta\mathcal{P}^{in}$ for Subsystem 3 based on the differences $\bar{P}_8 - \hat{P}_8$ and $\bar{P}_{14} - \hat{P}_{14}$ for edges $e_8$ and $e_{14}$.

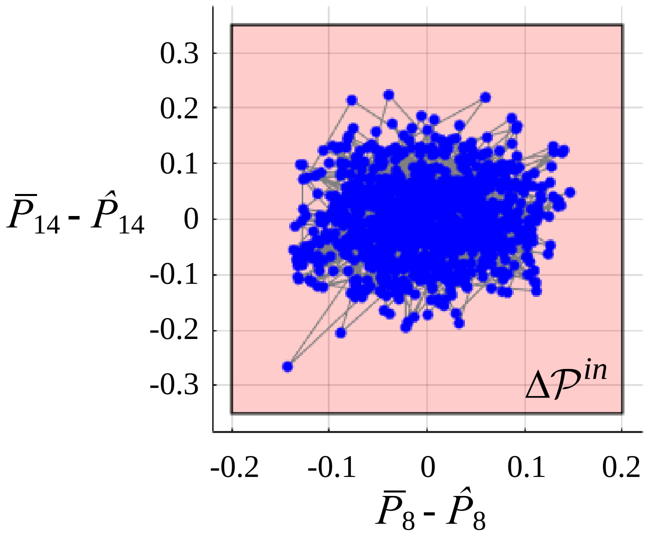


**Figure 5.12 Deviations between the desired and nominal power flows for edges $e_8$ and $e_{14}$, which form $\Delta P^{in}$ for Subsystem 3, with the set $\Delta\mathcal{P}^{in}$ for Subsystem 3.**

Finally, Fig. 5.13 shows the nominal and actual state trajectories for vertex $e_{10}$. Based on the constraint (5.31g), the nominal trajectory $\hat{x}_{10}$, determined by the Subsystem 2 controller at Level 3, must remain close to the desired value $\hat{x}_{des,10}^{low}$ determined by the Vehicle controller at Level 1. By satisfying $\hat{x}_{10} \in \hat{x}_{des,10}^{low} \oplus \Delta\mathcal{X}_{low,10}$, the Subsystem controller is able to optimize the trajectory of $\hat{x}_{10}$ with respect to its own local cost function within a neighborhood of the assumed trajectory determined by the Vehicle controller.

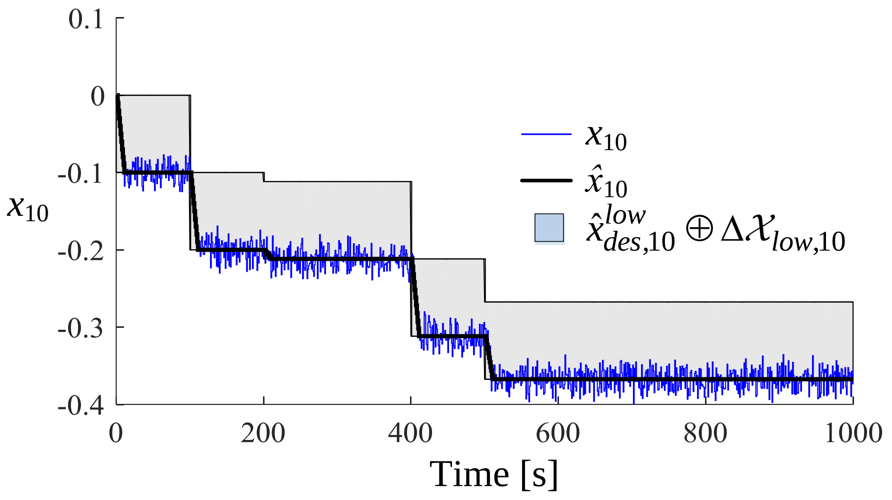


**Figure 5.13 Nominal and actual state trajectories for vertex $v_{10}$ with error set showing the bounds on the deviation between $\hat{x}_{10}$ and $\hat{x}_{des,10}^{low}$ determined by the Vehicle controller at Level 1.**

## 5.9 Chapter Summary

This Chapter presented a generic $N$-level hierarchical control framework based on MPC and a graph-based model of a power flow system that was proven to be robustly feasible at each level of the hierarchy in the presence of model and signal uncertainty. The novel approach utilizes a constraint tightening procedure where all tightened constraints are simple and numerically efficient to calculate offline and implement online. This concludes the theoretical contributions of this dissertation and the remaining Chapters assess the practicality of hierarchical control through graph-based modeling and control of an experimental thermal fluid system.

## 5.10 Chapter Appendix

The following is used to calculate the matrices $T \in \mathbb{R}^{N_e \times (N_e - N_f)}$ and $Y \in \mathbb{R}^{N_e \times N_s}$ for the model reduction from (5.21). Let $R \in \mathbb{R}^{N_f \times (N_e + N_s)}$ be the reduced row echelon form of $A = \begin{bmatrix} B_f & V_f \end{bmatrix}$ using Gauss-Jordan elimination. Let $b = \{b_j\}$, $j \in [1, N_f]$ be a set of indices such that $A(:,b)$ is a basis of the range of $A$. Note that $A(:,b)$ denotes a matrix with columns of $A$ indexed by $b$. Let $E_r = \left\{e_j\right\},\ j \in \left[1, N_e - N_f\right]$ be the set of edges corresponding to the reduced power flow vector $\hat{P}_r$ where $E_r = E \setminus b = \{e_j : e_j \in E \text{ and } e_j \notin b\}$. Let $R' = -R(:,E_r)$. Then, $T = \left[t_{i,j}\right]$ where

$$t_{i,j} = \left\{ \begin{array}{cc} 1 & \text{if } i = e_j,\ e_j \in E_r \\ R'(k,j) & \text{if } i = b_k,\ b_k \in b \\ 0 & \text{else} \end{array} \right\}. \tag{5.45}$$

Let $R^s = R\left(:,\left[N_e + 1 : N_e + N_s\right]\right)$. Then, $Y = \left[y_{i,j}\right]$ where

$$y_{i,j} = \left\{ \begin{array}{cc} R^s(k,j) & \text{if } i = b_k,\ b_k \in b \\ 0 & \text{else} \end{array} \right\}. \tag{5.46}$$

# Chapter 6

# Graph-based Modeling of a Thermal Fluid System

## 6.1 Motivation

From the generic graph-based modeling framework presented in Chapter 2, the objective of this Chapter is to demonstrate the value and applicability of the graph-based modeling framework for thermal fluid systems through experimental validation. A modular, and readily expandable experimental testbed is presented and used to showcase the ability of a graph-based modeling framework to capture the dynamics of a thermal fluid system. Furthermore, it is shown that a graph-based modeling approach provides a single flexible framework in which power flow dynamics can be represented using nonlinear or linear relationships.

## 6.2 Background

Conventional approaches to modeling and control of complex systems-of-systems are often limited to decentralized high-fidelity modeling and robust, low performance proportional-integral and logic-based control [26]. Holistic modeling, analysis, and control design is inhibited by the complexity and size of the systems, especially when dynamics evolve over a wide range of timescales and energy domains. As the complexity of systems continues to increase, developing, analyzing, and validating control designs must be conducted in simulation prior to application to the physical system. Due to the complexity of the systems and corresponding models, modular, toolbox-based modeling frameworks are often developed. Examples in the fields of building and vehicle energy management include the Thermosys™ [80] toolbox for modeling air-conditioning and refrigeration systems, the ATTMO [81] toolbox for modeling aircraft vapor cycle systems, and the PowerFlow toolbox for holistic aircraft power system modeling [82]. Each of these toolboxes consists of individual component models that can be interconnected to form complete systems. This modularity allows for individual sizing and validation of components and permits a wide range of system configurations and sizes to be implemented in simulation.

To validate both modeling toolboxes and control approaches, experimental testbed systems have been developed across a range of application areas. Examples include the vapor compression refrigeration testbeds of [80] and [83], the hydraulic hybrid vehicle testbed of [84], the aircraft fuel thermal management system testbed of [85], and the shipboard chilled water distribution system testbed of [86].

As shown in previous Chapters, a graph-based approach to modeling power flow systems can be particularly convenient for facilitating model-based control design. However, in order to prove the efficacy of these control techniques for real-world implementation, it is essential to demonstrate experimentally that graph-based modeling approaches can accurately capture the dynamics of power flow systems.

## 6.3 Graph-based Modeling

### 6.3.1 Generic Graph Formulation

The generic graph-based modeling framework presented in Chapter 2 is extended in this Chapters to capture the hydrodynamic and thermodynamic behavior of a thermal fluid system. Thus, graph-based models in this Chapter are derived from application of either conservation of mass or conservation of thermal energy. A graph derived from conservation of mass is referred to as a “hydraulic” graph, while a graph derived from conservation of thermal energy is referred to as a “thermal” graph. In both cases, each vertex has an associated dynamic state $x_i$ representing an amount of stored mass or energy. Similarly, each edge has an assigned value $y_j$ describing the rate of transfer of mass or energy (i.e., power flow) between adjacent vertices. While previous Chapters refer to this transfer rate as power $P_j$, for this Chapter the generic term $y_j$ is used since a set of two interacting graphs will be developed where $y_j$ refers to mass flow rate in a hydraulic graph and thermal power in the thermal graph. For either graph, the dynamics of each vertex satisfy the conservation equation

$$C_i \dot{x}_i = \sum_{e_j \in E_i^{in}} y_j - \sum_{e_j \in E_i^{out}} y_j, \tag{6.1}$$

where $C_i$ is the storage capacitance of the vertex. The transfer rate $y_j$ along edge $e_j$ is a function of the states $x_j^{tail}$ and $x_j^{head}$ of the incident vertices $v_j^{tail}$ and $v_j^{head}$ as well as an input $u_j$. Thus $y_j$ is given as

$$y_j = f_j(x_j^{tail}, x_j^{head}, u_j). \tag{6.2}$$

Based on the same formulation from Chapter 2, the dynamics of the states in system $\mathbf{S}$ are

$$\mathbf{S}: \; C\dot{x} = -\bar{M}y + Dy^{in}, \tag{6.3}$$

where $y^{in} \in \mathbb{R}^{N_s}$ now takes the place of $P^{in}$ and represents the disturbance source transfer rates from the environment into the system and $y = \left[ y_j \right] \in \mathbb{R}^{N_e}$ is

$$y = F\left(x, x^t, u\right), \tag{6.4}$$

where $x = \left[ x_i \right] \in \mathbb{R}^{N_v}$ are the states, $x^t = \left[ x_i^t \right] \in \mathbb{R}^{N_t}$ are the disturbance sink states, $u = \left[ u_j \right] \in \mathbb{R}^{N_e}$ are the inputs, and $F\left(x, x^t, u\right) = \left[ f_j\left(x_j^{tail}, x_j^{head}, u_j\right) \right]$.

## 6.3.2 Hydraulic Graph Modeling

When conservation of mass is used as the continuity equation for a graph of a fluid flow system, a model of its hydrodynamic relationships is obtained. A hydraulic graph is denoted as $\mathcal{G}^{\mathbf{m}}$, with corresponding system $\mathbf{S}^{\mathbf{m}}$ and the superscript $\mathbf{m}$ denoting conservation of mass. States of the hydraulic graph's vertices are pressures $p = [p_i], \; i \in [1, N_v^{\mathbf{m}}]$, while the transfer rates along its edges are mass flow rates $\dot{m} = [\dot{m}_j], \; j \in [1, N_e^{\mathbf{m}}]$. For this Chapter, all inputs to edges of the graph $u_i^{\mathbf{m}}$ are actuator effort in units of % duty cycle of a pulse width modulation (PWM) signal. It is assumed that no fluid is added to or drained from to the system, so $N_s^{\mathbf{m}} = N_t^{\mathbf{m}} = 0$.

Following from (6.3), the dynamics of the nonlinear hydraulic graph-based model are

$$\mathbf{S^m} : \; C^{\mathbf{m}} \dot{p} = -\bar{M}^{\mathbf{m}} \dot{m}, \tag{6.5}$$

where $C^{\mathbf{m}} = diag([C_i^{\mathbf{m}}])$ is the matrix of hydraulic capacitances of the vertices. The mass flow rate $\dot{m}_j$ along $e_j$ is a function of the pressure differential $p_j^{tail} - p_j^{head}$ between the incident vertices $v_j^{tail}$, $v_j^{head}$ and the state of the actuator $u_j^{\mathbf{m}}$. Therefore, following from (6.2),

$$\dot{m}_j = f_j^{\mathbf{m}}(p_j^{tail} - p_j^{head}, u_j^{\mathbf{m}}). \tag{6.6}$$

### 6.3.3 Thermal Graph Modeling

When conservation of thermal energy is used as the continuity equation for a graph of a fluid flow system, a model of its thermodynamic relationships is obtained. A thermal energy graph is denoted as $\mathcal{G}^{\mathbf{e}}$, with corresponding system $\mathbf{S^e}$ and the superscript $\mathbf{e}$ denoting conservation of energy. The states of the thermal graph's vertices are temperatures $T = [T_i],\, i \in [1, N_v^{\mathbf{e}}]$, while the transfer rates along its edges are thermal power flows $P = [P_j],\; j \in [1, N_e^{\mathbf{e}}]$. All inputs to edges of the graph are mass flow rates $\dot{m}^{\mathbf{e}} = [\dot{m}_j^{\mathbf{e}}],\; j \in [1, N_e^{\mathbf{e}}]$.

Following from (6.3), the dynamics of the nonlinear thermal graph-based model are

$$\mathbf{S^e} : \; C^{\mathbf{e}} \dot{T} = -\bar{M}^{\mathbf{e}} P + D^{\mathbf{e}} P^{in}, \tag{6.7}$$

where $C^{\mathbf{e}} = diag([C_i^{\mathbf{e}}])$ is the matrix of thermal capacitances of the vertices and $P^{in}$ is the vector of power flows along the source edges of the graph. The power flow $P_j$ along $e_j$ is a function of the temperatures $T_j^{tail}$, $T_j^{head}$ of the incident vertices $v_j^{tail}$, $v_j^{head}$ and the mass flow rate associated with the edge $\dot{m}_j^{\mathbf{e}}$. Therefore, following from (6.2),

$$P_j = f_j^{\mathbf{e}}(T_j^{tail}, T_j^{head}, \dot{m}_j^{\mathbf{e}}). \tag{6.8}$$

## 6.3.4 Multi-graph System Representation

Many physical components and systems are governed by both conservation of mass and conservation of thermal energy. Therefore, they can be represented by both a hydraulic graph $\mathcal{G}^{\mathbf{m}}$ with corresponding system $\mathbf{S}^{\mathbf{m}}$ as in (6.5) and a thermal energy graph $\mathcal{G}^{\mathbf{e}}$ with corresponding system $\mathbf{S}^{\mathbf{e}}$ as in (6.7). The coupling between the hydraulic and thermal graphs is limited to a unidirectional influence of mass dynamics on the thermal energy dynamics.

Mass flow rates are calculated in the hydraulic graph as its transfer rates $\dot{m}$. A subset of $\dot{m}$ are the edge inputs $\dot{m}^{\mathbf{e}}$ to the thermal graph, affecting the power flows along the thermal graph's edges. An example of this interconnection of the hydraulic and thermal graphs is shown in Fig. 6.1. There may also be mass flow rates affecting the power flows that are not calculated within the hydraulic graph. For example, this could include flow rates on the secondary side of heat exchangers by which heat is transferred to and from neighboring systems. These flow rates are denoted by $\dot{m}^{ext} = [\dot{m}_i^{ext}],\ i \in [1, N_{\dot{m}^{ext}}]$ and treated as disturbances to the thermal model.

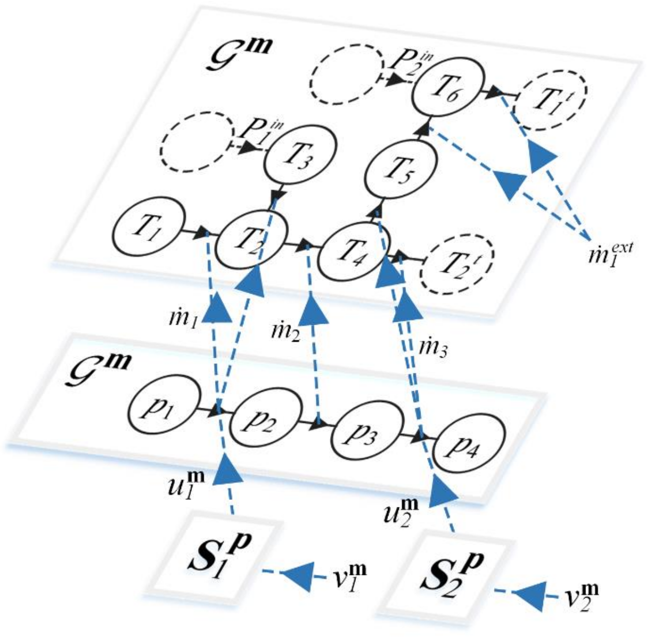


**Figure 6.1 Sample interconnection of thermal (top) and hydraulic (middle) graphs, with actuator dynamics (bottom) affecting the hydraulic edge inputs.**

The matrix $Z_1 \in \mathbb{R}^{N_e^{\mathbf{e}} \times (N_e^{\mathbf{m}} + N_{\dot{m}^{ext}})}$ is defined as a mapping from the mass flow rates $\dot{m}$ and $\dot{m}^{ext}$ to the input mass flow rates $\dot{m}^{\mathbf{e}}$, such that

$$\dot{m}^{\mathbf{e}} = Z_1 \begin{bmatrix} \dot{m} \\ \dot{m}^{ext} \end{bmatrix}. \tag{6.9}$$

Let $N_p^{\mathbf{m}}$ be the number of hydraulic actuators in the system. To account for dynamics including rate limits and time delays between each actuator command $v_i^{\mathbf{m}}, i \in [1, N_p^{\mathbf{m}}]$ and the actual actuator state $u_i^{\mathbf{m}}$ which affects the hydraulic graph, each $u_i^{\mathbf{m}}$ is paired with a single-input-single-output (SISO) system $\mathbf{S}_i^{\mathbf{p}}$ as shown in Fig. 6.1. Each $\mathbf{S}_i^{\mathbf{p}}$ models the state of the $i^{th}$ actuator as a function of its commanded value $v_i^{\mathbf{m}}$.

For this chapter, all actuators are pumps with states and inputs both in units of % duty cycle of PWM. The actuator dynamic of each pump is modeled as a first-order response with time constant $\tau_i^{\mathbf{p}}$ and delay $\xi_i^{\mathbf{p}} > 0$. This dynamic can be expressed as a transfer function by

$$\mathbf{S}_i^{\mathbf{p}} : \; u_i^{\mathbf{m}}(s) = \frac{e^{-\xi_i^{\mathbf{p}} s}}{\tau_i^{\mathbf{p}} s + 1} v_i^{\mathbf{m}}(s). \tag{6.10}$$

## 6.4 Conservation-based Modeling

The generic graph-based modeling framework presented in the previous Section can be used to capture the dynamics of a wide variety of thermal fluid systems that consist of a heterogeneous mix of components. Often it is useful to model components individually and then combine the individual component models to build up an entire system model.

Graph-based modeling relies on the assumption of lumped parameters. For example, the mass stored in a fluid volume is captured by a single representative pressure while the thermal energy stored in a thermal mass is captured by a single representative temperature. The first step to modeling a component is to identify the capacitive elements within the component and corresponding state values that represent the stored quantities. It is recommended that each component be represented with as few vertices as necessary to capture the relevant dynamics. If additional fidelity is needed, the component model can easily be further discretized with additional vertices and states. Once the vertices are identified, it is often a simple matter to determine the possible paths by which mass or energy can enter or exit that storage element and to represent these paths as edges. In order to keep models simple, it is suggested that only dominant power flows are represented as edges. If, during validation of the graph, it becomes apparent that a significant power flow was omitted from the graph, such as heat loss to ambient, edges can easily be added to improve the accuracy of the model.

For demonstration purposes, the remainder of this Section develops a set of models for components often found in an experimental thermal fluid systems to be presented in Section 6.6. These components include a fluid reservoir, a flow split/junction, a pump, a pipe, a cold plate heat exchanger, and a liquid-to-liquid brazed plate heat exchanger. Fig. 6.2 shows the mass conservation and thermal energy conservation graphs for each component. Dashed lines, indicating disturbances to each component, consist of variables determined by neighboring components. For example, the reservoir and flow split/junction only calculate their own pressure based on mass flow rates determined by neighboring components. However, the pump and heat exchangers calculate their own outlet pressure and inlet mass flow rate based on the upstream pressure and downstream mass flow rate. The following details the modeling of these components based on their graph frameworks from Fig. 6.2.

### 6.4.1 Mass Conservation

All pressure dynamics are derived from the mass conservation equation $\dot{M} = \dot{m}_1 - \dot{m}_2$, where $\dot{M}$ is the rate-of-change of fluid mass stored in the component and $\dot{m}_1$ and $\dot{m}_2$ are the total flow rates into and out of the component. For components with a fixed volume $V$, the change in mass stored in a component is based on the change in density $\rho$ of the fluid as a function of pressure $p$. Thus $\dot{M} = V\dot{\rho} = V\left(\partial\rho/\partial p\right)\dot{p}$. Noting that the change in density with pressure is based on the bulk modulus of the fluid $E$, where $\partial\rho/\partial p = \rho/E$, the mass conservation equation provides a dynamic equation for pressure within the component, where

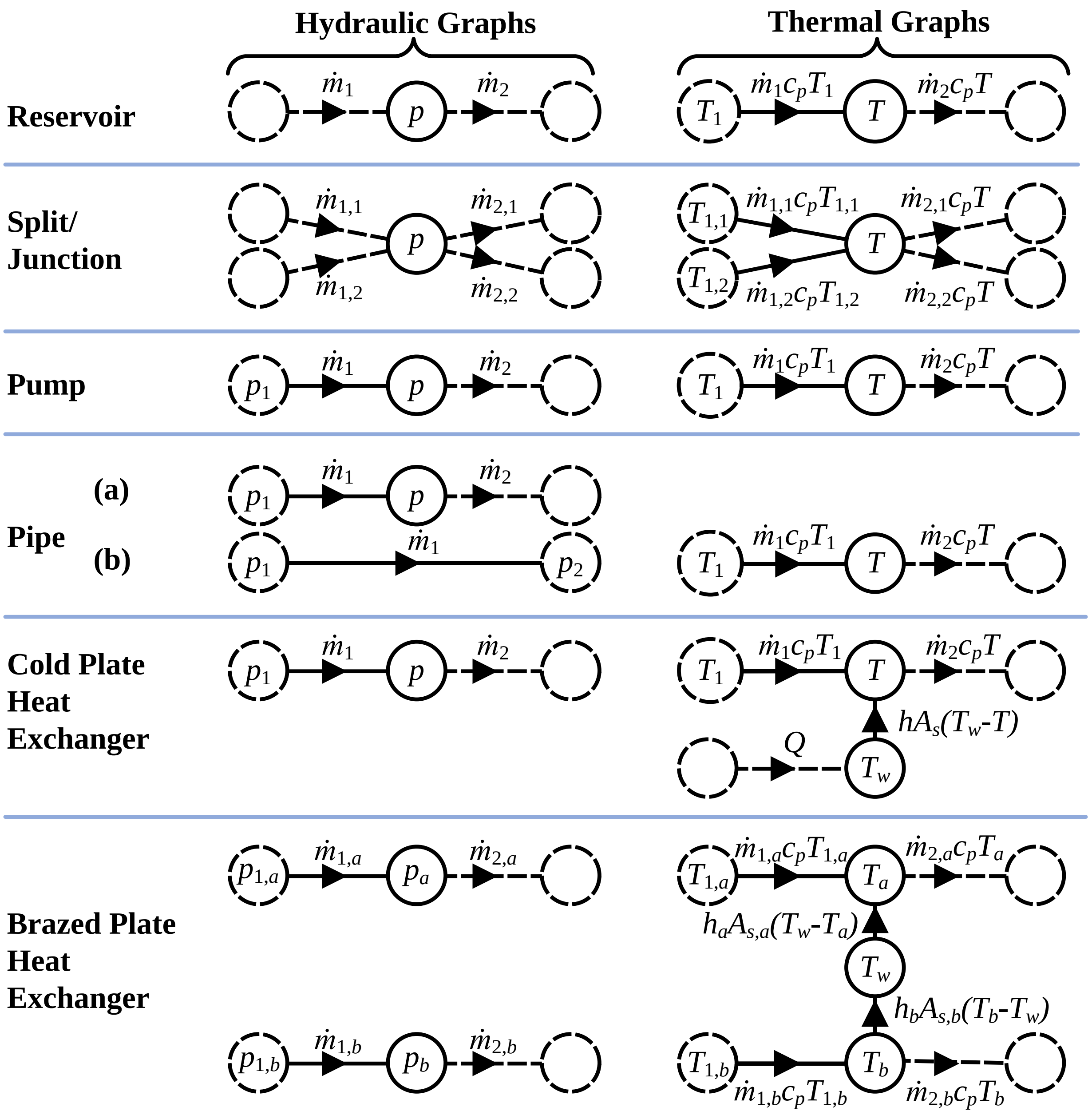


**Figure 6.2 Hydraulic and thermal graphs for individual components.**

$$V\frac{\rho}{E}\dot{p} = \dot{m}_1 - \dot{m}_2. \tag{6.11}$$

Currently, the only component without a fixed volume is the reservoir. The reservoir has a constant cross sectional area $A_{c,r}$ with a liquid height $h_r$. The top of the reservoir is subject to ambient air pressure $p_{amb}$. Fluid flows into and out of the reservoir from the bottom with flow rates $\dot{m}_1$ and $\dot{m}_2$. The mass stored in the reservoir $M_r$ changes as a function of these flow rates: $\dot{M}_r = \dot{m}_1 - \dot{m}_2$. This dynamic is expressed in terms of the pressure $p_r$ at the bottom of the reservoir using the relationship between mass and liquid height, $M_r = \rho A_{c,r} h_r$, and the relationship for static pressure in a liquid, $p_r = p_{amb} + \rho g h_r$. The resulting pressure dynamic is

$$A_c \dot{p}_r = g\left(\dot{m}_1 - \dot{m}_2\right). \tag{6.12}$$

The flow split/junction has $n$ inlets and $m$ outlets and thus when using (6.11) the inlet and outlet flow rates are calculated as $\dot{m}_1 = \sum_{i=1}^{n} \dot{m}_{1,i}$ and $\dot{m}_2 = \sum_{i=1}^{n} \dot{m}_{2,i}$.

The mass flow rates through pipes and heat exchangers are based on the pressure drop across the component $\Delta p = p_1 - p$ and the height difference between the inlet and outlet flow $\Delta h$. Fluid flows through a cross sectional area of $A_c$, based on tube diameter $D$, for a length $L$. Major losses are determined based on the friction factor $f$ and minor losses are modeled using the a minor loss coefficient $K_L$. Pipes may include the pressure drop effects of various sensors via this minor loss coefficient. With $s$ sensors along the pipe the total minor loss coefficient is $K_L = K_L^{pipe} + \sum_{i=1}^{s} K_L^i$. The resulting equation for the mass flow rate through a pipe or heat exchanger is

$$\dot{m} = \rho A_c \sqrt{\frac{2\left(p_1 - p + \rho g \Delta h\right)}{\rho\left(f\dfrac{L}{D} + K_L\right)}}. \tag{6.13}$$

Note that Fig. 6.2 shows two forms of pipes. Pipe version (a) is the standard component that calculates a dynamic outlet pressure $p$ and the inlet mass flow rate $\dot{m}_1$. Pipe version (b) only calculates a mass flow rate $\dot{m}$ between two pressures $p_1$ and $p_2$, which are determined by neighboring components. Version (b) of the pipe is used at the inlet to the reservoir and flow split/junction, since these components do not calculate their own inlet mass flow rates.

For the brazed plate heat exchangers, there are $N_c$ channels for each fluid, the width of each plate is $W$, and the spacing between plates is $b$. Thus when using (6.13), $A_c = N_c b W$ is the cross sectional area of a single channel multiplied by the number of channels and $D = 4bW/(2b+2W)$ is the hydraulic diameter of a single channel.

The mass flow rate calculation for the pump is a function of the pressure differential across the pump $\Delta p_p = p - p_1$ and the pump speed $\omega$. The mass flow rate is $\dot{m} = \rho A_c u_m$, where $u_m$ is the mean fluid velocity. From conservation of mechanical energy the fluid velocity is $u_m = \sqrt{2g\left(H - \Delta p_p/(\rho g)\right)}$, where $H = H\left(\Delta p_p, \omega\right)$ is the pump head. Thus the mass flow rate through the pump is

$$\dot{m} = \rho A_c \sqrt{2g\left(H - \frac{\Delta p_p}{\rho g}\right)}. \tag{6.14}$$

Fig. 6.3 shows an example of a experimentally obtained pump head map with $H = k_1 + k_2 \Delta p_p + k_3 \omega$.

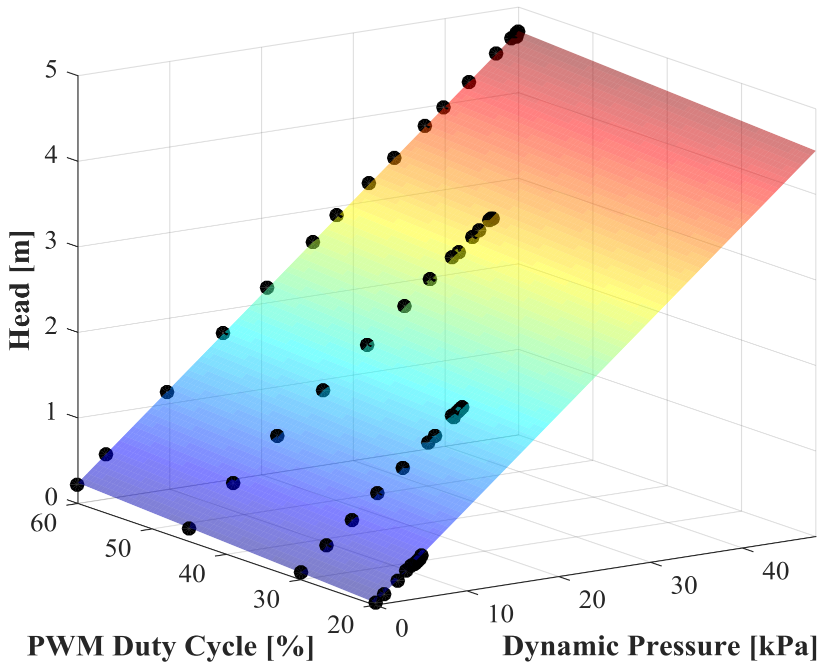


**Figure 6.3 Example pump head map.**

## 6.4.2 Thermal Energy Conservation

All temperature dynamics are derived from the thermal energy conservation equation $\dot{E}_{st} = P_1 - P_2$, where $E_{st} = Mc_pT$ is the stored thermal energy and $P_1$ and $P_2$ are the rate of thermal energy entering or exiting the storage element. In general, $\dot{E}_{st} = \dot{M}c_pT + Mc_p\dot{T}$, which accounts for the change in thermal energy associated with the change of mass $M$. The first term is important to consider for components, such as the reservoir, which may undergo a significant change in mass. However, for most components, $\dot{E}_{st} \cong Mc_p\dot{T} = \rho Vc_p\dot{T}$. For the reservoir, pump, and pipes, the power flow due to fluid flow into the component $P_1$ is $P_1 = \dot{m}_1c_pT_1$ and the power flow out of the component $P_2$ is $P_2 = \dot{m}_2c_pT$. The lumped temperature represents the fluid temperature at the outlet of the component with the dynamic

$$\rho Vc_p\dot{T} = \dot{m}_1c_pT_1 - \dot{m}_2c_pT. \tag{6.15}$$

For the flow split/junction, the temperature dynamic is similar with

$$\rho Vc_p\dot{T} = \sum_{i=1}^{n}\dot{m}_{1,i}c_pT_{1,i} - \sum_{i=1}^{m}\dot{m}_{2,i}c_pT. \tag{6.16}$$

The cold plate heat exchanger has an additional temperature dynamic capturing the thermal capacitance of the wall. With a heat load of $Q$, the cold plate wall temperature dynamic is

$$M_wc_{p,w}\dot{T}_w = Q - hA_s\left(T_w - T\right), \tag{6.17}$$

where $h$ is the heat transfer coefficient and $A_s$ is the convective surface area. The heat transfer coefficient is calculated based on a Nusselt number, $Nu = hD/k$, of $Nu = 3.66$ for laminar flow or the Gnielinski equation [87]

$$Nu = \frac{\left(f/8\right)\left(\mathrm{Re}-1000\right)\mathrm{Pr}}{1+12.7\left(f/8\right)^{1/2}\left(\mathrm{Pr}^{2/3}-1\right)}, \tag{6.18}$$

for turbulent flow. With the additional convective heat flow, the fluid outlet temperature dynamic for the cold plate is

$$\rho V c_p \dot{T} = \dot{m}_1 c_p T_1 + h A_s \left(T_w - T\right) - \dot{m}_2 c_p T. \tag{6.19}$$

Finally, the brazed plate heat exchanger is modeled similarly to the cold plate heat exchanger where the heat load $Q$ is replaced by secondary fluid flow. The plates of the heat exchanger are assumed to be at a uniform lumped temperature $T_w$ with the dynamic

$$M_w c_{p,w} \dot{T}_w = h_b A_{s,b} \left(T_b - T_w\right) - h_a A_{s,a} \left(T_w - T_a\right), \tag{6.20}$$

where subscripts $a$ and $b$ denote the primary and secondary fluids channels, $M_w$ is the mass of a single plate, and $A_s$ is the convective surface area for a single channel. For the plate heat exchangers, the heat transfer coefficient is based on the empirical results from [88], where

$$Nu = 0.277 \, \mathrm{Re}^{0.766} \, \mathrm{Pr}^{0.333}. \tag{6.21}$$

Note that all components are assumed to be adiabatic and do not exchange heat with the surroundings. If this heat loss needs to be considered, the component graphs in Fig. 6.2 could easily be modified with an additional edge directed to a new vertex with a corresponding state equal to the ambient air temperature.

In general, the equations used to represent the hydraulic and thermodynamic behaviors in this Section have a nonlinear form but satisfy the generic conservation and power flow relationships from (6.1) and (6.2). For control design in particular, it is often useful to use a linear representation of the system dynamics. One of the key benefits of a graph-based approach is that this linearization can be performed for each power flow relationship individually as discussed in the following Section.

## 6.5 Linearization and Discretization

### 6.5.1 Hydraulic Graph Linearization

To generate a linear hydraulic graph model for use in control design, the generic mass flow rate relationship of (6.6) is linearized about an equilibrium operating condition using a first-order Taylor Series, giving

$$\Delta \dot{m}_j = a_j^{\mathbf{m}} (\Delta p_j^{tail} - \Delta p_j^{head}) + b_j^{\mathbf{m}} \Delta u_j^{\mathbf{m}}, \tag{6.22}$$

where, for a generic signal $x(t)$, $\Delta x(t) := x(t) - x_0$ and $x_0$ is the equilibrium value of $x$ about which the linearization is taken. The dynamics of the linearized hydraulic model are then given by

$$\dot{p} = A^{\mathbf{m}} \Delta p + B^{\mathbf{m}} \Delta u^{\mathbf{m}}, \tag{6.23}$$

where

$$A^{\mathbf{m}} = -\left(C^{\mathbf{m}}\right)^{-1} \bar{M}^{\mathbf{m}} diag\left(\left[a_j^{\mathbf{m}}\right]\right)\left(\bar{M}^{\mathbf{m}}\right)^T, \tag{6.24}$$

$$B^{\mathbf{m}} = -\left(C^{\mathbf{m}}\right)^{-1} \tilde{\bar{M}}^{\mathbf{m}} diag\left(\left[\tilde{b}_j^{\mathbf{m}}\right]\right), \tag{6.25}$$

and $\tilde{\bar{M}}^{\mathbf{m}}$ represents the columns of $\bar{M}^{\mathbf{m}}$ corresponding to edges with associated actuators and $\tilde{b}_j^{\mathbf{m}}$ are the input coefficients for edges with actuators.

The output equation of the linearized hydraulic model relating pressures and actuator efforts to mass flow rates is given by

$$\dot{m} = C^{\mathbf{m}} \Delta p + D^{\mathbf{m}} \Delta u^{\mathbf{m}}, \tag{6.26}$$

where

$$C^{\mathbf{m}} = diag\left(\left[a_j^{\mathbf{m}}\right]\right)\left(\bar{M}^{\mathbf{m}}\right)^T, \tag{6.27}$$

$$D^{\mathbf{m}} = \left[d_{j,k}^{\mathbf{m}}\right] \in \mathbb{R}^{N_e^{\mathbf{m}} \times N_p^{\mathbf{m}}}, \tag{6.28}$$

and

$$d_{j,k}^{\mathbf{m}} = \begin{Bmatrix} b_j^{\mathbf{m}} & \text{if } e_j \text{ is associated with actuator } k \\ 0 & \text{else} \end{Bmatrix}. \tag{6.29}$$

## 6.5.2 Hydraulic Graph Discretization

The dynamics of the hydraulic system evolve relatively quickly, on the order of fractions of seconds. Therefore, care must be taken to preserve stability and maintain sufficient numerical tolerances when discretizing the continuous model to relatively slow update rates, for example on the order of 1 Hz. Furthermore, conservation of mass dictates that $A^{\mathbf{m}}$ is singular, which violates the assumptions of several common discretization approaches. These issues motivate the multistep process described below for obtaining discrete models of the hydraulic system. The linear hydraulic model is first discretized at a relatively fast update rate on the order of 10,000 Hz using a zero-order hold. This yields the discrete dynamic model

$$\Delta p(k+1) = A_{d,fast}^{\mathbf{m}} \Delta p(k) + B_{d,fast}^{\mathbf{m}} \Delta u^{\mathbf{m}}(k), \tag{6.30}$$

with state matrices given by

$$A_{d,fast}^{\mathbf{m}} = \exp\left(A^{\mathbf{m}} \Delta t_{fast}^{\mathbf{m}}\right), \tag{6.31}$$

$$B_{d,fast}^{\mathbf{m}} = \int_{\tau=0}^{\Delta t_{fast}^{\mathbf{m}}} \exp\left(A^{\mathbf{m}} \tau\right) d\tau B^{\mathbf{m}}, \tag{6.32}$$

where $\Delta t_{fast}^{\mathbf{m}}$ is the time step between consecutive updates.

For the component size and configuration of the system to be implemented in this Chapter, 10,000 Hz has been found to be a sufficiently fast update rate to preserve stability properties of the continuous system. However, as demonstrated in Chapter 7, this is orders of magnitude faster than what is required to control the hydrodynamics of such a system. Therefore, a hydraulic model at a slower update rate is desired. To better capture the continuous behavior of the actuators at a slower time step, such a model is derived by downsampling the "fast" discrete model using a *first-order* hold rather than a zero-order hold. This essentially preserves knowledge of the rate limit of the actuator effort, which under a first-order hold is assumed to ramp between time steps. By comparison, a zero-order hold would assume stepped actuator inputs with an instantaneous rate of change.

The discrete model used for control has a time step of $\Delta t^{\mathbf{m}}$ given by

$$\Delta t^{\mathbf{m}} = N_{ds}^{\mathbf{m}} \Delta t_{fast}^{\mathbf{m}} \text{ s.t. } N_{ds}^{\mathbf{m}} > 1 \text{ and } N_{ds}^{\mathbf{m}} \in \mathbb{N}. \tag{6.33}$$

The use of a first-order hold in performing this downsampling yields the discrete dynamic model

$$\Delta p(k+1) = A_d^{\mathbf{m}} \Delta p(k) + B_{d,1}^{\mathbf{m}} \Delta u^{\mathbf{m}}(k) + B_{d,2}^{\mathbf{m}} \Delta u^{\mathbf{m}}(k+1), \tag{6.34}$$

with state space matrices given by

$$A_d^{\mathbf{m}} = \left(A_{d,fast}^{\mathbf{m}}\right)^{N_{ds}^{\mathbf{m}}}, \tag{6.35}$$

$$B_{d,1}^{\mathbf{m}} = \left(\sum_{i=0}^{N_{ds}^{\mathbf{m}}-1} \left(A_{d,fast}^{\mathbf{m}}\right)^{N_{ds}^{\mathbf{m}}-1-i} \left(1 - \frac{i}{N_{ds}^{\mathbf{m}}}\right)\right) B_{d,fast}^{\mathbf{m}}, \tag{6.36}$$

$$B_{d,2}^{\mathbf{m}} = \left(\sum_{i=1}^{N_{ds}^{\mathbf{m}}-1} \left(A_{d,fast}^{\mathbf{m}}\right)^{N_{ds}^{\mathbf{m}}-1-i}\right) B_{d,fast}^{\mathbf{m}}. \tag{6.37}$$

Because (6.34) depends on knowledge of $u^{\mathbf{m}}(k+1)$ to compute $\Delta p(k+1)$, (6.34) is clearly a non-causal system. However, as shown in Chapter 7, this does not present a problem when using this model for an MPC controller.

### 6.5.3 Thermal Graph Linearization

To generate a linear thermal graph model, the power flow relationship of (6.8) is linearized about an equilibrium operating condition using a first-order Taylor Series, giving

$$\Delta P_j = a_j^{\mathbf{e}} \Delta T_j^{tail} + b_j^{\mathbf{e}} \Delta T_j^{head} + c_j^{\mathbf{e}} \Delta \dot{m}_j^{\mathbf{e}}. \tag{6.38}$$

The dynamics of the linearized thermal model are then given by

$$\dot{T} = A^{\mathbf{e}} \Delta T + B^{\mathbf{e}} \Delta \dot{m}^{\mathbf{e}} + V_1^{\mathbf{e}} \Delta P^{in} + V_2^{\mathbf{e}} \Delta T^{t}, \tag{6.39}$$

where

$$A^{\mathbf{e}} = -\left(C^{\mathbf{e}}\right)^{-1} \bar{M}^{\mathbf{e}} \left(\bar{M}^{\mathbf{e}}_{a,b}\right)^T, \tag{6.40}$$

$$B^{\mathbf{e}} = -\left(C^{\mathbf{e}}\right)^{-1} diag\left(\left[c^{\mathbf{e}}_j\right]\right), \tag{6.41}$$

$$V^{\mathbf{e}}_1 = \left(C^{\mathbf{e}}\right)^{-1} D^{\mathbf{e}}, \tag{6.42}$$

$$V^{\mathbf{e}}_2 = -\left(C^{\mathbf{e}}\right)^{-1} \bar{M}^{\mathbf{e}} \left(\underline{M}^{\mathbf{e}}_{a,b}\right)^T, \tag{6.43}$$

and $M^{\mathbf{e}}_{a,b} = \left[m_{i,j}\right] \in \mathbb{R}^{\left(N^{\mathbf{e}}_v + N^{\mathbf{e}}_t\right) \times N^{\mathbf{e}}_e}$ is a weighted incidence matrix for the thermal graph with

$$m_{i,j} = \begin{Bmatrix} a^{\mathbf{e}}_j & \text{if } v_i \text{ is the tail of } e_j \\ b^{\mathbf{e}}_j & \text{if } v_i \text{ is the head of } e_j \\ 0 & \text{else} \end{Bmatrix}. \tag{6.44}$$

## 6.5.4 Thermal Graph Discretization

The dynamics of the thermal system evolve much slower than those of the hydraulic system, on the order of tens of seconds. Therefore, the use of a zero-order hold is sufficient for generating a discrete model at the rate desired for control. This yields the discrete dynamic model

$$\Delta T(k+1) = A^{\mathbf{e}}_d \Delta T(k) + B^{\mathbf{e}}_d \Delta \dot{m}^{\mathbf{e}}(k) + V^{\mathbf{e}}_{d,1} \Delta P^{in}(k) + V^{\mathbf{e}}_{d,2} \Delta T^t(k), \tag{6.45}$$

with state space matrices given by

$$A^{\mathbf{e}}_d = \exp\left(A^{\mathbf{e}} \Delta t^{\mathbf{e}}\right), \tag{6.46}$$

$$B^{\mathbf{e}}_d = \left(A^{\mathbf{e}}\right)^{-1} \left(A^{\mathbf{e}}_d - I\right) B^{\mathbf{e}}, \tag{6.47}$$

$$V^{\mathbf{e}}_{d,1} = \left(A^{\mathbf{e}}\right)^{-1} \left(A^{\mathbf{e}}_d - I\right) V^{\mathbf{e}}_1, \tag{6.48}$$

$$V_{d,2}^{\mathbf{e}} = \left(A^{\mathbf{e}}\right)^{-1}\left(A_d^{\mathbf{e}} - I\right)V_2^{\mathbf{e}}, \tag{6.49}$$

where $\Delta t^{\mathbf{e}}$ is the time step between consecutive updates.

## 6.5.5 Actuator Dynamics

The continuous-time model of the actuator dynamics given as a transfer function in (6.10) can be equivalently expressed as

$$\dot{u}_i^{\mathbf{m}}(t) = a_i^{\mathbf{p}} u_i^{\mathbf{m}}(t) + b_i^{\mathbf{p}} v_i^{\mathbf{m}}\left(t - \xi_i^{\mathbf{p}}\right), \tag{6.50}$$

where

$$a_i^{\mathbf{p}} = -\frac{1}{\tau_i^{\mathbf{p}}},\ b_i^{\mathbf{p}} = \frac{1}{\tau_i^{\mathbf{p}}}. \tag{6.51}$$

The update rate of the discrete actuator model $\Delta t^{\mathbf{p}}$ is defined such that the delay $\xi_i^{\mathbf{p}}$ is the integer $N_{ds,i}^{\mathbf{p}}$ multiple of $\Delta t^{\mathbf{p}}$. The discrete model is then given by

$$u_i^{\mathbf{m}}(k+1) = a_{d,i}^{\mathbf{p}} u_i^{\mathbf{m}}(k) + b_{d,i}^{\mathbf{p}} v_i^{\mathbf{m}}\left(k - N_{ds,i}^{\mathbf{p}}\right), \tag{6.52}$$

where

$$a_{d,i}^{\mathbf{p}} = \exp\left(a_i^{\mathbf{p}} \Delta t_i^{\mathbf{p}}\right),\ b_i^{\mathbf{p}} = 1 - a_{d,i}^{\mathbf{p}}. \tag{6.53}$$

## 6.6 Experimental System Description

The following experimental system is used to demonstrate the applicability and validity of the graph-based modeling framework presented in the previous Sections. This experimental testbed was developed to emulate features of power flow systems while being rapidly reconfigurable to allow for numerous system architectures. Currently, the experimental system focuses on the thermal and hydrodynamic energy domains, with future work concentrating on expansion to the electrical domain. This experimental system is used to demonstrate the implementation of a hierarchical model predictive control framework in Chapter 7.

### 6.6.1 Overall System

Fig. 6.4 shows the testbed with a sample system configuration along with the corresponding system schematic. The slatted design of the testbed allows components to be placed in arbitrary horizontal or vertical positions, similar to a breadboard for electrical circuits. The working fluid is an equal parts mixture of propylene glycol and water. Components use standard G1/4 threaded barbs and are connected via flexible tubing. Sensors and pumps are connected to a National Instruments CompactDAQ via custom USB plug interfaces.

### 6.6.2 Individual Components

Fig. 6.5 presents images and specifications of the components currently included in the testbed.

Centrifugal pumps are the primary fluid movers in the system. Speed is controlled via a PWM duty cycle with <20% being a constant 1300RPM, 65% and above being 4500RPM, and a linear trend between. Peak power consumption of the pumps is 20W with a peak efficiency of 35%.

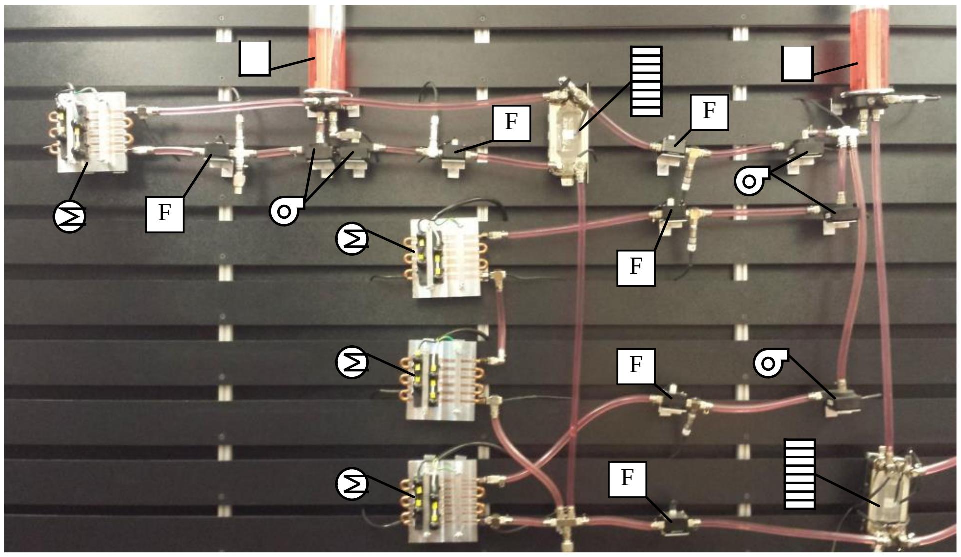
Liquid-to-liquid brazed plate heat exchangers (HX) allow for the transfer of heat among various fluid loops in either a parallel-flow or counter-flow configuration.

The cold plate heat exchanger consists of two 47Ω resistive heater wired in parallel, capable of 2kW peak power output, mounted to an aluminum cold plate that has copper tubing passing through. The heater is connected to a solid-state relay which allows for 0-100% power output using the 208VAC wall power supply.

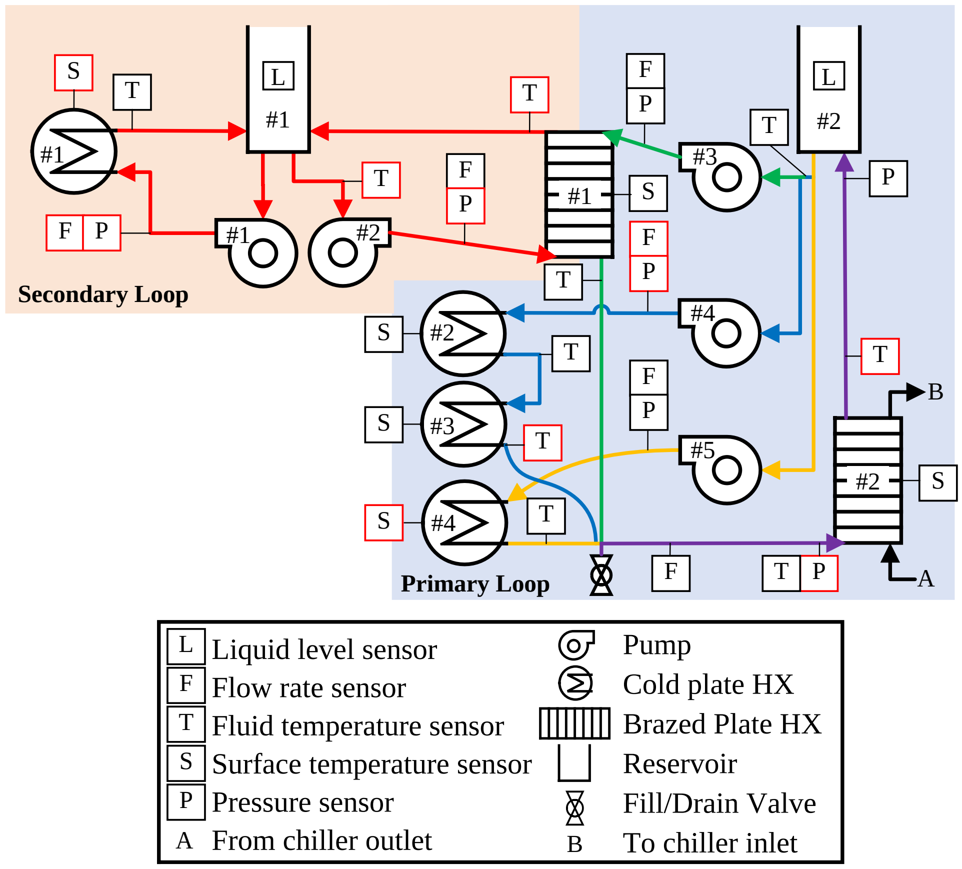
The reservoir acts as a thermal storage element. A liquid level sensor inside the reservoir allows for the calculation of the liquid mass and therefore thermal capacitance of the reservoir.

A 1.5HP (1.12kW) industrial chiller acts as a heat sink (e.g. a vapor compression system). With variable temperature control from -10$^o$C to 70$^o$C, the chiller can emulate a wide range of source and sink temperatures.

Temperature and pressure sensors utilize G1/4 threads and integrate seamlessly into the tube junctions. As such, limited pressure drops are incurred due to the inclusion of these sensors within the system. Similarly, mass flow sensors use G1/4 threads to attach in line with pipes but the paddlewheel-based design does introduce significant pressure drops.

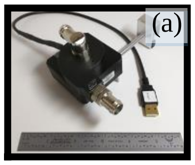


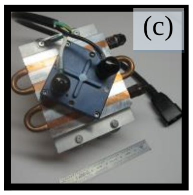
(a) Experimental thermal fluid system.



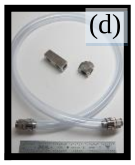
(b) System Schematic (red sensors plotted for model validation in Section 6.8)

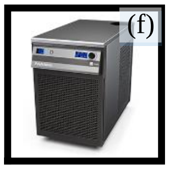
**Figure 6.4 Candidate thermal power architecture for simulation and experimental validation.**

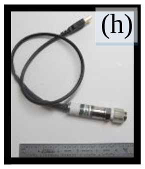




| *Component* | *Number* | *Details* |
|---|---|---|
| (a) Pump | 8 | • Swiftech MCP35X<br>• 12VDC, 1.5A max, PWM ctrl.<br>• 4.4 m max head<br>• 17.5 LPM max flow |
| (b) Brazed Plate HX | 4 | • Koolance HXP-193<br>• 12 plates<br>• 4.0 kW @ 5 LPM and 20°C inlet temp. diff. |
| (c) Cold Plate HX | 4 | • Ohmite CP4 with TAP2000 thick film resistor<br>• 0.018 °C/W thermal resistance<br>• 2000W |
| (d) Pipe | - | • Koolance HOS-13CL<br>• Clear PVC<br>• 13mm x 16mm |
| (e) Reservoir | 4 | • Koolance 80x240mm<br>• Acrylic<br>• 8” eTape Liquid Level Sensor |
| (f) Chiller | 1 | • Polyscience 6000 Series<br>• Up to 2900W @ 20°C<br>• -10°C to +70°C |
| (g) Temp. Sensor | 16 | • Koolance SEN-AP008B<br>• 10K ohm thermistor |
| (h) Pressure. Sensor | 7 | • Measurement Specialties US300<br>• 0 – 100kPa gauge |
| (i) Flow Rate Sensor | 8 | • Aqua Computer High Flow<br>• 0.5 – 25 LPM |







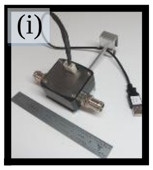


**Figure 6.5 Individual components and specification with a 6” ruler for scale.**

## 6.7 Graph-based System Representation

To represent an entire system as a graph, the individual component models from Fig. 6.2 in Section 6.4 are simply connected to reflect the given system architecture. The example system configuration shown in Fig. 6.4 is modeled using the graph-based framework with the resulting hydraulic and thermal energy graphs shown in Fig. 6.6. The following Subsections demonstrate how the conservation-based modeling equations from Section 6.4 are assembled into the generic graph-based models from Section 6.3 for the example experimental system configuration.

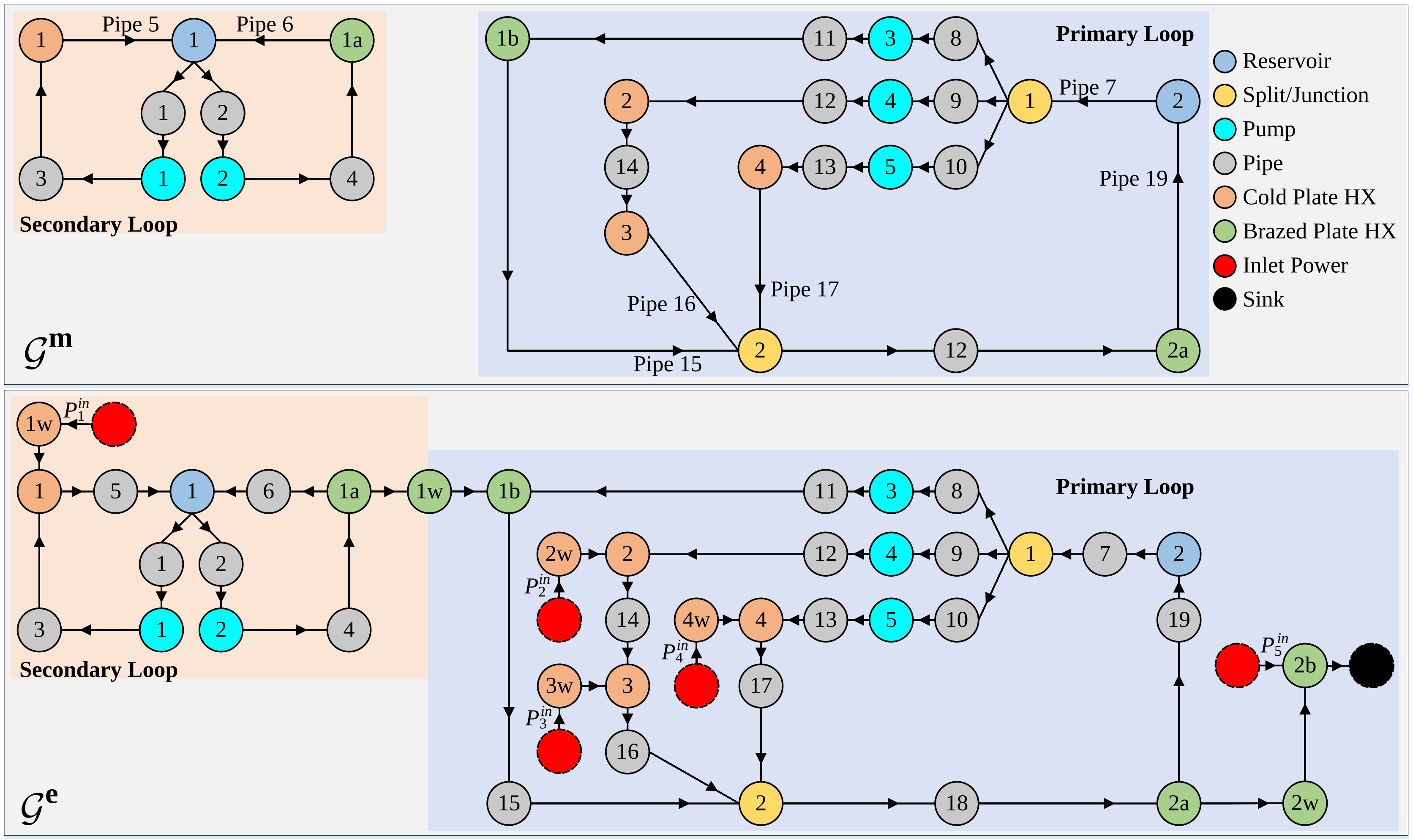


**Figure 6.6 Hydraulic and thermal graphs for the example experimental system configuration.**

### 6.7.1 Mass Conservation System

Since the primary and secondary flow loops do not exchange mass, the hydraulic graph in Fig. 6.6 has two independent components. The dynamics of this system follow from (6.5), where $C_i^{\mathbf{m}} = V\rho/E$ for all vertices except the reservoir where $C_i^{\mathbf{m}} = A_c/g$. The mass flow rate function $\dot{m}_j = f_j^{\mathbf{m}}\left(p_j^{tail} - p_j^{head}, u_j^{\mathbf{m}}\right)$ equals (6.13) for the pipes and heat exchangers and (6.14) for the pumps. To simplify the graph equations, a constant fluid density of $\rho = 1041 kg/m^3$ is used. For the linear graph results in the following Section, (6.13) and (6.14) are linearized about a nominal operating condition (50% pump PWM duty cycles).

### 6.7.2 Thermal Energy Conservation System

Fig. 6.6 also shows the thermal graph for the experimental system. The five inlet power flows consist of the four heat loads to the cold plates and the fluid flow from the chiller entering the secondary side of heat exchanger 2. Thus in (6.7),

$$P^{in} = \begin{bmatrix} Q_1 & Q_2 & Q_3 & Q_4 & \dot{m}_{2,b} c_p T_{2,b} \end{bmatrix}^T, \tag{6.54}$$

where $Q_i$ is the heat load to the $i^{th}$ cold plate heat exchanger and $\dot{m}_{2,b}$ is the secondary fluid flow rates through heat exchanger 2 with corresponding inlet temperature $T_{2,b}$ set by the chiller. The thermal capacitances $C_i^{\mathbf{e}} = \rho V c_p$ for all fluid temperatures and $C_i^{\mathbf{e}} = M_w c_{p,w}$ for all heat exchanger wall temperatures. The thermal power flow function $P_j = f_j^{\mathbf{e}}\left(T_j^{tail}, T_j^{head}, \dot{m}_j^{\mathbf{e}}\right)$ equals $P_j = \dot{m}_j^{\mathbf{e}} c_p T_j^{tail}$ for all thermal power flow due to fluid flow and $P_j = h_j A_{s,j}\left(T_j^{tail} - T_j^{head}\right)$ for convective thermal power flows in the heat exchangers. To simplify the graph system equations, a constant fluid specific heat of $c_p = 3500\, J/(kg \cdot K)$ is used. Additionally, to reduce the complexity of the power flow equations, the heat transfer correlation from (6.18) is approximated, with (6.17) and (6.19) using $h = \alpha_1 + \alpha_2 \dot{m} T$, where $\alpha_1 = 2000$ and $\alpha_2 = 0$ for the cold plate heat exchanger.

Similarly, the heat transfer correlation from (6.21) for the brazed plate heat exchanger is approximated, and (6.20) uses $\alpha_1 = 2572$ and $\alpha_2 = 1136$. While these approximations are used successfully within the range of operating conditions seen in the current experimental systems, the nonlinear (6.18) and (6.21) may be used over wider ranges of conditions.

## 6.8 Model Validation

In this Section, the graph-based modeling approach of Section 6.7 is validated by comparison of experimental data from the testbed of Section 6.6 to the linear graph-based models, using the configuration shown in Fig. 6.4. This linear model is used for hierarchical control in Chapter 7. Separate experimental tests are used to validate the hydrodynamic and thermodynamic domains so that each is validated under excitation on an appropriate timescale (i.e., the hydrodynamics are validated using rapid steps in pump speed, while the significantly slower thermodynamics are validated with slower steps in pump speed and heat load).

### 6.8.1 Hydrodynamic Validation

Fig. 6.7 shows the pump input sequence used to validate the hydrodynamics of the models, where the pump numbering follows from that of Fig. 6.4. Fig. 6.8 shows a subset of the measured outputs from the testbed (labeled as “Measured”), as well as the graph-based model (labeled as “Linear Graph Model”). The pressures at the outlet of pumps 1, 2, and 4 are shown along with the pressure at the inlet to the primary side of heat exchanger 2 and the mass flow rates at the outlet of pumps 1 and 4. From Fig. 6.8, the linear graph model captures some pressures very accurately, such as the pressures for pumps 1 and 4, while fails to accurately model other pressures, such as the pressure for pump 2. This model inaccuracy is likely due to the nonlinearity of the pump equation (6.14) and the pressure drop equation (6.13) for flow through pipes and heat exchangers. Additional fine tuning of the minor and major loss coefficients would also improve model accuracy. However, as will be shown in Chapter 7, a hierarchical controller using this linear graph model is capable of effectively controlling the nonlinear experimental system.

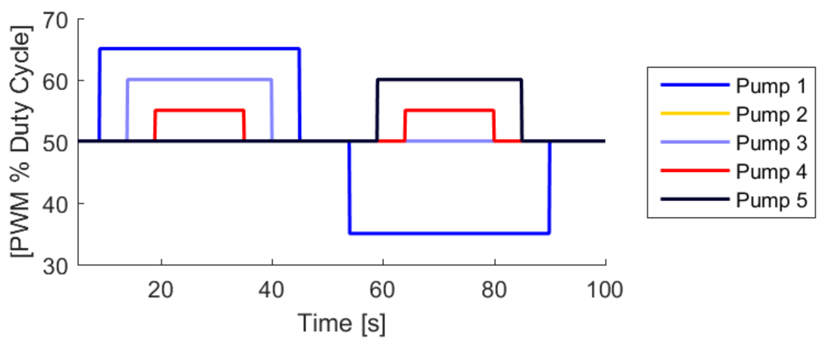


**Figure 6.7 Pump PWM duty cycle inputs for hydrodynamic validation.**

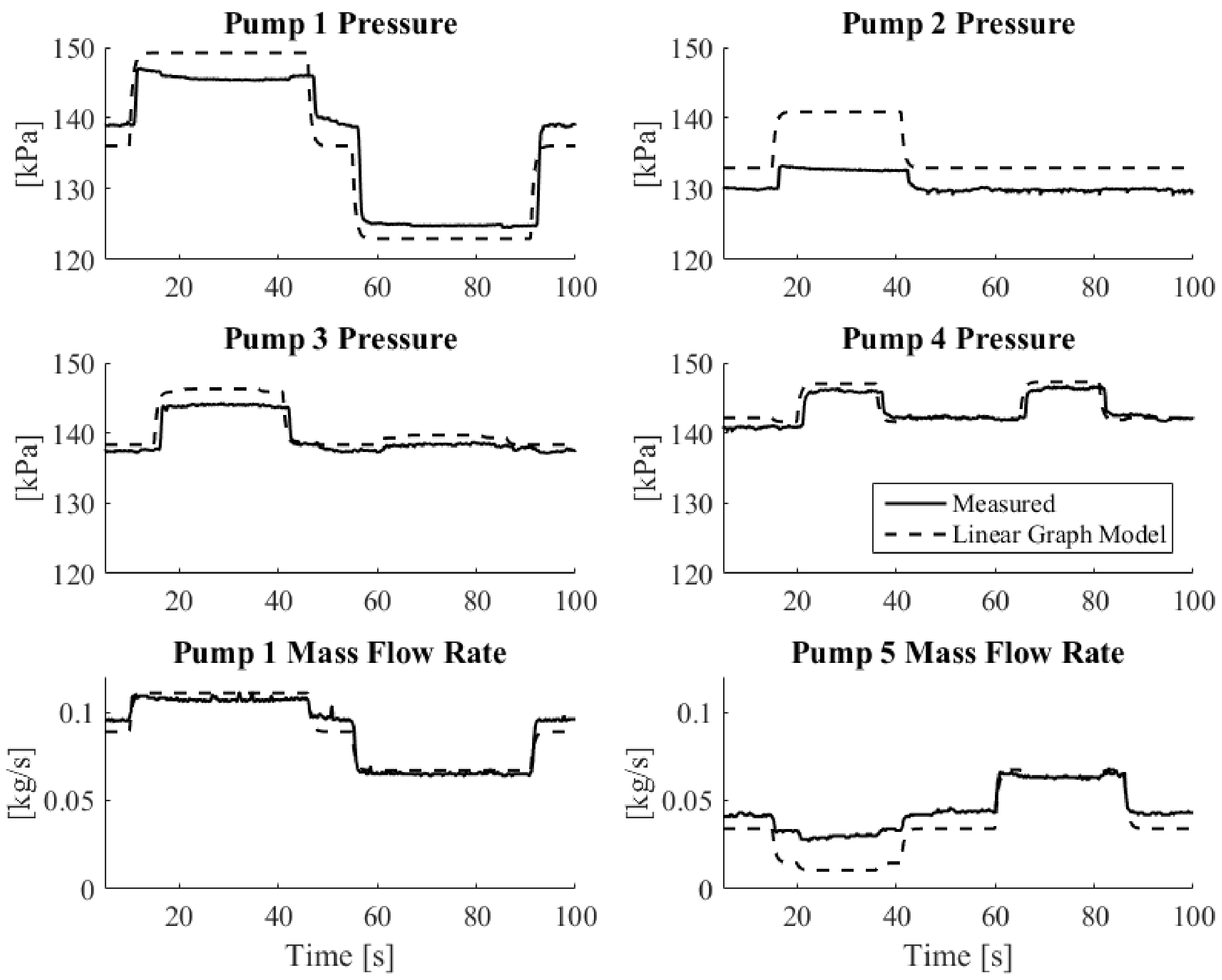


**Figure 6.8 Selected outputs for hydrodynamic validation of experimental data with linear graph-based models.**

## 6.8.2 Thermal Validation

Fig. 6.9 shows a sequence of pump inputs and heat loads to the cold plate walls used to validate the thermodynamics of the models. From Fig. 6.10, one can confirm that general temperature behaviors at multiple locations in the experimental system are captured by the linear model. The model error from the hydraulic graph in the previous Subsection also affects the accuracy of this model since the mass flow rates modeled by the hydraulic graph are inputs to the thermal graph. Discrepancies between the graph-based models and the experimental data are likely due to the lumped capacitance approach used to represent a component with spatially varying temperature, such as the wall of a cold plate heat exchanger, by a single vertex with a single temperature. While these models could be improved, at the cost of increased complexity, the accuracy of the models is sufficient for the hierarchical control in Chapter 7.

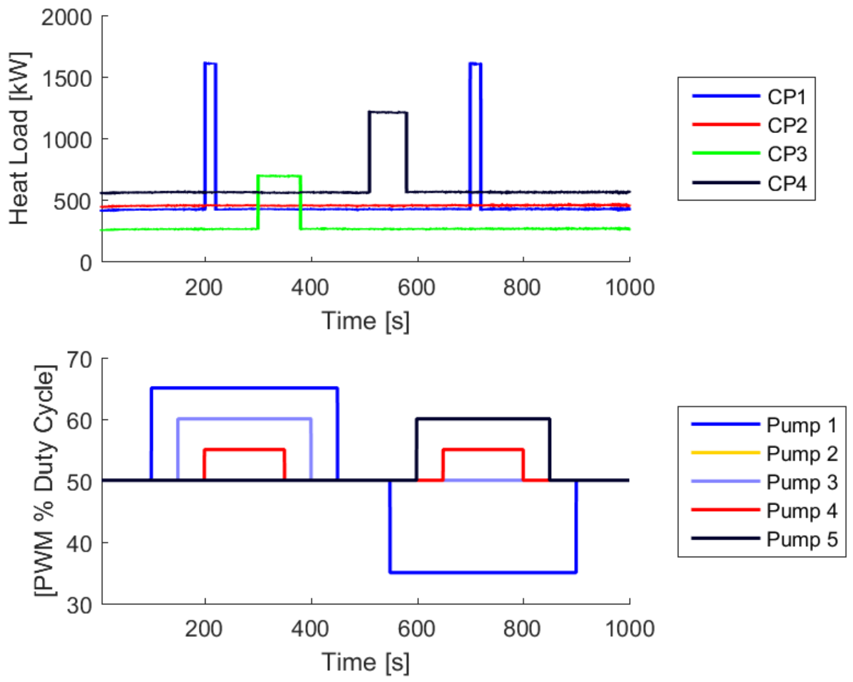


**Figure 6.9 Pump and heater inputs for thermodynamic validation.**

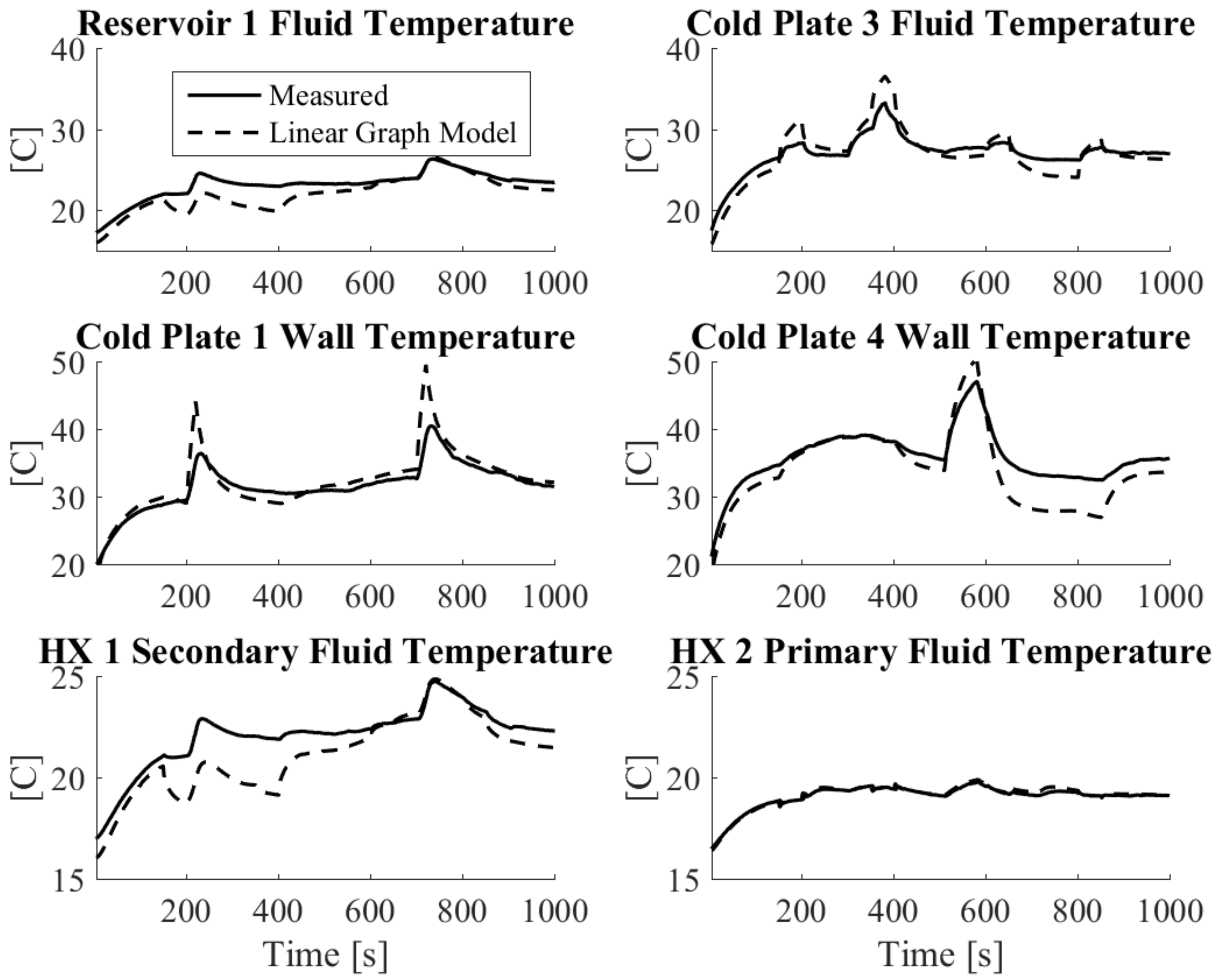


**Figure 6.10 Selected temperatures for thermodynamic validation of experimental data with linear graph-based models.**

## 6.9 Chapter Summary

The results in this Chapter demonstrate the capabilities of a graph-based modeling framework to capture the hydrodynamic and thermodynamic behavior of an experimental thermal fluid system. Conceptualizing and modeling a system based on the underlying structure of mass and energy storage and transport provides numerous benefits. First, when viewed as a graph, systems of different energy domains look and behave identically. Energy, and/or mass, is transported along edges and stored at the vertices, regardless of whether the vertex state represents a temperature, a pressure, or a voltage. This unifying framework natively captures the interactions between energy domains and thus facilitates system-wide design, analysis, and control.

The second benefit of a graph-based approach comes from the modularity. Vertices and edges are all modeled individually. This allows for rapid development of complex systems with many vertices and edges through the combination of components modeled individually. From this modularity, alternative system configurations can be rapidly evaluated through the rearrangement of components or the addition/subtraction of various edges and vertices. Along these lines, if the overall model validity is not sufficient for the intended purposes of the model, additional fidelity can be easily added through the discretization of components captured by additional vertices and edges in the graph.

An additional benefit comes from the flexibility of a graph-based modeling framework. The majority of the system specific behaviors are captured by the edge transfer rate equation (6.2). The general, nonlinear form of this equation allows for a wide variety of relationships to be captured within a single framework. While the general form may be nonlinear, (6.2) may be easily restricted to specific forms, such as input affine, bilinear, or linear, to best suit the needs of the modeling and control efforts. The following Chapter demonstrates the ability to directly utilize the hydraulic and thermal graphs developed in this Chapter to develop a hierarchical MPC controller for the experimental system from Fig. 6.4.

# Chapter 7

# Hierarchical Control of a Thermal Fluid System

## 7.1 Motivation

Chapter 2 demonstrated how a graph-based modeling framework can capture the storage and routing of energy throughout out the complex systems found in vehicles. Chapter 6 showed how an experimental thermal fluid system can be represented as a pair of interacting hydraulic and thermal graphs, which capture the conservation of mass and conservation of thermal energy that govern the complex nonlinear dynamics of a real physical system. This Chapter demonstrates the development of a hierarchical control framework for managing the thermal and hydraulic states of a system by directly accounting for the coupling between these two domains based on the generic modeling and control procedure presented in Chapters 2 and 3. The control hierarchy in this Chapter represents the lower-level Subsystem, Component, and Physical Level controllers from Fig. 3.1. It is intended that the control hierarchy from this Chapter could be readily integrated with the generic hierarchical control formulation from Chapter 3, which would form the upper-level controllers from Fig. 3.1, to create a highly functional controller for more complex, multiple energy domain systems. While the following control formulation does not benefit from the theoretical stability and feasibility guarantees from Chapters 4 and 5, the future research directions in presented in Chapter 8 are intended to help extend the applicability of these theories to physical systems with more complex dynamics.

## 7.2 Hierarchical Control Framework

The proposed control framework consists of three layers, arranged in the hierarchy shown in Fig. 7.1. The thermal layer optimizes the thermal performance of the system by selecting references for its mass flow rates $\dot{m}_{ref}$. In doing so, the thermal layer leverages available preview of upcoming thermal disturbances (i.e., power flows along source edges and states of sink vertices of the thermal graph). The hydraulic layer controls the system mass flow rates to track $\dot{m}_{ref}$ by selecting references for the actuator states $u_{ref}^{\mathbf{m}}$. In the actuator layer, a set of $N_p^{\mathbf{m}}$ decoupled SISO controllers track $u_{ref}^{\mathbf{m}}$ by commanding the actuator inputs $v^{\mathbf{m}}$.

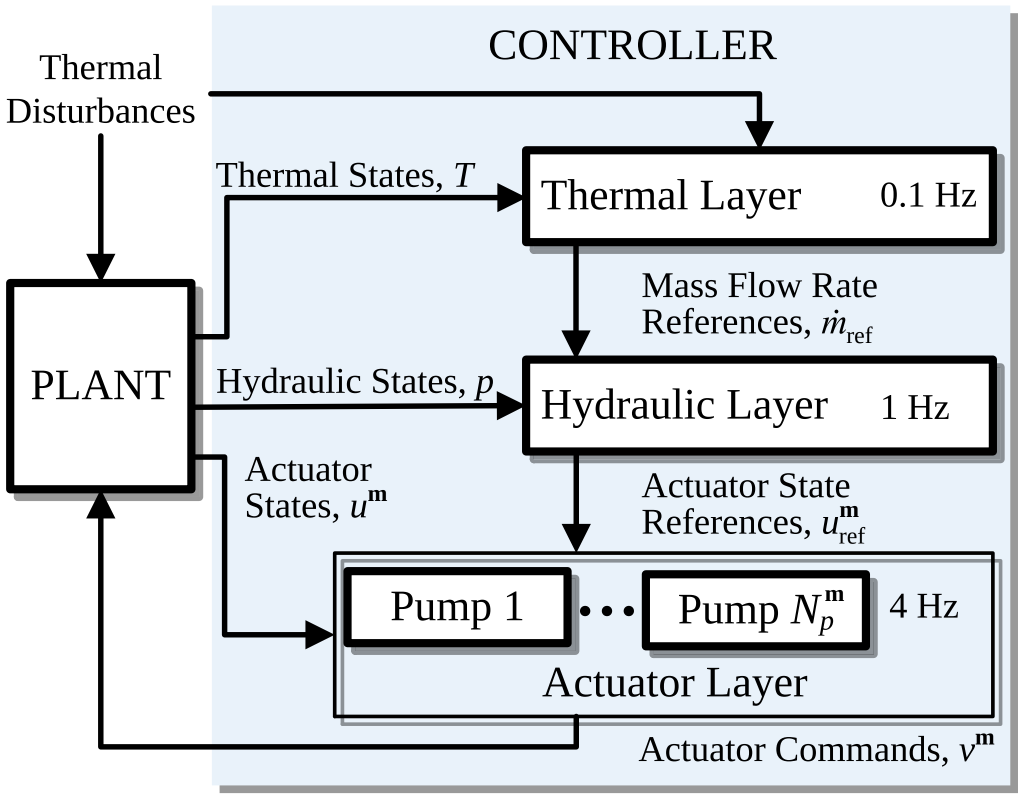

CONTROLLER
Thermal Disturbances
Thermal States, $T$
Thermal Layer
0.1 Hz
Mass Flow Rate References, $\dot{m}_{\text{ref}}$
PLANT
Hydraulic States, $p$
Hydraulic Layer
1 Hz
Actuator States, $u^{\mathbf{m}}$
Actuator State References, $u_{\text{ref}}^{\mathbf{m}}$
Pump 1
Pump $N_p^{\mathbf{m}}$
4 Hz
Actuator Layer
Actuator Commands, $v^{\mathbf{m}}$

**Figure 7.1 Three-level graph-based control hierarchy and signals.**

### 7.2.1 Thermal Control Layer

The thermal control layer leverages available preview of upcoming thermal disturbances in selecting references for system mass flow rates $\dot{m}_{ref}$ that optimize the thermal performance of the system over a prediction horizon. For this Chapter, “optimal thermal performance” primarily involves controlling temperature states of the system $T_i,\ i \in [1, N_v^{\mathbf{e}}]$ such that $\underline{T}_i \le T_i \le \overline{T}_i,\ \forall i$, where $\underline{T}_i$ and $\overline{T}_i$ are lower and upper bounds, respectively, on the $i^{th}$ temperature. A temperature regulation objective is also included to maintain critical components near the desired operating temperature. A final objective is minimizing the mass flow rate references, which reduces the actuator effort required of the system. Additional constraints are included in the thermal control layer to ensure that $\dot{m}_{ref}$ is an achievable reference to be tracked by the hydraulic control layer by the system. The MPC controller at the thermal control layer solves the constrained quadratic program

$$\min_{\dot{m}_{ref}(\cdot),s^{\mathbf{e}}(\cdot)} \sum_{k=0}^{N_h^{\mathbf{e}}-1} \left(\lambda_s^{\mathbf{e}} \| s^{\mathbf{e}}(k) \|_2^2 + \lambda_r^{\mathbf{e}} \| T(k) - T_{des} \|_2^2 + \lambda_{\dot{m}}^{\mathbf{e}} \| \dot{m}_{ref}(k) \|_2^2 \right) \tag{7.1a}$$

$$s.t. \qquad \Delta T(k+1) = A_d^{\mathbf{e}} \Delta T(k) + B_d^{\mathbf{e}} \Delta \dot{m}_{ref}(k) + V_{d,1}^{\mathbf{e}} \Delta P_{preview}^{in}(k) + V_{d,2}^{\mathbf{e}} \Delta T_{preview}^{t}(k), \tag{7.1b}$$

$$\underline{T}_i - s_i^{\mathbf{e}}(k) \le T_i(k) \le \overline{T}_i + s_i^{\mathbf{e}}(k), \quad i \in [1, N_v^{\mathbf{e}}], \tag{7.1c}$$

$$s_i^{\mathbf{e}}(k) \ge 0, \quad i \in [1, N_v^{\mathbf{e}}], \tag{7.1d}$$

$$\underline{\dot{m}} \le \dot{m}_{ref,i}(k), \quad i \in [1, N_e^{\mathbf{m}}], \tag{7.1e}$$

$$\underline{u}_i^{\mathbf{m}} \le u_i^{\mathbf{m}}(k) \le \overline{u}_i^{\mathbf{m}}, \quad i \in [1, N_p^{\mathbf{m}}], \tag{7.1f}$$

$$0 = (A_{d,fast}^{\mathbf{m}} - I)\Delta p(k) + B_{d,fast}^{\mathbf{m}} \Delta u^{\mathbf{m}}(k), \tag{7.1g}$$

$$\dot{m}_{ref}(k) = C^{\mathbf{m}} \Delta p(k) + D^{\mathbf{m}} \Delta u^{\mathbf{m}}(k), \tag{7.1h}$$

for $k \in \left[0, N_h^{\mathbf{e}} - 1\right]$.

In the above MPC optimization problem, $N_h^{\mathbf{e}}$ is the prediction horizon of the thermal control layer. The cost function (7.1a) minimizes the thermal slack variable $s^{\mathbf{e}} = \left[ s_i^{\mathbf{e}} \right] \in \mathbb{R}^{N_v^{\mathbf{e}}}$

with weighting $\lambda_s^{\mathbf{e}}$, the desired state tracking error $T - T_{des}$ with weighting $\lambda_r^{\mathbf{e}}$, and the mass flow rate references $\dot{m}_{ref}$ with weighting $\lambda_{\dot{m}}^{\mathbf{e}}$. The discretized dynamics from the thermal graph model are imposed by (7.1b). The temperature states $T_i$ are constrained by (7.1c) where the slack variables are required to be non-negative by (7.1d). All mass flow rates are required to be greater than $\underline{\dot{m}}$ by (7.1e), where $\underline{\dot{m}}$ is a small positive number to prevent reverse flow conditions.

Constraints (7.1f)-(7.1h) ensure that the set of reference values $\dot{m}_{ref}$ are simultaneously achievable by $\dot{m}$ of the hydraulic graph model at steady-state. This represents a key difference between the graph-based control designs in previous Chapters in which generic inputs are individually constrained as $\underline{u}_i \le u_i \le \overline{u}_i,\ \forall i$. In hydraulic fluid systems with flow splits and junctions, there exists a high degree of coupling among pressure and mass flow rates through different flow paths. This means that constraining individual mass flow rates by constants can either result in over-conservative bounds or result combinations of mass flow references that are not simultaneously achievable by the system. The use of the steady-state hydraulic model in (7.1g) greatly reduces this conservatism.

The upper and lower bound constraints on the actuator inputs are enforced by (7.1f). Using the steady-state hydraulic model, (7.1g) constraints the relationship between system pressures and actuator inputs that are used to calculate the mass flow rates in (7.1h). As will be shown in Section 7.3, due to the significant timescale separation between the thermodynamics and hydrodynamics, there is negligible error incurred by using a steady-state hydraulic model to make control decisions at the timescale of the thermal control layer.

### 7.2.1 Hydraulic Control Layer

The hydraulic control layer forces the system mass flow rates to track $\dot{m}_{ref}$ by selecting references for the actuator states $u_{ref}^{\mathbf{m}}$. While (7.1f)-(7.1h) in the thermal control layer ensure that $\dot{m}_{ref}$ is achievable at steady-state, the hydraulic control layer is responsible for managing the transient behavior of the hydraulic states to closely track $\dot{m}_{ref}$, to minimize the prediction error of the thermal controller. Because the hydrodynamics evolve significantly faster than the thermodynamics, the hydraulic control layer has an order of magnitude faster update rate than the thermal control layer.

References $\dot{m}_{ref}$ from the thermal control layer are not provided for a single instant in time, but instead are provided for all steps over the thermal control layer's prediction horizon. Using a zero-order hold between steps of the thermal horizon, these references are resampled and truncated to match the update rate and prediction horizon of the hydraulic control layer, giving $\dot{m}_{ref,preview}$. This allows the hydraulic layer to take preemptive action in minimizing tracking errors, preparing for references anticipated of the future rather than only reacting to their current values.

The MPC controller at the hydraulic control layer solves the constrained quadratic program

$$\min_{u_{ref}(\cdot),s^{\mathbf{m}}(\cdot)} \sum_{k=0}^{N_h^{\mathbf{m}}-1} \left(\lambda_s^{\mathbf{m}} \| s^{\mathbf{m}}(k) \|_2^2 + \lambda_u^{\mathbf{m}} \| u_{ref}^{\mathbf{m}}(k) \|_2^2 + \lambda_{\dot{m}}^{\mathbf{m}} \| \dot{m}^{\mathbf{m}}(k) - \dot{m}_{ref,preview}(k) \|_2^2 \right) \quad (7.2a)$$

$$s.t. \quad \Delta p(k+1) = A_d^{\mathbf{m}} \Delta p(k) + B_{d,1}^{\mathbf{m}} \Delta u_{ref}^{\mathbf{m}}(k) + B_{d,2}^{\mathbf{m}} \Delta u_{ref}^{\mathbf{m}}(k+1), \quad (7.2b)$$

$$\dot{m}(k) = C^{\mathbf{m}} \Delta p(k) + D^{\mathbf{m}} \Delta u_{ref}^{\mathbf{m}}(k), \quad (7.2c)$$

$$\underline{p}_i - s_i^{\mathbf{m}}(k) \le p_i(k) \le \bar{p}_i + s_i^{\mathbf{m}}(k), \quad i \in [1, N_v^{\mathbf{m}}], \quad (7.2d)$$

$$s_i^{\mathbf{m}}(k) \ge 0, \quad i \in [1, N_v^{\mathbf{m}}], \quad (7.2e)$$

$$\underline{u}_i^{\mathbf{m}} \le u_{ref,i}^{\mathbf{m}}(k) \le \bar{u}_i^{\mathbf{m}}, \quad i \in [1, N_p^{\mathbf{m}}], \quad (7.2f)$$

$$u_{ref}^{\mathbf{m}}(1) = u_{ref,last}^{\mathbf{m}}(2), \quad (7.2g)$$

for $k \in \left[0, N_h^{\mathbf{m}} - 1\right]$.

In the above MPC optimization problem, $N_h^{\mathbf{m}}$ is the prediction horizon of the hydraulic control layer. The cost function (7.2a) minimizes the thermal slack variable $s^{\mathbf{m}} = \left[ s_i^{\mathbf{m}} \right] \in \mathbb{R}^{N_v^{\mathbf{m}}}$

with weighting $\lambda_s^{\mathbf{m}}$, the actuator state references $u_{ref}^{\mathbf{m}}$ with weighting $\lambda_u^{\mathbf{m}}$, and the mass flow rate reference tracking error with weighting $\lambda_{\dot{m}}^{\mathbf{m}}$. The discretized dynamics from the hydraulic graph model are imposed by (7.2b) with mass flow rate related to pressure and actuator state references through (7.2c). The pressure states $p_i$ are constrained by (7.2d) where the slack variables are required to be non-negative by (7.2e). The upper and lower bound constraints on the actuator inputs are enforced by (7.2f). Since (7.2b) is non-causal, (7.2g) constrains the actuator state references at the current time step $u_{ref}^{\mathbf{m}}(1)$ to be equal to the actuator state references at the corresponding time in the previous time step $u_{ref,last}^{\mathbf{m}}(2)$.

## 7.2.1 Actuator Control Layer

In the actuator control layer, a set of $N_p^{\mathbf{m}}$ decoupled SISO controllers track $u_{ref}^{\mathbf{m}}$ by commanding the actuator inputs $v^{\mathbf{m}}$, accounting for the dynamics of the actuators.

Similar to $\dot{m}_{ref}$ from the thermal control layer as discussed previously, references $u_{ref}^{\mathbf{m}}$ are not provided for a single instant in time, but instead are provided for all steps over the hydraulic control layer's prediction horizon. Using a zero-order hold between steps of the hydraulic horizon, these references are resampled and truncated to match the update rate and prediction horizon of the actuator control layer, giving $u_{ref,preview}^{\mathbf{m}}$.

The $i^{th}$ MPC controller at the actuator control layer solves the constrained quadratic program

$$\min_{v_{delayed,i}^{\mathbf{m}}(\cdot)} \quad \sum_{k=0}^{N_{h,i}^{\mathbf{p}}-1} \lambda_{u,i}^{\mathbf{p}} \| u_i^{\mathbf{m}}(k) - u_{ref,preview,i}^{\mathbf{m}}(k) \|_2^2 + \sum_{k=N_{ds,i}^{\mathbf{p}}}^{N_{h,i}^{\mathbf{p}}-1} \lambda_{dv,i}^{\mathbf{p}} \| v_{delayed,i}^{\mathbf{m}}(k) - v_{delayed,i}^{\mathbf{m}}(k-1) \|_2^2 \tag{7.3a}$$

$$s.t. \quad u_i^{\mathbf{m}}(k+1) = a_{d,i}^{\mathbf{p}} u_i^{\mathbf{m}}(k) + b_{d,i}^{\mathbf{p}} v_{delayed,i}^{\mathbf{m}}(k), \ k \in \left[0, N_{h,i}^{\mathbf{p}} - 1\right], \tag{7.3b}$$

$$\underline{u}_i^{\mathbf{m}} \le v_{delayed,i}^{\mathbf{m}}(k) \le \bar{u}_i^{\mathbf{m}}, \quad k \in [N_{ds,i}^{\mathbf{p}}, N_{h,i}^{\mathbf{p}} - 1], \tag{7.3c}$$

$$v_{delayed,i}^{\mathbf{m}}(k) = v_{delayed,last,i}^{\mathbf{m}}(k + N_{ds,i}^{\mathbf{p}}), \;\; k \in \left[0, N_{ds,i}^{\mathbf{p}} - 1\right]. \tag{7.3d}$$

In the above MPC optimization problem, $N_{h,i}^{\mathbf{p}}$ is the prediction horizon of the $i^{th}$ MPC controller at the actuator control layer. The cost function (7.3a) minimizes the actuator state reference tracking error with weighting $\lambda_{u,i}^{\mathbf{p}}$ and the change in consecutive actuator inputs with weighting $\lambda_{dv,i}^{\mathbf{p}}$ over the portion of the prediction horizon for which $v_{delayed,i}^{\mathbf{m}}$ is not fixed to equal the actuator inputs $v_{delayed,last,i}^{\mathbf{m}}$ determined at the previous iterations of the controller by (7.3d). The discretized dynamics from the actuator model are imposed by (7.3b). The upper and lower bound constraints on the actuator inputs are enforced by (7.3c).

## 7.3 Simulation Results

In this Section, the proposed hierarchical control framework is demonstrated in simulation. The controller is provided with full state feedback of the temperatures, pressures, and actuator states of the plant. The update rates, horizons, weightings, and constraints of the control framework as described in the previous Section are parametrized as follows:

Thermal Layer:
$$\begin{aligned}
&\Delta t^{\mathbf{e}} = 10s, \quad N_h^{\mathbf{e}} = 10,\\
&\lambda_s^{\mathbf{e}} = 10^6, \quad \lambda_r^{\mathbf{e}} = 10^2, \quad \lambda_{\dot{m}}^{\mathbf{e}} = 10^{-1},\\
&\underline{T}_i = 15^o C, \quad \bar{T}_i = 40^o C, \quad \forall i \in \left[1, N_v^{\mathbf{e}}\right],\\
&\underline{\dot{m}} = 0.03 kg/s,\\
&\underline{u}_i^{\mathbf{m}} = 20\%, \quad \bar{u}_i^{\mathbf{m}} = 65\%, \quad \forall i \in \left[1, N_p^{\mathbf{m}}\right],
\end{aligned} \tag{7.4a}$$

Hydraulic Layer:
$$\begin{aligned}
&\Delta t^{\mathbf{m}} = 1s, \quad N_h^{\mathbf{m}} = 15,\\
&\lambda_s^{\mathbf{m}} = 10^0, \quad \lambda_u^{\mathbf{m}} = 0, \quad \lambda_{\dot{m}}^{\mathbf{m}} = 10^4,\\
&\underline{p}_i = 50kPa, \quad \bar{p}_i = 200kPa, \quad \forall i \in \left[1, N_v^{\mathbf{m}}\right],\\
&\underline{u}_i^{\mathbf{m}} = 20\%, \quad \bar{u}_i^{\mathbf{m}} = 65\%, \quad \forall i \in \left[1, N_p^{\mathbf{m}}\right],
\end{aligned} \tag{7.3b}$$

$$\text{Actuator Layer:} \quad \Delta t^{\mathbf{p}} = 0.25s, \quad N_h^{\mathbf{p}} = 10,$$
$$\lambda_{u,i}^{\mathbf{p}} = 10^0, \quad \lambda_{dv,i}^{\mathbf{p}} = 10^0, \quad \forall i \in \left[1, N_p^{\mathbf{m}}\right], \tag{7.3b}$$
$$\underline{u}_i^{\mathbf{m}} = 20\%, \quad \bar{u}_i^{\mathbf{m}} = 65\%, \quad \forall i \in \left[1, N_p^{\mathbf{m}}\right].$$

Simulations are conducted in MATLAB/Simulink using the YALMIP toolbox [41] and Gurobi optimization suite [42] to solve the constrained quadratic programs.

Fig. 7.2 shows the heat load inputs to cold plates 1-5 for the simulation example. These loads serve as disturbances to the system. Upcoming loads are assumed to be known exactly by the thermal control layer over the duration of its prediction horizon, equal to $\Delta t^{\mathbf{e}} N_h^{\mathbf{e}} = 100s,$. The load profile consists of sequential steps of varying magnitude and duration in the heat load to each cold plate.

Fig. 7.3 shows temperature states of the linear plant, including a selection of cold plate wall temperatures in the top subplot and fluid temperatures in the bottom subplot. In advance of the heat loads to cold plates 1 and 4, the thermal controller seeks to minimize temperature constraint violation by strategically "precooling" the cold plate walls prior to the increase in load. While this precooling helps to minimize the constraint violation, the loads are large enough to cause constraint violations. The coupling between cold plates 2 and 3 is shown during the increase in heat load to cold plate 3 between 300 and 400 seconds. Since these two cold plates are in series, the controller increases the mass flow rate through both cold plates to prevent constraint violation for cold plate 3. This increase in mass flow rate causes the temperature of cold plate 2 to decrease.

Fig. 7.4 shows the resulting pressures, mass flow rates, and pump speed inputs. Interestingly, the hierarchical controller chooses to reduce the speed of pump 1 during the increase heat load to cold plate 4 between 500 and 600 seconds. This reduces the heat transfer from the secondary loop to the primary loop, which helps minimize the constraint violation for the cold plate 4 wall temperature.

Fig. 7.5 shows a closer view of a portion of the actuator effort for pump 5, and also includes the references from the Hydraulic Control Layer and the commands issued by the Actuator Control Layer. The commands are seen to lead the references, leveraging preview of upcoming references and accounting for the 1 second time delay in pump dynamics to minimize the overall tracking error.

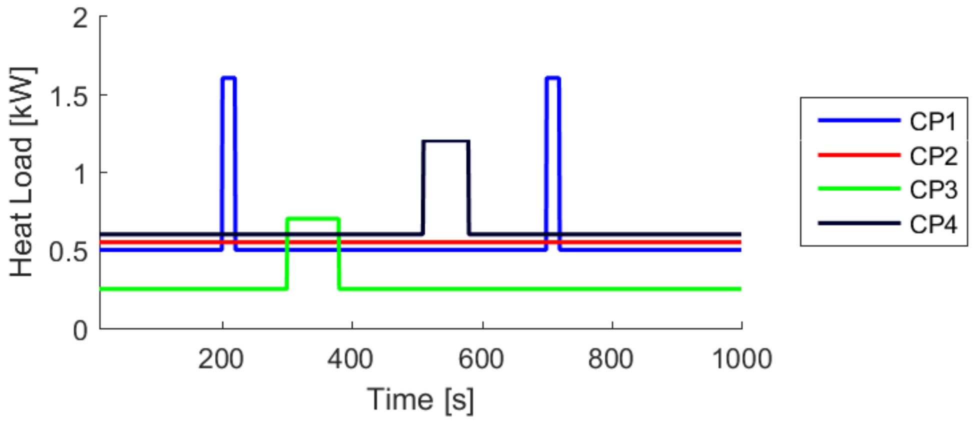


**Figure 7.2 Thermal disturbances for the simulation example, consisting of step changes in heat load to each cold plate (CP) heat exchanger.**

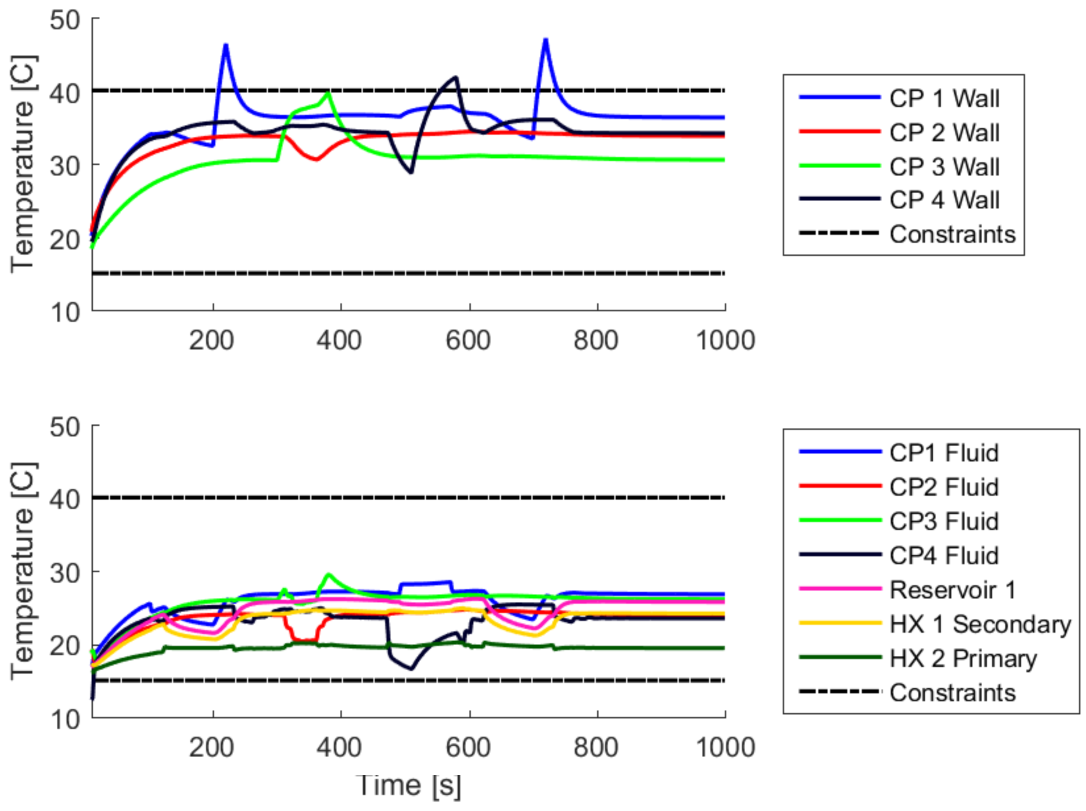


**Figure 7.3 States of the linear plant in the closed-loop simulation example, including cold plate wall temperatures (top) and a selection of outlet fluid temperatures (bottom).**

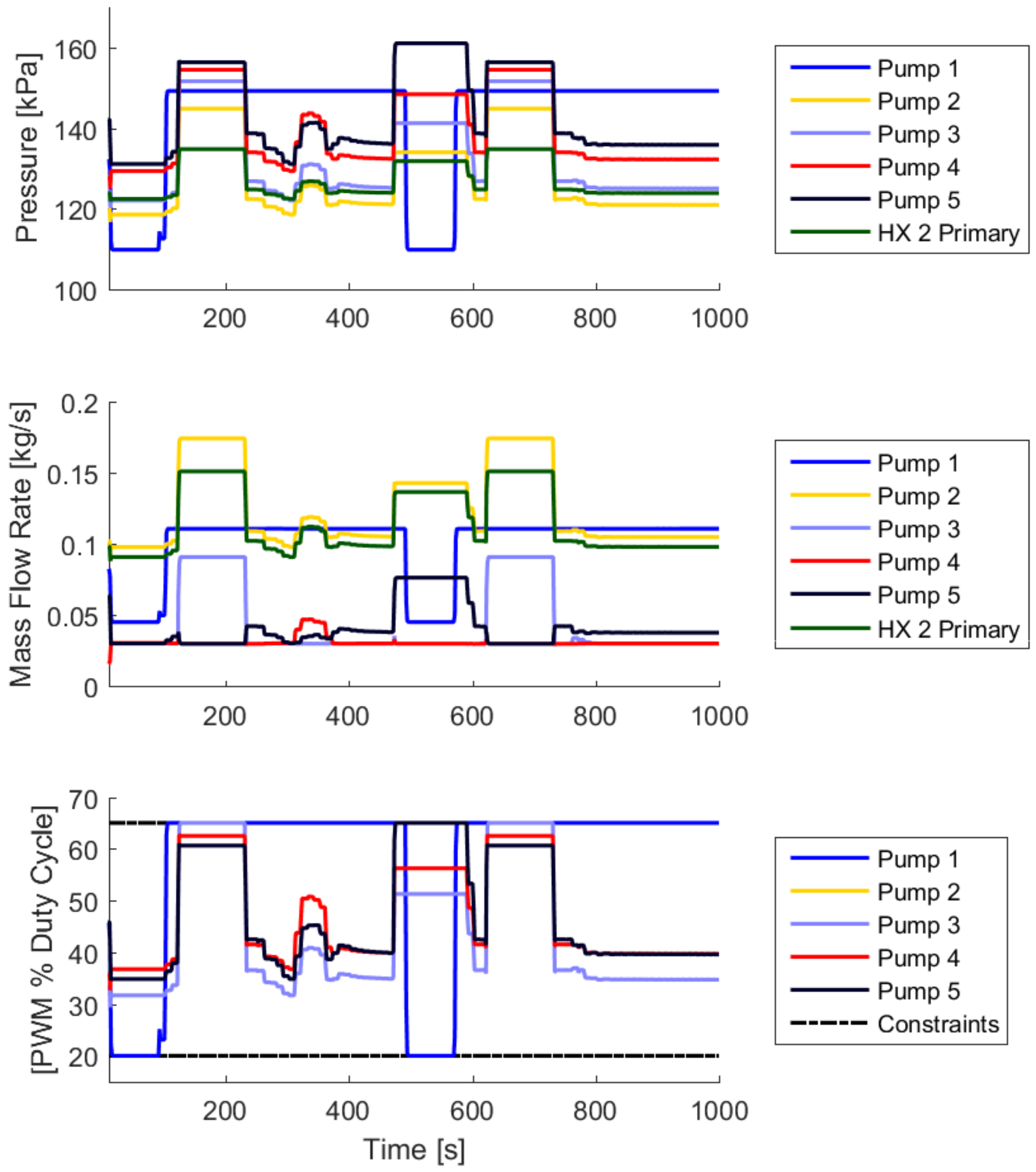


**Figure 7.4 Fluid pressures at the outlet of each pump and inlet to heat exchanger 2 (top), mass flow rates at the outlet of each pump and inlet to heat exchanger 2 (middle), and pump input signals (bottom).**

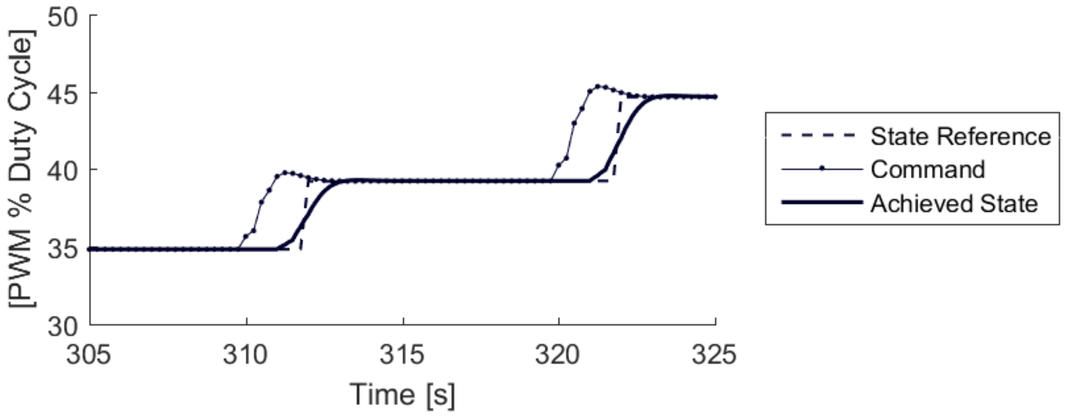


**Figure 7.5 Close-up view of state signals for pump 5 showing the pump state reference from the Hydraulic Control Layer, the commanded pump input by the Actuator Control Layer, and the achieved pump state.**

## 7.4 Experimental Results

The same hierarchical controller used to control a linear graph model of the system in the previous Section is applied to the experimental system. The same controller parameters from (7.4a-c) are used. A Kalman filter estimates the full set of states based on the subset of measured pressures and temperature. Fig. 7.6 shows the measured heat load applied to each cold plate heat exchanger based on the measured current and the measured resistance of the resistors from Fig. 6.5. Fig. 7.7 shows the measured temperatures corresponding to the temperatures from the linear simulation presented in Fig. 7.3. There are some clear differences between the measured experimental and simulated temperatures, most notably the wall temperature of cold plate 1. The fact that this cold plate temperature increases significantly more in the closed-loop simulation results than in the closed-loop experimental results is a product of the open-loop model error seen in Fig. 6.10 from the previous Chapter, where the linear model predicts larger temperature rises as a result of the increased heat load. This error is likely due to the approximated heat transfer coefficients in the linear graph model used to capture the heat transfer between the cold plate wall and the fluid flowing through the cold plate. The over-prediction of cold plate wall temperatures in the linear model also results in differences in the control of pump 1, as seen by comparing Fig. 7.8 with Fig. 7.4. In simulation, this pump is operated at its upper constraint in simulation and at its lower constraint on the experimental system for most of the scenario.

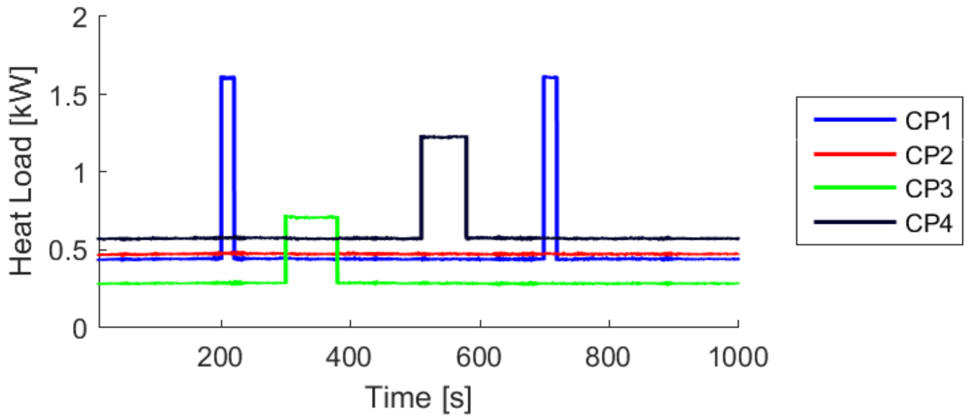


**Figure 7.6 Measured heat load applied to each cold plate matching the disturbance profile from Fig 7.2.**

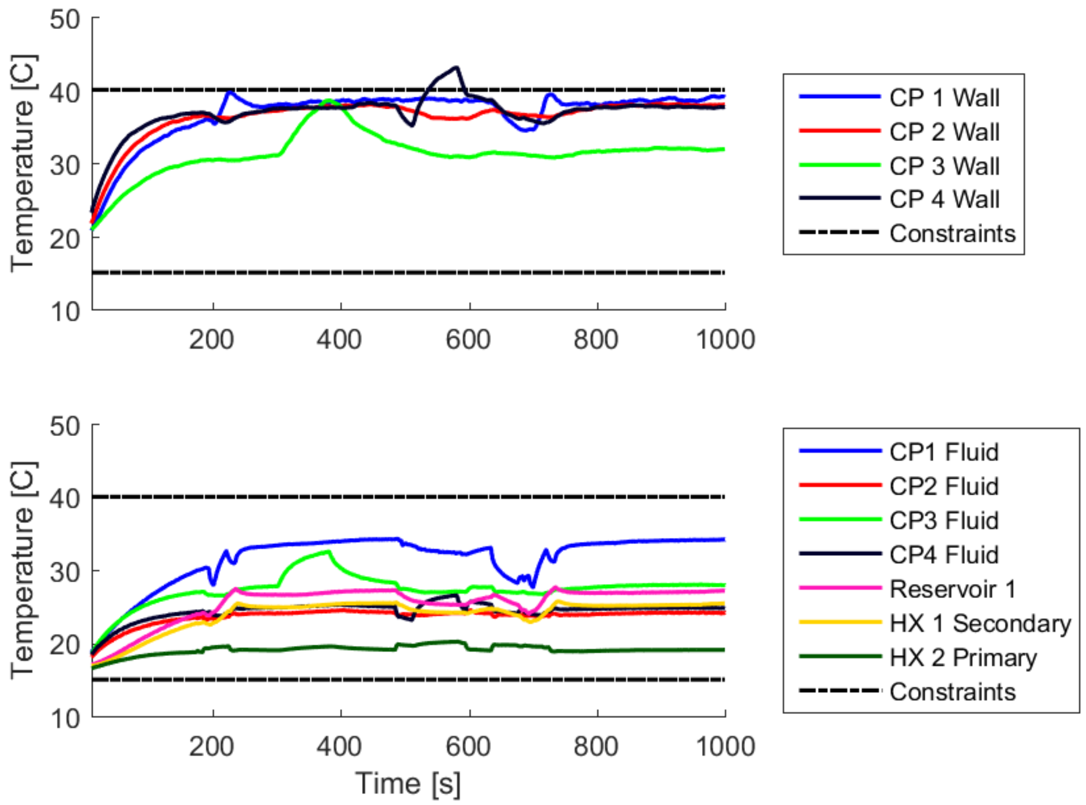


**Figure 7.7 Measured temperatures from closed-loop control of the experimental system, including cold plate wall temperatures (top) and a selection of outlet fluid temperatures (bottom).**

Despite these difference, the linear model-based hierarchical controller is able to effectively controller the nonlinear experimental system. As with the simulated results in the previous Section, the thermal controller seeks to minimize temperature constraint violation by strategically precooling the cold plate walls prior to the increase in load, as seen by the temperatures of cold plates 1 and 4.

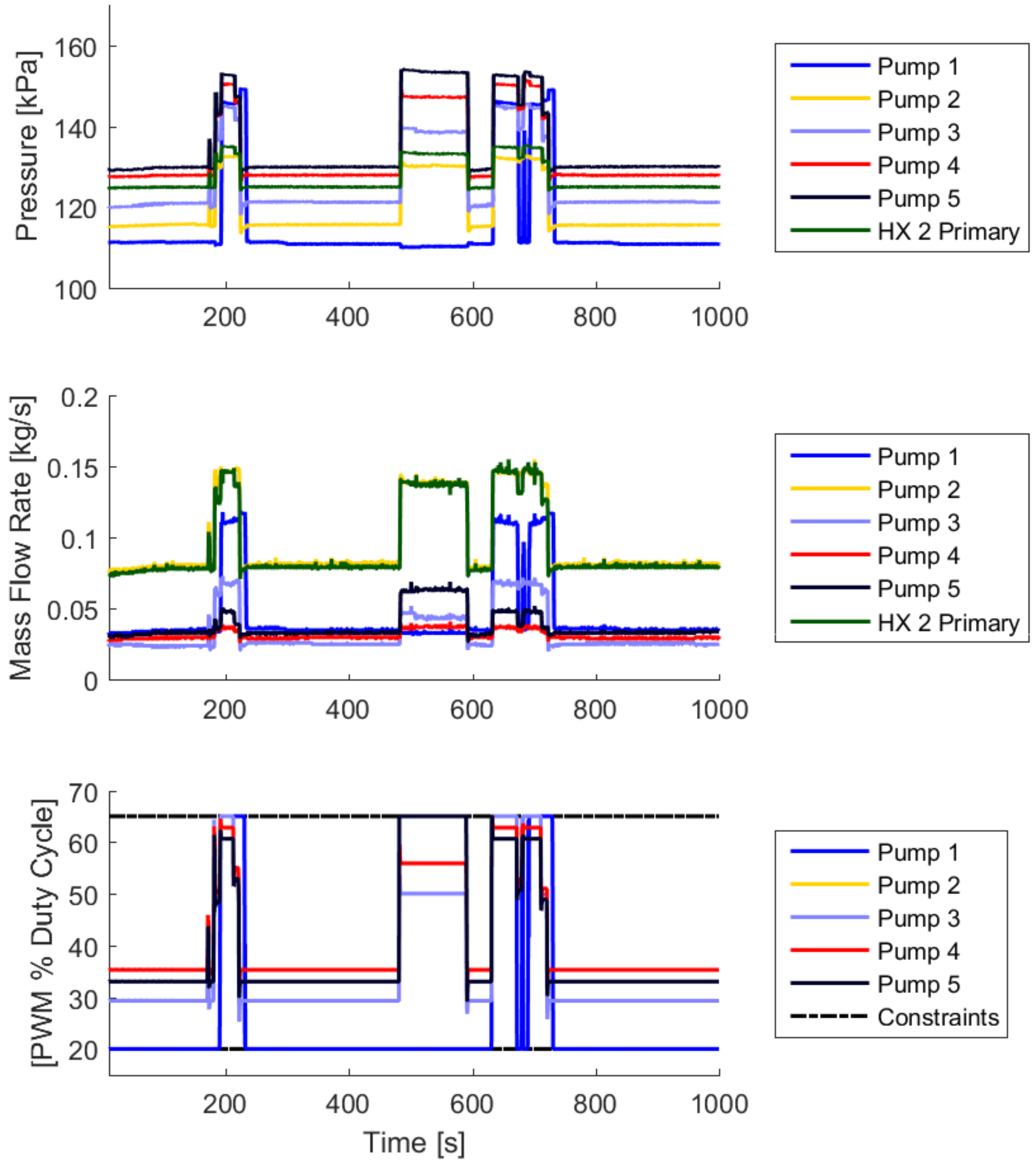


**Figure 7.8 Measured fluid pressures at the outlet of each pump and inlet to heat exchanger 2 (top), measured mass flow rates at the outlet of each pump and inlet to heat exchanger 2 (middle), and pump input signals (bottom).**

## 7.5 Chapter Conclusions

The results in this Chapter demonstrate the ability to develop a practical hierarchical controller based on linear graph models for an experimental system without guaranteed stability or robust feasibility. The proposed control framework shows how hydraulic and thermal graph models, developed and validated in Chapter 6, can be used by controllers at different levels of the hierarchy. With the three levels of control, the thermal, hydraulic, and actuator dynamics can be effectively controlled, despite the timescale separation between these dynamics. These controllers form the lowest levels of the larger control hierarchy shown in Fig. 3.1. Within the overall hierarchy, these controllers have the difficult task of compensating for the nonlinearity, sensor noise, model mismatch, and time delays common to many physical systems. Thus, the preliminary experimental results in this Chapter demonstrate the potential of model-based hierarchical control in practice. In addition to summarizing of the contribution of this dissertation, the following Chapter discusses several practical extensions of the theoretical result from Chapters 5 and 6 that would help create a highly functional controller for more complex, multiple energy domain systems with guaranteed stability and robust feasibility.

# Chapter 8
# Conclusion

## 8.1 Summary of Research Contributions

This dissertation develops, analyzes, and demonstrates a hierarchical control framework for energy management in vehicle systems. Effective energy management is vital to maximizing the capability of these complex systems to meet the constantly growing demands for performance, efficiency, and reliability. With multiple systems and subsystems of various energy domains interacting over a wide range of timescales, these vehicle systems require both modeling and control frameworks that are *widely applicable*, *scalable*, *robust*, *high performance*, and *computationally efficient*.

This need is addressed through contributions in the following four areas.

1. Chapter 2 develops a generic graph-based modeling framework that captures the energy storage and power flow dynamics in multiple energy domains and timescales. Thus, this modeling approach is *widely applicable* to many types of systems and is *scalable* to large systems due to the modularity of graph-based modeling. The approach is also *computationally efficient*, using a relatively few number of states to capture the complex dynamics of the energy storage and routing throughout a system.
2. Chapter 3 utilizes these graph-based models to develop a multi-level hierarchical control framework, where each level consists of multiple MPC-based controllers. The graph-based modeling framework is directly used to formulate the structure of the control hierarchy as well as the models used by the controllers at each level

of the hierarchy via a novel graph-based model reduction technique. Simulation results demonstrate the *high performance* achieved when the update rate of controllers at each level of the hierarchy is paired with a particular timescale of the multi-timescale system. Through model reduction and large prediction horizons with relatively few prediction steps, the hierarchical controller achieves this performance with high *computational efficiency*.

3. Chapters 4 and 5 analyze the theoretical properties of the proposed hierarchical controller with respect to stability and robust feasibility. Chapter 4 presents a *widely applicable* and *scalable* procedure for augmenting a hierarchical controller with simple, local constraints based on passivity that guarantee closed-loop stability of the system. Chapter 5 presents a hierarchical control framework for linear graph-based power flow systems that is *robust* to model and disturbance signal uncertainty. This control formulation achieves *high performance* by maintaining critical state and actuator constraints while achieving system-specific objectives.
4. Chapters 6 and 7 demonstrate the practical application of both the graph-based modeling and hierarchical control frameworks through application to an experimental thermal fluid system. These Chapters prove that the proposed approaches are *widely applicable*; being capable of capturing the complex dynamics of a real-world system that includes nonlinearity, unknown disturbances, and time delays.

In conclusion, this dissertation shows that graph-based modeling and hierarchical control are promising approaches to energy management onboard vehicles, worthy of continued development both in theory and application.

## 8.2 Future Work

This dissertation presents the *initial* formulation, analysis, and implementation of a hierarchical control framework for energy management in vehicle systems and future work should build off this foundation through advancements in theory and application. Fig. 1.3, reshown as Fig. 8.1, provides an outline for this future research.

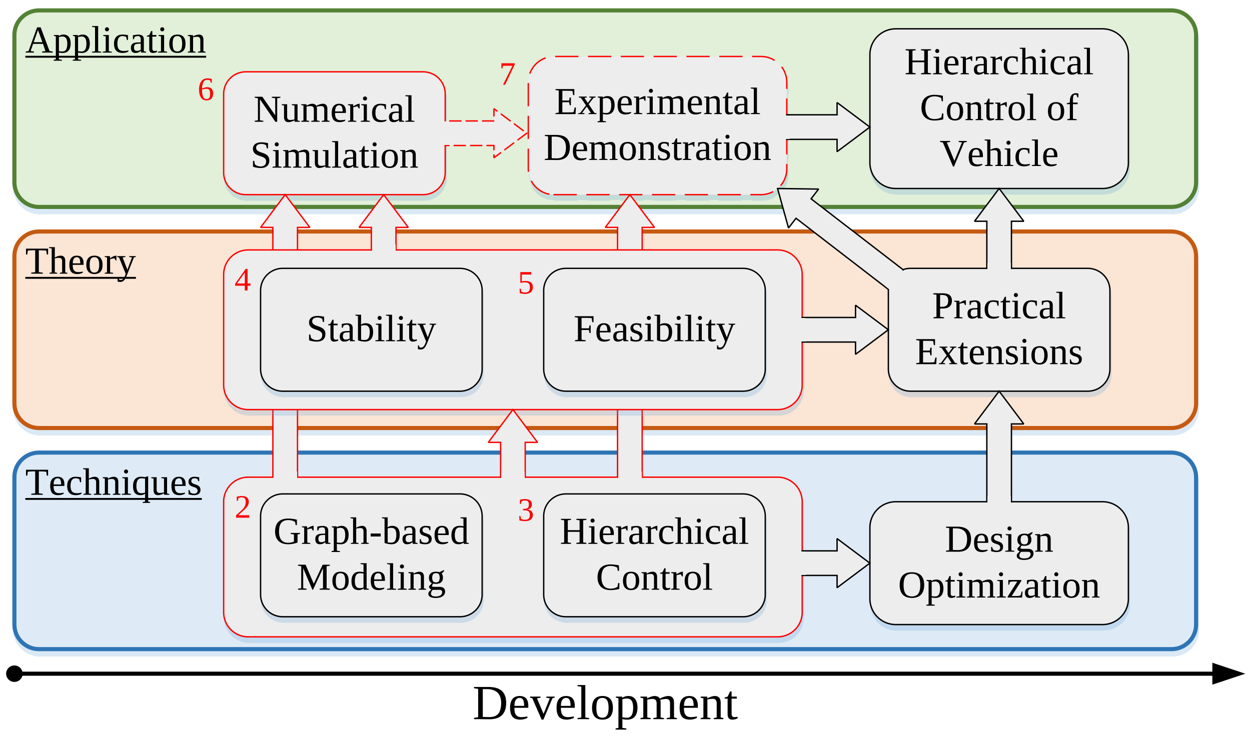


**Figure 8.1 Outline of developments required for the realization of hierarchical control of power flow in vehicle systems.**

### 8.2.1 Techniques

Several aspects of the generic graph-based modeling and hierarchical control development procedures presented in Chapters 2 and 3 should be studied further to make these approaches more practical for modeling vehicle systems.

1. Further modeling and validation should be performed for representing systems as graphs in other energy domains, electrical systems in particular. For these systems, it will become vital to incorporate the ability to represent discrete actuator inputs such as the on/off of a switch.
2. Novel methods for integrating graph models based on different conserved quantities should be developed. In Chapter 6, hydraulic and thermal graph models are developed based on conservation of mass and energy, respectively. A method for combining these graphs into a single system representation would enable the

hierarchical control methods developed in Chapters 3-5 to be more directly applied to systems with more than one conserved quantity.

3. Methods for system decomposition, both spatially and temporally, should be developed specifically for hierarchical control. The effects of system and subsystem boundaries should be investigated as well as the designation of timescales when vertex capacitances may not be obviously grouped.
4. The tradeoff between computational demand and control performance should be analyzed to provide guidance for the choice of controller update rates and prediction horizons at each level of the hierarchy.
5. Cost function design for each level of the hierarchy should be investigated to better understand how local objectives can be designed to achieve global objectives for the vehicle.

Numbers 3-5 are a few examples of the Design Optimization efforts shown in Fig. 8.1 that are needed to improve the performance of the hierarchical control approach and maximize the capability of the vehicle.

### 8.2.2 Theory

Within the classes of graph-based power flow systems considered in Chapters 4 and 5, there are numerous aspects of hierarchical control that warrant additional attention from a theoretical perspective.

1. The passivity-based approach used to establish stability in Chapter 4, like with most passivity approaches, suffers from conservatism that could detract from the performance of the overall hierarchical controller. Understanding this conservatism would be valuable, especially in relation to the common control behavior of “precooling,” where states are purposely driven away from their equilibrium to accommodate larger future power flows through the system.
2. The proposed robust hierarchical controller in Chapter 5 could be improved and analyzed in a number of ways.

a. Currently, known disturbances can only change at the rate of the slowest controller at the top of the hierarchy. Allowing intersample disturbance changes would greatly improve the practicality of this approach.

b. The feedback integralization approach requires an actuator input along each edge of the graph. Extending this method to systems with edges that do not have a dedicated actuator would significantly improve the applicability of this approach.

c. The proposed approach requires desired state trajectories and power flows to be tracked perfectly by lower level controllers. Allowing imperfect tracking may result in improved overall system-level control performance and should be investigated.

d. Fast states are currently bounded to be close to an assumed value used by the upper level controllers. The effect of the magnitude of this bound on performance should be analyzed to determine the tradeoff between flexibility at the lower levels and conservative constraint tightening at the upper levels.

e. While not discussed in this dissertation, many vehicle systems have hybrid dynamics, where in addition to continuous dynamics, the vehicle undergoes discrete changes such as the turning on and off of entire systems or subsystems. Instead of continuously variable actuators, many systems also have actuators that operate at discrete values. A hierarchical control framework capable of effectively controlling this class of systems would be highly valuable.

Each of these suggested topics of continued research are examples of the Practical Extensions shown in Fig. 8.1 that will enable the application of these valuable theoretical guarantees to a real physical system, such as the experimental system from Chapters 6 and 7 and future vehicle systems.

### 8.2.3 Application

The proposed graph-based modeling and hierarchical control frameworks need to be further tested on a wide variety of systems including physical systems with multiple energy domains and wider ranges of timescales. To maximize the performance of the control hierarchy, the following practical extensions should be investigated.

1. Controlling the interactions between the electrical and thermal systems is key to increasing the performance and reliability of the vehicle as a whole. A hierarchical controller that encompasses the control of both systems would allow the operation of the electrical system to be partially governed based on current and anticipated thermal constraints.
2. While the hierarchical controller presented in Chapter 7 utilized a linearized model of the nonlinear experimental system dynamics, future controller development should explore the use of nonlinear graph-based models and the optimization routines, such as genetic algorithms, used to solve the resulting nonlinear optimization problems. The trade-off between control performance and computational burden would be of particular interest.
3. The hierarchical controllers presented in this dissertation use a time-based control updating procedure where the time step between control updates is predetermined. As preliminarily demonstrated in [89], an event-based updating of each controller in the hierarchy could result in significant performance enhancements.
4. All controller computations in this dissertation are performed on powerful desktop computers. Thus, while the hierarchical controllers are designed to be computationally efficient, computation time never restricted the design of the controllers. In practice, the computational resources onboard vehicle may be very limited and each computation adds heat to the vehicle that must also be managed. Future work should investigate how computational cost and overall controller performance are related for hierarchical control.

As shown in Fig. 8.1, the ultimate extension of the proposed approach is the application of a highly functional, five-level hierarchical controller to the energy management of a real vehicle system.

# References

[1] W. D. Gerstler and R. S. Bunker, "Aircraft Engine Thermal Management: The Impact of Aviation Electric Power Demands," *ASME Global Gas Turbine News*, 2008.

[2] S. S. L. Mitchell, "Luke AFB changes refueling truck color, mitigates F-35 shutdowns," *United States Air Force*, 2014. [Online]. Available: http://www.af.mil/News/ArticleDisplay/tabid/223/Article/555558/luke-afb-changes-refueling-truck-color-mitigates-f-35-shutdowns.aspx.

[3] J. Rubinstein, "Study of the Light Utility Helicopter (LUH) acquisition program as a model for defense acquisition of non-developmental items," Ph.D. Dissertation, Naval Postgraduate School, Monterey, CA, 2014.

[4] C. Bailey, "Thermal Management Technologies for Electronic Packaging: Current Capabilities and Future Challenges for Modeling Tools," *10th Electron. Packag. Technol. Conf.*, 2008.

[5] G. Andersson, "Dynamics and Control of Electric Power Systems," *Lect. Notes Electr. Eng. Power Syst. Lab. ETH Zurich*, 2012.

[6] P. de Bock, "GE Aviation Systems Avionics: Avionics Thermal Management," *GE Glob. Res.*, 2012.

[7] M. Bodie, G. Russell, K. Mccarthy, E. Lucus, J. Zumberge, and M. Wolff, "Thermal Analysis of an Integrated Aircraft Model," *48th AIAA Aerosp. Sci. Meet.*, 2010.

[8] K. Mccarthy, M. Amrhein, P. Lamm, M. Wolff, K. Yerkes, O. Connell, B. Raczkowski, J. Wells, and W. Borger, "INVENT Modeling, Simulation, Analysis and Optimization," *48th AIAA Aerosp. Sci. Meet.*, 2010.

[9] Y. Ma, F. Borrelli, B. Hencey, B. Coffey, S. Bengea, and P. Haves, "Model Predictive Control for the Operation of Building Cooling Systems," *IEEE Trans. Control Syst. Technol.*, vol. 20, 2012.

[10] P.-D. Morosan, R. Bourdais, D. Dumur, and J. Buisson, "Building temperature regulation using a distributed model predictive control," *Energy Build.*, 2010.

[11] N. Jain, J. P. Koeln, S. Sundaram, and A. G. Alleyne, "Partially decentralized control of large-scale variable-refrigerant-flow systems in buildings," *J. Process Control*, 2014.

[12] M. Cantoni, E. Weyer, L. Yuping, S. K. Ooi, I. Mareels, and M. Ryan, "Control of Large-Scale Irrigation Networks," *Proc. IEEE*, 2007.

[13] R. R. Negenborn, A. Sahin, Z. Lukszo, B. De Schutter, and M. Morari, "A non-iterative cascaded predictive control approach for control of irrigation canals," *IEEE Int. Conf. Syst. Man Cybern.*, 2009.

[14] C. Ocampo-Martinez, D. Barcelli, V. Puig, and A. Bemporad, "Hierarchical and decentralised model predictive control of drinking water networks: application to Barcelona case study," *IET Control Theory Appl.*, 2012.

[15] D. Gayme and U. Topcu, "Optimal Power Flow With Large-Scale Storage Integration," *IEEE Trans. Power Syst.*, 2012.

[16] X. Xu, H. Jia, D. Wang, D. C. Yu, and H.-D. Chiang, "Hierarchical energy management system for multi-source multi-product microgrids," *Renew. Energy*, 2015.

[17] P. D. Christofides, R. Scattolini, D. Muñoz de la Peña, and J. Liu, "Distributed model predictive control: A tutorial review and future research directions," *Comput. Chem. Eng.*, 2013.

[18] M. J. Tippett and J. Bao, "Distributed Model Predictive Control Based on Dissipativity," *AIChE J.*, 2012.

[19] J. A. Rosero, J. A. Ortega, E. Aldabas, and L. Romeral, "Moving Towards a More Electric Aircraft," *IEEE A&E Syst. Mag.*, 2007.

[20] T. Mahefkey, K. Yerkes, B. Donovan, and M. L. Ramalingam, "Thermal Management Challenges For Future Military Aircraft Power Systems," *SAE Tech. Pap. 2004-01-3204*, 2004.

[21] M. A. Williams, J. P. Koeln, and A. G. Alleyne, "Hierarchical Control of Multi-Domain Power Flow in Mobile Systems - Part II: Aircraft Application," *ASME 2015 Dyn. Syst. Control Conf.*, 2015.

[22] B. Frank, J. Pohl, and J.-O. Palmberg, "Estimation of the potential in predictive control in a hybrid wheel loader," *11th Scand. Int. Conf. Fluid Power*, 2009.

[23] F. Wang, A. M. Zulkefli, Z. Sun, and K. A. Stelson, "Investigation on the Energy Management Strategy for Hydraulic Hybrid Wheel Loaders," *ASME 2013 Dyn. Syst. Control Conf.*, 2013.

[24] T. O. Deppen, A. G. Alleyne, K. A. Stelson, and J. J. Meyer, "Optimal Energy Use in a Light Weight Hydraulic Hybrid Passenger Vehicle," *J. Dyn. Syst. Meas. Control*, vol. 134, 2012.

[25] C. Alan, S. Ali, H. Alaa, and B. Eric, "Optimal sizing of an energy storage system for a hybrid vehicle applied to an off-road application," *IEEE/ASME Int. Conf. Adv. Intell. Mechatronics*, 2014.

[26] B. G. Liptak, *Process Control*, 3rd ed. Chilton, 1995.

[27] D. C. Karnopp, D. L. Margolis, and R. C. Rosenberg, *System Dynamics: Modeling, Simulation, and Control of Mechatronic Systems*, Fifth. Wiley, 2012.

[28] H. A. Preisig, "A graph-theory-based approach to the analysis of large-scale plants," *Comput. Chem. Eng.*, 2009.

[29] S. S. Jogwar, S. Rangarajan, and P. Daoutidis, “Reduction of complex energy-integrated process networks using graph theory,” *Comput. Chem. Eng.*, vol. 79, 2015.

[30] S. Mukherjee, S. Mishra, and J. T. Wen, “Building Temperature Control: A Passivity-Based Approach,” *Proc. IEEE Conf. Decis. Control*, 2012.

[31] K. L. Moore, T. L. Vincent, F. Lashhab, and C. Liu, “Dynamic Consensus Networks with Application to the Analysis of Building Thermal Processes,” *IFAC World Congr.*, 2011.

[32] H. Behjati, A. Davoudi, and F. Lewis, “Modular DC-DC Converters on Graphs: Cooperative Control,” *IEEE Trans. Power Electron.*, 2014.

[33] F. Blanchini, E. Franco, G. Giordano, V. Mardanlou, and P. L. Montessoro, “Compartmental flow control: Decentralization, robustness and optimality,” *Automatica*, 2016.

[34] G. Bastin and V. Guffens, “Congestion control in compartmental network systems,” *Syst. Control Lett.*, 2006.

[35] J. Jacquez and C. Simon, “Qualitative theory of Compartmental Systems,” *SIAM Rev.*, 1993.

[36] D. B. West, *Introduction to Graph Theory*. Prentice-Hall Inc., 2001.

[37] J. Maestre and R. Negenborn, *Distributed Model Predictive Control Made Easy*. Springer, 2014.

[38] R. Mohler, “Natural Bilinear Control Processes,” *IEEE Trans. Syst. Sci. Cybern.*, vol. 6, 1970.

[39] C. Bruni, G. DiPillo, and G. Koch, “Bilinear systems: An appealing class of ‘nearly linear’ systems in theory and applications,” *IEEE Trans. Automat. Contr.*, vol. 19, Aug. 1974.

[40] N. Motee and B. Sayyar-rodsari, “Optimal Partitioning in Distributed Model Predictive Control,” *Am. Control Conf.*, 2003.

[41] J. Lofberg, “YALMIP : a toolbox for modeling and optimization in MATLAB,” *2004 IEEE Int. Conf. Comput. Aided Control Syst. Des.*, 2004.

[42] “Gurobi Optimizer Reference Manual.” Gurobi Optimization Inc., 2016.

[43] D. Hill and P. Moylan, “The stability of nonlinear dissipative systems,” *IEEE Trans. Automat. Contr.*, 1976.

[44] H. Khalil, *Nonlinear Systems*, Third. Prentice-Hall Inc., 2002.

[45] R. Sepulchre, M. Jankovic, and P. Kokotovic, *Constructive Nonlinear Control*. Springer-Verlag London, 1997.

[46] A. van der Schaft, *L2-Gain and Passivity Techniques in Nonlinear Control*. Springer, 1996.

[47] R. Ortega, A. Loria, P. J. Nicklasson, and H. Sira-Ramirez, *Passivity-based Control of Euler-Lagrange Systems*. Springer, 1998.

[48] A. Ulbig, “Passivity-based Nonlinear Model Predictive Control,” University of Stuttgart, 2007.

[49] J. Bao and P. L. Lee, *Process Control: The Passive Systems Approach*. Springer, 2007.

[50] P. Falugi, "Model predictive control : a passive scheme," *IFAC World Congr.*, 2014.

[51] C. Løvaas, M. M. Seron, and G. C. Goodwin, "A dissipativity approach to robustness in constrained model predictive control," *Proc. IEEE Conf. Decis. Control*, 2007.

[52] T. Raff, C. Ebenbauer, and F. Allgöwer, "Nonlinear Model Predictive Control: A Passivity-Based Approach," in *Assessment and Future Directions of Nonlinear Model Predictive Control*, 2007.

[53] S. Sredojev and R. Eaton, "Model Predictive Controller for a Class of Nonlinear Dissipative Systems," *Am. Control Conf.*, 2014.

[54] H. Yu, F. Zhu, M. Xia, and P. J. Antsaklis, "Robust Stabilizing Output Feedback Nonlinear Model Predictive Control by Using Passivity and Dissipativity," *Proc. 2013 Eur. Control Conf.*, 2013.

[55] P. Varutti, B. Kern, and R. Findeisen, "Dissipativity-based Distributed Nonlinear Predictive Control for Cascaded Nonlinear Systems," *IFAC Symp. Adv. Control Chem. Process.*, 2012.

[56] M. Arcak and E. D. Sontag, "A passivity-based stability criterion for a class of biochemical reaction networks," *Math. Biosci. Eng.*, 2008.

[57] H. Yu and P. J. Antsaklis, "A passivity measure of systems in cascade based on passivity indices," *Proc. IEEE Conf. Decis. Control*, 2010.

[58] S. Riverso, M. Farina, and G. Ferrari-Trecate, "Plug-and-Play Decentralized Model Predictive Control for Linear Systems," *IEEE Trans. Automat. Contr.*, 2013.

[59] P. J. Moylan and D. J. Hill, "Stability criteria for large-scale systems," *IEEE Trans. Automat. Contr.*, 1978.

[60] T. Tran and J. Bao, "Supervisory Stability Assurance Layer for Hierarchical Plant-wide Process Control," *Am. Control Conf.*, 2010.

[61] A. Wächter, "An Interior Point Algorithm for Large-Scale Nonlinear Optimization with Applications in Process Engineering," Dept. Chem. Eng., Carnegie Mellon Univ., Pittsburg, PA., 2002.

[62] D. Q. Mayne, M. M. Seron, and S. V. Raković, "Robust Model Predictive Control of Constrained Linear Systems with Bounded Disturbances," *Automatica*, 2005.

[63] W. Langson, I. Chryssochoos, S. V. Raković, and D. Q. Mayne, "Robust model predictive control using tubes," *Automatica*, vol. 40, 2004.

[64] D. Limon, I. Alvarado, T. Alamo, and E. F. Camacho, "MPC for tracking piecewise constant references for constrained linear systems," *Automatica*, vol. 44, 2008.

[65] D. Limon, I. Alvarado, T. Alamo, and E. F. Camacho, "Robust tube-based MPC for tracking of constrained linear systems with additive disturbances," *J. Process Control*, vol. 20, 2010.

[66] M. Farina and R. Scattolini, "Distributed predictive control: A non-cooperative algorithm with neighbor-to-neighbor communication for linear systems," *Automatica*, vol. 48, 2012.

[67] S. Riverso and G. Ferrari-Trecate, “Tube-based distributed control of linear constrained systems,” *Automatica*, vol. 48, 2012.

[68] P. Trodden and A. Richards, “Distributed model predictive control of linear systems with persistent disturbances,” *Int. J. Control*, vol. 83, 2010.

[69] P. Trodden, “Feasible parallel-update distributed MPC for uncertain linear systems sharing convex constraints,” *Syst. Control Lett.*, vol. 74, 2014.

[70] R. Scattolini and P. Colaneri, “Hierarchical model predictive control,” *IEEE Conf. Decis. Control*, 2007.

[71] R. Scattolini, P. Colaneri, and D. De Vito, “A switched MPC approach to hierarchical control,” *17th IFAC World Congr.*, 2008.

[72] B. Picasso, D. De Vito, R. Scattolini, and P. Colaneri, “An MPC approach to the design of two-layer hierarchical control systems,” *Automatica*, vol. 46, 2010.

[73] D. Barcelli, A. Bemporad, and G. Ripaccioli, “Hierarchical Multi-Rate Control Design for Constrained Linear Systems,” *IEEE Conf. Decis. Control*, 2010.

[74] D. Barcelli, A. Bemporad, and G. Ripaccioli, “Decentralized Hierarchical Multi-Rate Control of Constrained Linear Systems,” *18th IFAC World Congr.*, 2011.

[75] C. Vermillion, A. Menezes, and I. Kolmanovsky, “Stable hierarchical model predictive control using an inner loop reference model and $\lambda$-contractive terminal constraint sets,” *Automatica*, vol. 50, 2014.

[76] B. Picasso, C. Romani, and R. Scattolini, “On the Design of Hierarchical Control Systems with MPC,” *Eur. Control Conf.*, 2009.

[77] V. Chandan and A. G. Alleyne, “Optimal partitioning for the Decentralized Thermal Control of Buildings,” *IEEE Trans. Control Syst. Technol.*, vol. 21, 2013.

[78] S. Skogestad and I. Postlethwaite, *Multivariable Feedback Control: Analysis and design*, 2nd ed. John Wiley & Sons, Inc., 2005.

[79] M. Herceg, M. Kvasnica, C. N. Jones, and M. Morari, “Multi-Parametric Toolbox 3.0,” in *European Control Conference*, 2013.

[80] B. P. Rasmussen, “Dynamic Modeling and Advanced Control of Air Conditioning and Refrigeration Systems,” Ph.D. Dissertation, Dept. Mech. Eng., Univ. Illinois Urbana-Champaign, Urbana, IL, 2005.

[81] M. Kania, J. Koeln, and A. Alleyne, “A Dynamic Modeling Toolbox for Air Vehicle Vapor Cycle Systems,” *Proc. 2012 SAE Power Syst. Conf.*, 2012.

[82] M. Williams, S. Sridharan, S. Banerjee, C. Mak, C. Pauga, P. Krein, A. Alleyne, A. Jacobi, and S. D. Urso, “PowerFlow : A Toolbox for Modeling and Simulation of Aircraft Systems,” *SAE Tech. Pap. 2015-01-2417*, 2015.

[83] A. Puntel, S. Emo, T. E. Michalak, J. Ervin, L. Byrd, V. Tsao, and T. Reitz, “Refrigerant Charge Management and Control for Next-Generation Aircraft Vapor Compression Systems,” *SAE Tech. Pap. 2013-01-2241*, 2013.

[84] T. O. Deppen, “Optimal Energy Use in Mobile Applications with Storage,” Ph.D.

Dissertation, Dept. Mech. Eng., Univ. Illinois Urbana-Champaign, Urbana, IL, 2013.

[85] T. O. Deppen, J. E. Hey, A. G. Alleyne, and T. S. Fisher, “A Model Predictive Framework for Thermal Management of Aircraft,” *ASME 2015 Dyn. Syst. Control Conf.*, 2015.

[86] S. K. Srivastava, D. A. Cartes, F. Maturana, F. Ferrese, M. Pekala, M. Zink, R. Meeker, D. Carnahan, R. Staron, D. Scheidt, and K. Huang, “A Control System Test Bed for Demonstration of Distributed Computational Intelligence Applied to Reconfiguring Heterogeneous Systems,” *IEEE Instrum. Meas. Mag.*, 2008.

[87] V. Gnielinski, “New Equations for Heat and Mass-transfer in Turbulent Pipe and Channel Flow,” *Int. Chem. Eng.*, 1976.

[88] G. A. Longo and A. Gasparella, “Refrigerant R134a vaporisation heat transfer and pressure drop inside a small brazed plate heat exchanger,” *Int. J. Refrig.*, 2007.

[89] J. P. Koeln and A. G. Alleyne, “Event-based Hierarchical Control for Power Flow in Vehicle Systems,” *Am. Control Conf.*, 2016.

# Appendix

# Robust Hierarchical Controller Code

The following Matlab code is used to implement the numerical example of the robust hierarchical controller from Section 5.8. Figs. A.1 and A.2 show the structure of the code for the generation of the controllers and the Simulink model used to execute the simulation.

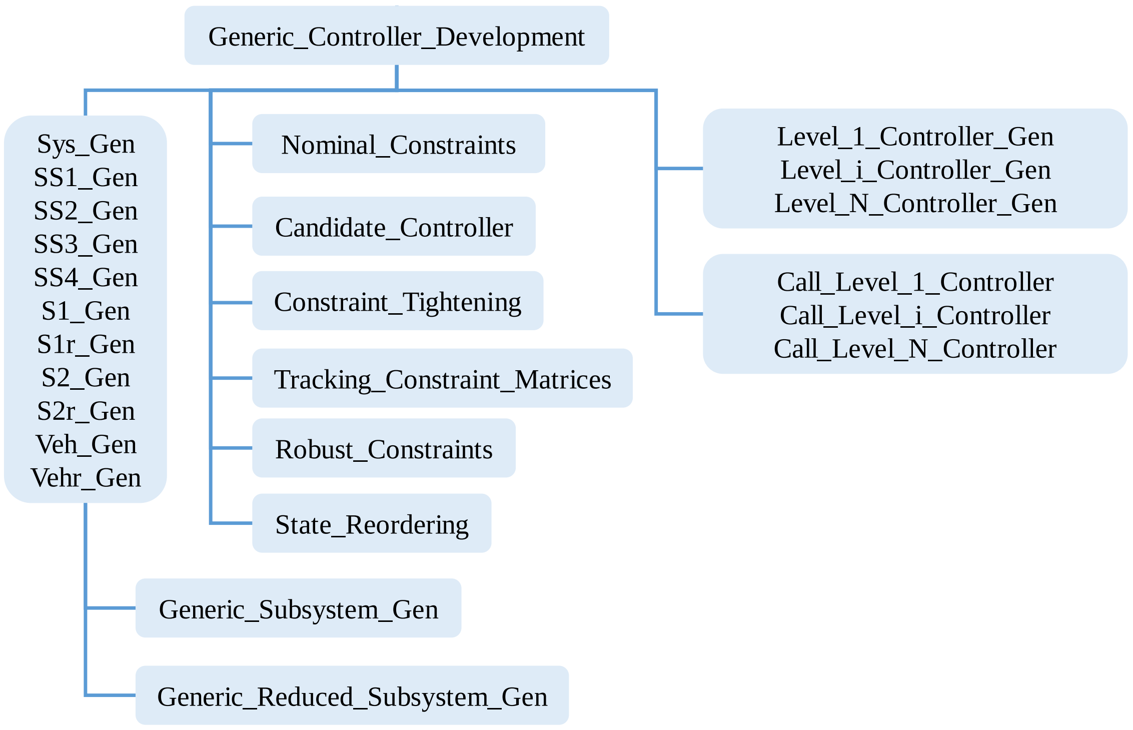


**Figure A.1 Structure of Matlab files used to generate the robust hierarchical controller.**

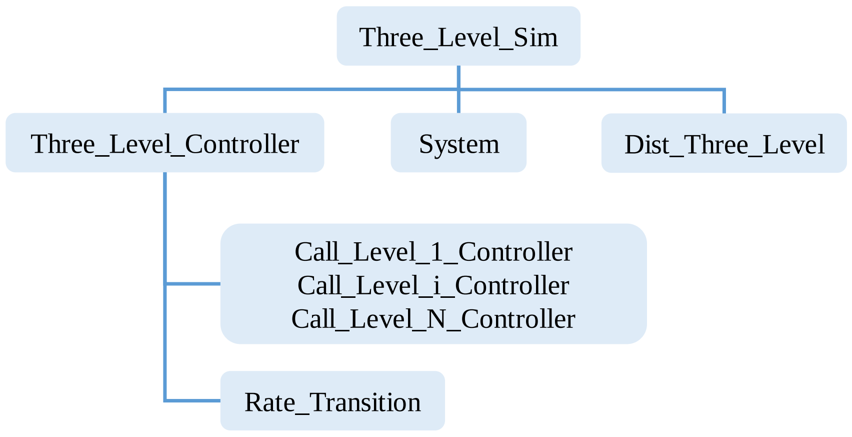


**Figure A.2 Structure of Matlab files called within the Simulink model used to simulate the robust hierarchical controller.**

## A.1 Generic_Controller_Development.m

```
%% Generate Systems
run('Sys_Gen')
run('SS1_Gen')
run('SS2_Gen')
run('SS3_Gen')
run('SS4_Gen')
run('S1_Gen')
run('S1r_Gen')
run('S2_Gen')
run('S2r_Gen')
run('Veh_Gen')
run('Vehr_Gen')
% Flags to plot and time result of controller calls
plot_ = 1;
time = 1;
% Control design parameters
Sys.Dx_LL_max_value = 0.1;      % Bound on lower level tracking
Sys.DP_max_value = 0.1;         % Bound on power flow uncertainty
Sys.DPin_max_value = 0.1;       % Bound on inlet power flow uncertainty
Sys.Dxt_max_value = 0.1;        % Bound on sink state uncertainty

%% Simulation Parameters
% Simulation time
Sys.Tsim = 1000;
% Disturbance preview flag
Sys.preview = 1;
% Size of controller output
Sys.Output_size = Sys.Ne+Sys.Ns+Sys.Nt;

%% Nominal Sets for Controllers
SS1 = Nominal_Constraints(SS1);
SS2 = Nominal_Constraints(SS2);
SS3 = Nominal_Constraints(SS3);
```

```
SS4 = Nominal_Constraints(SS4);
S1 = Nominal_Constraints(S1);
S1r = Nominal_Constraints(S1r);
S2 = Nominal_Constraints(S2);
S2r = Nominal_Constraints(S2r);
Veh = Nominal_Constraints(Veh);
Vehr = Nominal_Constraints(Vehr);

%% Constraint tightening for Subsystem Controller
% Candidate (nilpotent) feedback controller
SS1 = Candidate_Controller(SS1);
SS2 = Candidate_Controller(SS2);
SS3 = Candidate_Controller(SS3);
SS4 = Candidate_Controller(SS4);
% Upper bounds on DeltaP (assumed the same for all edges)
SS1.DP_max = Sys.DP_max_value*ones(SS1.Ne,1);
SS2.DP_max = Sys.DP_max_value*ones(SS2.Ne,1);
SS3.DP_max = Sys.DP_max_value*ones(SS3.Ne,1);
SS4.DP_max = Sys.DP_max_value*ones(SS4.Ne,1);
% Lower bounds on DeltaP (assumed negative of upper bound)
SS1.DP_min = -SS1.DP_max;
SS2.DP_min = -SS2.DP_max;
SS3.DP_min = -SS3.DP_max;
SS4.DP_min = -SS4.DP_max;
% Generate polyhedron for set DeltaP
SS1.Set_DP = Polyhedron('lb',SS1.DP_min,'ub',SS1.DP_max);
SS2.Set_DP = Polyhedron('lb',SS2.DP_min,'ub',SS2.DP_max);
SS3.Set_DP = Polyhedron('lb',SS3.DP_min,'ub',SS3.DP_max);
SS4.Set_DP = Polyhedron('lb',SS4.DP_min,'ub',SS4.DP_max);

%% Calcuate Error Sets
% Following a sequence such that each system only has power entering
% from the environment or a subsystem earlier in the sequence
% SS1
% Upper bound on inlet power uncertainty
SS1.DPin_max = Sys.DPin_max_value;
% Lower bound on inlet power uncertainty
SS1.DPin_min = -SS1.DPin_max;
% Generate polyhedron for set DeltaPin
SS1.Set_DPin = Polyhedron('lb',SS1.DPin_min,'ub',SS1.DPin_max);
% Calculate State Error Set
SS1.Set_E = plus(affineMap(SS1.Set_DP,SS1.B),...
                 affineMap(SS1.Set_DPin,SS1.V1));
% Calculate Power Error Set
SS1.Set_dP = affineMap(SS1.Set_E,SS1.K);

% SS2
% Upper bound on first inlet power uncertainty (power flow along edge 4)
SS2.Set_dPin = SS1.Set_dP.projection(4);  % Project power error for edge
SS2.dPin_max = SS2.Set_dPin.H(1,end);     % Isolate maximum value
SS2.DPin_max = [Sys.DPin_max_value;SS2.dPin_max];
% Lower bound on inlet power uncertainty
SS2.DPin_min = -SS2.DPin_max;
% Generate polyhedron for set DeltaPin
SS2.Set_DPin = Polyhedron('lb',SS2.DPin_min,'ub',SS2.DPin_max);
```

```
% Calculate State Error Set
SS2.Set_E = plus(affineMap(SS2.Set_DP,SS2.B),...
                 affineMap(SS2.Set_DPin,SS2.V1));
% Calculate Power Error Set
SS2.Set_dP = affineMap(SS2.Set_E,SS2.K);

% SS4
% Upper bound on first inlet power uncertainty (power flow along edge 9)
SS4.Set_dPin = SS2.Set_dP.projection(4);  % Project power error for edge
SS4.dPin_max = SS4.Set_dPin.H(1,end);     % Isolate maximum value
SS4.DPin_max = SS4.dPin_max;
% Lower bound on inlet power uncertainty
SS4.DPin_min = -SS4.DPin_max;
% Generate polyhedron for set DeltaPin
SS4.Set_DPin = Polyhedron('lb',SS4.DPin_min,'ub',SS4.DPin_max);
% Calculate State Error Set
SS4.Set_E = plus(affineMap(SS4.Set_DP,SS4.B),...
                 affineMap(SS4.Set_DPin,SS4.V1));
% Calculate Power Error Set
SS4.Set_dP = affineMap(SS4.Set_E,SS4.K);

% SS3
% Upper bound on first inlet power uncertainty (power flow along edge 8)
SS3.Set_dPin1 = SS1.Set_dP.projection(5); % Project power error for edge
SS3.dPin1_max = SS3.Set_dPin1.H(1,end);   % Isolate maximum value
% Upper bound on second inlet power uncertainty (power flow along edge 14)
SS3.Set_dPin2 = SS4.Set_dP.projection(4); % Project power error for edge
SS3.dPin2_max = SS3.Set_dPin2.H(1,end);   % Isolate maximum value
SS3.DPin_max = [SS3.dPin1_max;SS3.dPin2_max];
% Lower bound on inlet power uncertainty
SS3.DPin_min = -SS3.DPin_max;
% Generate polyhedron for set DeltaPin
SS3.Set_DPin = Polyhedron('lb',SS3.DPin_min,'ub',SS3.DPin_max);
% Calculate State Error Set
SS3.Set_E = plus(affineMap(SS3.Set_DP,SS3.B),...
                 affineMap(SS3.Set_DPin,SS3.V1));
% Calculate Power Error Set
SS3.Set_dP = affineMap(SS3.Set_E,SS3.K);

%% Sink State Uncertainty Sets
% SS1
% Upper bound on first sink state uncertainty (state of vertex 10)
SS1.Set_dXt1 = SS2.Set_E.projection(3);     % Project state error for vertex
SS1.Set_dXt1_max = SS1.Set_dXt1.H(1,end);   % Isolate maximum value
% Upper bound on second sink state uncertainty (state of vertex 11)
SS1.Set_dXt2 = SS3.Set_E.projection(3);     % Project state error for vertex
SS1.Set_dXt2_max = SS1.Set_dXt2.H(1,end);   % Isolate maximum value
SS1.Set_dXt_max = [SS1.Set_dXt1_max,SS1.Set_dXt2_max];
% Lower bound on sink state uncertainty
SS1.Set_dXt_min = -SS1.Set_dXt_max;
% Generate polyhedron for set DeltaXt
SS1.Set_dXt = Polyhedron('lb',SS1.Set_dXt_min,'ub',SS1.Set_dXt_max);

% SS2
% Upper bound on sink state uncertainty (state of vertex 7)
SS2.Set_dXt1 = SS4.Set_E.projection(2);      % Project state error for vertex
SS2.Set_dXt1_max = SS2.Set_dXt1.H(1,end);    % Isolate maximum value
SS2.Set_dXt_max = [SS2.Set_dXt1_max];
% Lower bound on sink state uncertainty
SS2.Set_dXt_min = -SS2.Set_dXt_max;
% Generate polyhedron for set DeltaXt
SS2.Set_dXt = Polyhedron('lb',SS2.Set_dXt_min,'ub',SS2.Set_dXt_max);

% SS3
% Upper bound on sink state uncertainty (sink state xt1)
SS3.Set_dXt_max = [Sys.Dxt_max_value];
% Lower bound on sink state uncertainty
SS3.Set_dXt_min = -SS3.Set_dXt_max;
% Generate polyhedron for set DeltaXt
SS3.Set_dXt = Polyhedron('lb',SS3.Set_dXt_min,'ub',SS3.Set_dXt_max);

% SS4
% Upper bound on first sink state uncertainty (state of vertex 11)
SS4.Set_dXt1 = SS3.Set_E.projection(3);      % Project state error for vertex
SS4.Set_dXt1_max = SS4.Set_dXt1.H(1,end);    % Isolate maximum value
% Upper bound on second sink state uncertainty (sink state xt2)
SS4.Set_dXt2_max = Sys.Dxt_max_value;
SS4.Set_dXt_max = [SS4.Set_dXt1_max,SS4.Set_dXt2_max];
% Lower bound on sink state uncertainty
SS4.Set_dXt_min = -SS4.Set_dXt_max;
% Generate polyhedron for set DeltaXt
SS4.Set_dXt = Polyhedron('lb',SS4.Set_dXt_min,'ub',SS4.Set_dXt_max);

%%  Generate Robust State and Input Constraint Sets
SS1 = Constraint_Tightening(SS1);
SS2 = Constraint_Tightening(SS2);
SS3 = Constraint_Tightening(SS3);
SS4 = Constraint_Tightening(SS4);

%% Subsystem tracking constraint set
SS1 = Tracking_Constraint_Matrices(SS1,SS1,S1r);
SS2 = Tracking_Constraint_Matrices(SS2,SS2,S1r);
SS3 = Tracking_Constraint_Matrices(SS3,SS3,S2r);
SS4 = Tracking_Constraint_Matrices(SS4,SS4,S2r);
S1r = Tracking_Constraint_Matrices(S1r,S1,Vehr);
S2r = Tracking_Constraint_Matrices(S2r,S2,Vehr);

%% Constraint Tightening for Upper Levels
% Robust constraints are initial formed from the subsystem robust constraints
% S1r
% Robust state constraints
S1r.x_max_robust = [SS2.Set_X_robust.H(1,end);SS1.Set_X_robust.H(1,end);...
                    SS2.Set_X_robust.H(2,end);SS2.Set_X_robust.H(3,end)];
S1r.x_min_robust = -S1r.x_max_robust;
% Robust input constraints
S1r.U_max_robust =
[SS1.Set_U_robust.H(1:4,end);SS2.Set_U_robust.H(1:3,end);...
                    SS1.Set_U_robust.H(5,end);SS2.Set_U_robust.H(4,end)];
```

```
S1r.U_min_robust = -S1r.U_max_robust;

% S2r
% Robust state constraints
S2r.x_max_robust = [SS3.Set_X_robust.H(1,end);SS3.Set_X_robust.H(2,end);...
                    SS4.Set_X_robust.H(1,end);SS4.Set_X_robust.H(2,end);...
                    SS3.Set_X_robust.H(3,end)];
S2r.x_min_robust = -S2r.x_max_robust;
% Robust input constraints
S2r.U_max_robust = [SS3.Set_U_robust.H(1:4,end);SS4.Set_U_robust.H(4,end);...
                    SS4.Set_U_robust.H(1:3,end);SS4.Set_U_robust.H(5,end)];
S2r.U_min_robust = -S2r.U_max_robust;

% Vehr
% Robust state constraints
Vehr.x_max_robust = [SS2.Set_X_robust.H(1,end);SS3.Set_X_robust.H(1,end);...
                     SS4.Set_X_robust.H(2,end);SS3.Set_X_robust.H(3,end)];
Vehr.x_min_robust = -Vehr.x_max_robust;
% Robust input constraints
Vehr.U_max_robust = [S1r.U_max_robust;S2r.U_max_robust];
Vehr.U_min_robust = -Vehr.U_max_robust;
% Generate polyhedron for robust sets
S1r = Robust_Constraints(S1r,Sys.Dx_LL_max_value);
S2r = Robust_Constraints(S2r,Sys.Dx_LL_max_value);
Vehr = Robust_Constraints(Vehr,Sys.Dx_LL_max_value);

%% Z matrices for state reordering
S1r = State_Reordering(S1r,S1);
S2r = State_Reordering(S2r,S2);
Vehr = State_Reordering(Vehr,Veh);

%% Weights
% Nominal weightings (1,1,0), pure economic (0,0,1)
weightings.x        = 0;
weightings.u        = 0;
weightings.u_eff    = 1;

%% Constraint Flags
constraints.robustOn             = 1;
constraints.stateTrackingOn      = 1;
constraints.pOutTrackingOn       = 1;
constraints.lowerStateBoundsOn   = 1;
constraints.Dx_LL_max_value      = Sys.Dx_LL_max_value;

%% Level N Controller Generation
% SS1
SS1.horizon = 5;    % Prediction horizon
SS1 = Level_N_Controller_Gen( SS1, weightings, constraints );
% SS2
SS2.horizon = 5;    % Prediction horizon
SS2 = Level_N_Controller_Gen( SS2, weightings, constraints );
% SS3
SS3.horizon = 5;    % Prediction horizon
SS3 = Level_N_Controller_Gen( SS3, weightings, constraints );
```

```
% SS4
SS4.horizon = 5;    % Prediction horizon
SS4 = Level_N_Controller_Gen( SS4, weightings, constraints );

%% Level i Controller Generation
% S1r
S1r.horizon = 5;    % Prediction horizon
S1r = Level_i_Controller_Gen( S1r, weightings, constraints );
% S2r
S2r.horizon = 5;    % Prediction horizon
S2r = Level_i_Controller_Gen( S2r, weightings, constraints );

%% Level 1 Controller Generation
% Vehr
Vehr.horizon = 5;    % Prediction horizon
Vehr = Level_1_Controller_Gen( Vehr, weightings, constraints );

%% Test Vehr Controller
Vehr.x0 = Vehr.x0;
Vehr.Pin = repmat(Veh.Pin0,1,Vehr.horizon);
Vehr.xt = repmat(Veh.xt0,1,Vehr.horizon+1);
Vehr.xlow0 = Veh.x0(Vehr.xf);

[ Vehr ] = Call_Level_1_Controller( Vehr, plot_, time );

%% Test S1r Controller
S1r.x0 = S1r.x0;
S1r.Pin = repmat(S1.Pin0,1,S1r.horizon);
S1r.xt = repmat(S1.xt0,1,S1r.horizon+1);
S1r.xdotDes = zeros(max(size(S1r.Zup,1),1),S1r.horizon);
S1r.PoutDes = zeros(max(size(S1r.Zout,1),1),S1r.horizon);
S1r.xlowDes = S1r.x0(S1r.xlow);
S1r.xlow0 = S1.x0(S1r.xf);

[ S1r ] = Call_Level_i_Controller( S1r, plot_, time );

%% Test S2r Controller
S2r.x0 = S2r.x0;
S2r.Pin = repmat(S2.Pin0,1,S2r.horizon);
S2r.xt = repmat(S2.xt0,1,S2r.horizon+1);
S2r.xdotDes = zeros(max(size(S2r.Zup,1),1),S2r.horizon);
S2r.PoutDes = zeros(max(size(S2r.Zout,1),1),S2r.horizon);
S2r.xlowDes = S2r.x0(S2r.xlow);
S2r.xlow0 = S2.x0(S2r.xf);

[ S2r ] = Call_Level_i_Controller( S2r, plot_, time );

%% Test SS1 Controller
SS1.x0 = SS1.x0;
SS1.Pin = repmat(SS1.Pin0,1,SS1.horizon);
SS1.xt = repmat(SS1.xt0,1,SS1.horizon);
SS1.xdotDes = zeros(max(size(SS1.Zup,1),1),SS1.horizon);
SS1.PoutDes = zeros(max(size(SS1.Zout,1),1),SS1.horizon);
```

```
SS1.xlowDes = SS1.x0(SS1.xlow);

[ SS1 ] = Call_Level_N_Controller( SS1, plot_, time );

%% Test SS2 Controller
SS2.x0 = SS2.x0;
SS2.Pin = repmat(SS2.Pin0,1,SS2.horizon);
SS2.xt = repmat(SS2.xt0,1,SS2.horizon);
SS2.xdotDes = zeros(max(size(SS2.Zup,1),1),SS2.horizon);
SS2.PoutDes = zeros(max(size(SS2.Zout,1),1),SS2.horizon);
SS2.xlowDes = 0;

[ SS2 ] = Call_Level_N_Controller( SS2, plot_, time );

%% Test SS3 Controller
SS3.x0 = SS3.x0;
SS3.Pin = repmat(SS3.Pin0,1,SS3.horizon);
SS3.xt = repmat(SS3.xt0,1,SS3.horizon);
SS3.xdotDes = zeros(max(size(SS3.Zup,1),1),SS3.horizon);
SS3.PoutDes = zeros(max(size(SS3.Zout,1),1),SS3.horizon);
SS3.xlowDes = 0;

[ SS3 ] = Call_Level_N_Controller( SS3, plot_, time );

%% Test SS4 Controller
SS4.x0 = SS4.x0;
SS4.Pin = repmat(SS4.Pin0,1,SS4.horizon);
SS4.xt = repmat(SS4.xt0,1,SS4.horizon);
SS4.xdotDes = zeros(max(size(SS4.Zup,1),1),SS4.horizon);
SS4.PoutDes = zeros(max(size(SS4.Zout,1),1),SS4.horizon);
SS4.xlowDes = SS4.x0(SS4.xlow);

[ SS4 ] = Call_Level_N_Controller( SS4, plot_, time );
```

### A.2 Sys_Gen.m

```
%% System Parameters
% Name of system
Sys.Name = 'Sys';
% Number of vertices
Sys.Nv = 12;          % Number of vertices
Sys.Ne = 18;          % Number of edges
Sys.Ns = 2;           % Number of sources
Sys.Nt = 2;           % Numper of sinks
Sys.Nvs = 2;          % Number of slow vertices (First states in vector)
Sys.Nvm = 5;          % Number of medium vertices (middle states in vector)
Sys.Nvf = 5;          % Number of fast vertices (Last states in vector)
% Head states of inlet power flows
Sys.xin = [3;4];
% Discrete update rate
Sys.DT = 1;

%% Edge matrix
% (row i corresponds to edge i, first column is tail vertex,
%  second colum is head vertex)
% (sink vertices numbers are [ Sys.Nv+1 : Sys.Nv+Sys.Nt ])
Sys.e = [3 8;
         3 9;
         9 8;
         9 10;
         4 10;
         4 1;
         1 10;
         8 11;
         10 7;
         11 2;
         11 5;
         2 5;
         5 13;
         6 11;
         7 6;
         7 12;
         12 6;
         12 14];

%% Capacitance Vector
Sys.Caps = [1000*ones(Sys.Nvs,1);100*ones(Sys.Nvm,1);10*ones(Sys.Nvf,1)];

%% Edge Parameters
Sys.a = ones(Sys.Ne,1);    % Tail coefficient
Sys.b = ones(Sys.Ne,1);    % Head coefficient
Sys.c = ones(Sys.Ne,1);    % Input coefficient

%% Initial Conditions
Sys.x0 = zeros(Sys.Nv,1);
Sys.u0 = zeros(Sys.Ne,1);
Sys.xt0 = zeros(Sys.Nt,1);
Sys.Pin0 = zeros(Sys.Ns,1);

%% Constraints
Sys.x_max = ones(Sys.Nv,1);
Sys.x_min = -Sys.x_max;
Sys.u_max = ones(Sys.Ne,1);
Sys.u_min = -Sys.u_max;

% End of user specified information
%% Incidence Matrix
Sys.M = zeros(Sys.Nv+Sys.Nt,Sys.Ne);
for i = 1:Sys.Ne;
    Sys.M(Sys.e(i,1),i) = 1;
    Sys.M(Sys.e(i,2),i) = -1;
end
clear i

Sys.M_upper = Sys.M(1:Sys.Nv,:);         % System dynamics
Sys.M_lower = Sys.M(Sys.Nv+1:end,:);     % Sink states
Sys.M_s = Sys.M_upper(1:Sys.Nvs,:);                     % Slow states
Sys.M_m = Sys.M_upper(1+Sys.Nvs:Sys.Nvs+Sys.Nvm,:);     % Medium states
Sys.M_f = Sys.M_upper(1+Sys.Nvs+Sys.Nvm:end,:);         % Fast states

%% Weighted Incidence Matrix
Sys.Mab = zeros(Sys.Nv+Sys.Nt,Sys.Ne);
for i = 1:Sys.Ne;
    Sys.Mab(Sys.e(i,1),i) = Sys.a(i);
    Sys.Mab(Sys.e(i,2),i) = -Sys.b(i);
end
clear i

Sys.Mab_upper = Sys.Mab(1:Sys.Nv,:);         % System dynamics
Sys.Mab_lower = Sys.Mab(Sys.Nv+1:end,:);     % Sink states
Sys.Mab_s = Sys.Mab_upper(1:Sys.Nvs,:);                    % Slow states
Sys.Mab_m = Sys.Mab_upper(1+Sys.Nvs:Sys.Nvs+Sys.Nvm,:);    % Medium states
Sys.Mab_f = Sys.Mab_upper(1+Sys.Nvs+Sys.Nvm:end,:);        % Fast states

%% Input Vector
Sys.D = zeros(Sys.Nv,Sys.Ns);
for i = 1:Sys.Ns
    Sys.D(Sys.xin(i),i) = 1;
end
clear i
Sys.D_s = Sys.D(1:Sys.Nvs,:);
Sys.D_m = Sys.D(1+Sys.Nvs:Sys.Nvs+Sys.Nvm,:);
Sys.D_f = Sys.D(1+Sys.Nvs+Sys.Nvm:end,:);

%% System Dynamics
% Continuous
Sys.A_c = diag(1./Sys.Caps)*(-Sys.M_upper*Sys.Mab_upper');
Sys.B_c = diag(1./Sys.Caps)*(-Sys.M_upper);
Sys.beta = diag(Sys.c);
Sys.V_c1 = diag(1./Sys.Caps)*(Sys.D);
Sys.V_c2 = diag(1./Sys.Caps)*(-Sys.M_upper*Sys.Mab_lower');
Sys.V_c3 = diag(1./Sys.Caps)*(-Sys.M_upper);
% Discrete
Sys.A = eye(Sys.Nv)+Sys.DT*Sys.A_c;
Sys.B = Sys.DT*Sys.B_c;
Sys.V1 = Sys.DT*Sys.V_c1;
Sys.V2 = Sys.DT*Sys.V_c2;
Sys.V3 = Sys.DT*Sys.V_c3;
% Initial power flow
Sys.P0 = Sys.Mab_upper'*Sys.x0+Sys.Mab_lower'*Sys.xt0+diag(Sys.c)*Sys.u0;
```

### A.3 SS1_Gen.m

```
%% SS1 Parameters
% Name of system
SS1.Name = 'SS1';
% Number of vertices
SS1.Nv = 3;             % Number of vertices
SS1.Ne = 5;             % Number of edges
SS1.Ns = 1;             % Number of sources
SS1.Nt = 2;             % Numper of sinks
SS1.Nvs = 0;            % Number of slow vertices (First states in vector)
```

```
SS1.Nvm = 1;            % Number of medium vertices (middle states in vector)
SS1.Nvf = 2;            % Number of fast vertices (Last states in vector)
% Head states of inlet power flows
SS1.xin = [1];
% Discrete update rate
SS1.DT = 1;
% Relation to full system
SS1.Sys_Vs = [3;8;9];       % Indices of vertices in full system
SS1.Sys_Es = [1;2;3;4;8];   % Indices of edges in full system
SS1.xt0 = Sys.x0([10 11]);  % Initial sink states
SS1.Pin0 = Sys.Pin0(1);     % Initial inlet power flows

%% Edge matrix
SS1.e = [1 2;
         1 3;
         3 2;
         3 4;
         2 5];

%% Generate remaining subsystem values
[SS1] = Generic_Subsystem_Gen(SS1,Sys);
```

### A.4 SS2_Gen.m

```
%% SS2 Parameters
% Name of system
SS2.Name = 'SS2';
% Number of vertices
SS2.Nv = 3;             % Number of vertices
SS2.Ne = 4;             % Number of edges
SS2.Ns = 2;             % Number of sources
SS2.Nt = 1;             % Numper of sinks
SS2.Nvs = 1;            % Number of slow vertices (First states in vector)
SS2.Nvm = 1;            % Number of medium vertices (middle states in vector)
SS2.Nvf = 1;            % Number of fast vertices (Last states in vector)
% Head states of inlet power flows
SS2.xin = [2;3];
% Discrete update rate
SS2.DT = 1;
% Relation to full system
SS2.Sys_Vs = [1;4;10];       % Indices of states in full system
SS2.Sys_Es = [5;6;7;9];   % Indices of edges in full system
SS2.xt0 = Sys.x0(7);  % Initial sink states
SS2.Pin0 = [Sys.Pin0(2);Sys.P0(4)];     % Initial inlet power flows

%% Edge matrix
SS2.e = [2 3;
         2 1;
         1 3;
         3 4];

%% Generate remaining subsystem values
[SS2] = Generic_Subsystem_Gen(SS2,Sys);
```

### A.5 SS3_Gen.m

```
%% SS3 Parameters
% Name of system
SS3.Name = 'SS3';
% Number of vertices
SS3.Nv = 3;                % Number of vertices
SS3.Ne = 4;                % Number of edges
SS3.Ns = 2;                % Number of sources
SS3.Nt = 1;                % Numper of sinks
SS3.Nvs = 1;               % Number of slow vertices (First states in vector)
SS3.Nvm = 1;               % Number of medium vertices (middle states in vector)
SS3.Nvf = 1;               % Number of fast vertices (Last states in vector)
% Head states of inlet power flows
SS3.xin = [3;3];
% Discrete update rate
SS3.DT = 1;
% Relation to full system
SS3.Sys_Vs = [2;5;11];       % Indices of states in full system
SS3.Sys_Es = [10;11;12;13];   % Indices of edges in full system
SS3.xt0 = Sys.xt0(1);  % Initial sink states
SS3.Pin0 = [Sys.P0(8);Sys.P0(14)];     % Initial inlet power flows

%% Edge matrix
SS3.e = [3 1;
         3 2;
         1 2;
         2 4];

%% Generate remaining subsystem values
[SS3] = Generic_Subsystem_Gen(SS3,Sys);
```

### A.6 SS4_Gen.m

```
%% SS4 Parameters
% Name of system
SS4.Name = 'SS4';
% Number of vertices
SS4.Nv = 3;                % Number of vertices
SS4.Ne = 5;                % Number of edges
SS4.Ns = 1;                % Number of sources
SS4.Nt = 2;                % Numper of sinks
SS4.Nvs = 0;               % Number of slow vertices (First states in vector)
SS4.Nvm = 2;               % Number of medium vertices (middle states in vector)
SS4.Nvf = 1;               % Number of fast vertices (Last states in vector)
% Head states of inlet power flows
SS4.xin = [2];
% Discrete update rate
SS4.DT = 1;
% Relation to full system
SS4.Sys_Vs = [6;7;12];       % Indices of states in full system
SS4.Sys_Es = [15;16;17;14;18];   % Indices of edges in full system
SS4.xt0 = [Sys.x0(11);Sys.xt0(2)];  % Initial sink states
SS4.Pin0 = Sys.P0(9);     % Initial inlet power flows
```

```
%% Edge matrix
SS4.e = [2 1;
         2 3;
         3 1;
         1 4;
         3 5];

%% Generate remaining subsystem values
[SS4] = Generic_Subsystem_Gen(SS4,Sys);
```

### A.7 S1_Gen.m

```
%% S1 Parameters
% Name of system
S1.Name = 'S1';
% Number of vertices
S1.Nv = 6;                % Number of vertices
S1.Ne = 9;                % Number of edges
S1.Ns = 2;                % Number of sources
S1.Nt = 2;                % Numper of sinks
S1.Nvs = 1;               % Number of slow vertices (First states in vector)
S1.Nvm = 2;               % Number of medium vertices (middle states in vector)
S1.Nvf = 3;               % Number of fast vertices (Last states in vector)
% Head states of inlet power flows
S1.xin = [2;3];
% Discrete update rate
S1.DT = 10;
% Relation to full system
S1.Sys_Vs = [1;3;4;8;9;10];        % Indices of states in full system
S1.Sys_Es = [1:9]';    % Indices of edges in full system
S1.xt0 = Sys.x0([11 7]);  % Initial sink states
S1.Pin0 = Sys.Pin0;       % Initial inlet power flows

%% Edge matrix
S1.e = [2 4;
        2 5;
        5 4;
        5 6;
        3 6;
        3 1;
        1 6;
        4 7;
        6 8];

%% Generate remaining subsystem values
[S1] = Generic_Subsystem_Gen(S1,Sys);
```

### A.8 S1r_Gen.m

```
%% S1r Parameters
% Name of system
S1r.Name = 'S1r';
% Relation to nominal system
S1r.Sys_Es = 1:S1.Ne;     % Keep all edges
S1r.Sys_Vs = [1 3 4 10]; % Keep slow and medium states and head of subsystem
```

```
%% Generate reduced subsystem
[S1r] = Generic_Reduced_Subsystem_Gen(S1r,S1);
```

### A.9 S2_Gen.m

```
%% S2 Parameters
% Name of system
S2.Name = 'S2';
% Number of vertices
S2.Nv = 6;               % Number of vertices
S2.Ne = 9;               % Number of edges
S2.Ns = 2;               % Number of sources
S2.Nt = 2;               % Numper of sinks
S2.Nvs = 1;              % Number of slow vertices (First states in vector)
S2.Nvm = 3;              % Number of medium vertices (middle states in vector)
S2.Nvf = 2;              % Number of fast vertices (Last states in vector)
% Head states of inlet power flows
S2.xin = [5;4];
% Discrete update rate
S2.DT = 10;
% Relation to full system
S2.Sys_Vs = [2;5;6;7;11;12];        % Indices of states in full system
S2.Sys_Es = [10:18]';    % Indices of edges in full system
S2.xt0 = Sys.xt0;   % Initial sink states
S2.Pin0 = [Sys.P0(8);Sys.P0(9)];      % Initial inlet power flows

%% Edge matrix
S2.e = [5 1;
        5 2;
        1 2;
        2 7;
        3 5;
        4 3;
        4 6;
        6 3;
        6 8];

%% Generate remaining subsystem values
[S2] = Generic_Subsystem_Gen(S2,Sys);
```

### A.10 S2r_Gen.m

```
%% S2r Parameters
% Name of system
S2r.Name = 'S2r';
% Relation to nominal system
S2r.Sys_Es = 1:S2.Ne;      % Keep all edges
S2r.Sys_Vs = [2 5 6 7 11];% Keep slow and medium states and head of subsystem

%% Generate reduced subsystem
[S2r] = Generic_Reduced_Subsystem_Gen(S2r,S2);
```

### A.11 Veh_Gen.m

```
%% Veh Parameters
% Name of system
Veh.Name = 'Veh';
% Number of vertices
Veh.Nv = 12;             % Number of vertices
Veh.Ne = 18;             % Number of edges
Veh.Ns = 2;              % Number of sources
Veh.Nt = 2;              % Numper of sinks
Veh.Nvs = 2;             % Number of slow vertices (First states in vector)
Veh.Nvm = 5;             % Number of medium vertices (middle states in vector)
Veh.Nvf = 5;             % Number of fast vertices (Last states in vector)
% Head states of inlet power flows
Veh.xin = [3;4];
% Discrete update rate
Veh.DT = 100;
% Relation to full system
Veh.Sys_Vs = [1:12]';        % Indices of states in full system
Veh.Sys_Es = [1:18]';    % Indices of edges in full system
Veh.xt0 = Sys.xt0;  % Initial sink states
Veh.Pin0 = Sys.Pin0;      % Initial inlet power flows

%% Edge matrix
Veh.e = [3 8;
         3 9;
         9 8;
         9 10;
         4 10;
         4 1;
         1 10;
         8 11;
         10 7;
         11 2;
         11 5;
         2 5;
         5 13;
         6 11;
         7 6;
         7 12;
         12 6;
         12 14];

%% Generate remaining subsystem values
[Veh] = Generic_Subsystem_Gen(Veh,Sys);
```

### A.12 Vehr_Gen.m

```
%% Vehr Parameters
% Name of system
Vehr.Name = 'Vehr';
% Relation to nominal system
Vehr.Sys_Es = 1:Veh.Ne;     % Keep all edges
Vehr.Sys_Vs = [1 2 7 11];  % Keep slow states and head of system

%% Generate reduced subsystem
[Vehr] = Generic_Reduced_Subsystem_Gen(Vehr,Veh);
```

## A.13 Generic_Subsystem_Gen.m

```
function [Output] = Generic_Subsystem_Gen(Input,Sys)

Output = Input;

%% Capacitance Vector
Output.Caps = Sys.Caps(Output.Sys_Vs);

%% Edge Parameters
Output.a = Sys.a(Output.Sys_Es);    % Tail coefficient
Output.b = Sys.b(Output.Sys_Es);    % Head coefficient
Output.c = Sys.c(Output.Sys_Es);    % Input coefficient

%% Incidence Matrix
Output.M = zeros(Output.Nv+Output.Nt,Output.Ne);
for i = 1:Output.Ne;
    Output.M(Output.e(i,1),i) = 1;
    Output.M(Output.e(i,2),i) = -1;
end
clear i

Output.M_upper = Output.M(1:Output.Nv,:);         % Outputtem dynamics
Output.M_lower = Output.M(Output.Nv+1:end,:);     % Sink states
% Incidence matrix for slow, medium, and fast states
Output.M_s = Output.M_upper(1:Output.Nvs,:);
Output.M_m = Output.M_upper(1+Output.Nvs:Output.Nvs+Output.Nvm,:);
Output.M_f = Output.M_upper(1+Output.Nvs+Output.Nvm:end,:);

%% Weighted Incidence Matrix
Output.Mab = zeros(Output.Nv+Output.Nt,Output.Ne);
for i = 1:Output.Ne;
    Output.Mab(Output.e(i,1),i) = Output.a(i);
    Output.Mab(Output.e(i,2),i) = -Output.b(i);
end
clear i

Output.Mab_upper = Output.Mab(1:Output.Nv,:);         % Outputtem dynamics
Output.Mab_lower = Output.Mab(Output.Nv+1:end,:);     % Sink states
% Weighted incidence matrix for slow, medium, and fast states
Output.Mab_s = Output.Mab_upper(1:Output.Nvs,:);
Output.Mab_m = Output.Mab_upper(1+Output.Nvs:Output.Nvs+Output.Nvm,:);
Output.Mab_f = Output.Mab_upper(1+Output.Nvs+Output.Nvm:end,:);

%% Input Vector
Output.D = zeros(Output.Nv,Output.Ns);
for i = 1:Output.Ns
    Output.D(Output.xin(i),i) = 1;
end
clear i
Output.D_s = Output.D(1:Output.Nvs,:);
Output.D_m = Output.D(1+Output.Nvs:Output.Nvs+Output.Nvm,:);
Output.D_f = Output.D(1+Output.Nvs+Output.Nvm:end,:);
```

```
%% Outputtem Dynamics
% Continuous
Output.A_c = diag(1./Output.Caps)*(-Output.M_upper*Output.Mab_upper');
Output.B_c = diag(1./Output.Caps)*(-Output.M_upper);
Output.beta = diag(Output.c);
Output.V_c1 = diag(1./Output.Caps)*(Output.D);
Output.V_c2 = diag(1./Output.Caps)*(-Output.M_upper*Output.Mab_lower');
% Discrete
Output.A = eye(Output.Nv)+Output.DT*Output.A_c;
Output.B = Output.DT*Output.B_c;
Output.V1 = Output.DT*Output.V_c1;
Output.V2 = Output.DT*Output.V_c2;
Output.A0 = eye(Output.Nv);

%% Model Reduction Matrices
Output.T = eye(Output.Ne);
Output.Y = zeros(Output.Ne,Output.Ns);

%% Initial Conditions
Output.x0 = Sys.x0(Output.Sys_Vs);
Output.u0 = Sys.u0(Output.Sys_Es);
Output.P0 = Sys.P0(Output.Sys_Es);

%% Constraints
Output.x_max = Sys.x_max(Output.Sys_Vs);
Output.x_min = -Output.x_max;
Output.u_max = Sys.u_max(Output.Sys_Es);
Output.u_min = -Output.u_max;

end
```

### A.14 Generic_Reduced_Subsystem_Gen.m

```
function [Output] = Generic_Reduced_Subsystem_Gen(Input,Full)

Output = Input;

%% Number of vertices
Output.Nv = length(Output.Sys_Vs);              % Number of reduced vertices
Output.Ne = Full.Ne-(Full.Nv-Output.Nv);        % Number of reduced edges
Output.Ns = Full.Ns;                            % Number of sources
Output.Nt = Full.Nt;                            % Numper of sinks
Output.Nvs = Full.Nv-Full.Nvf-Full.Nvm;         % Number os slow states
% Discrete update rate
Output.DT = Full.DT;
% Determine indices for vertices of reduced matrix
[trash,Output.Full_Vs] = ismember(Output.Sys_Vs,Full.Sys_Vs);
% Matrices for reduced power flow equation
Output.xf = setdiff(1:Full.Nv,Output.Full_Vs);     % Fast vertices
[Output.R,Output.jb] = rref([Full.M(Output.xf,:) Full.D(Output.xf,:)]);
Output.Sys_Es = setdiff(1:Full.Ne,Output.jb)';
Output.F = -Output.R(:,Output.Sys_Es);
Output.T = zeros(Full.Ne,Output.Ne);
```

```
for i = 1:Output.Ne
    Output.T(Output.Sys_Es(i),i) = 1;
end
for i = 1:size(Output.F,1)
    Output.T(Output.jb(i),:) = Output.F(i,:);
end
Output.G = Output.R(:,end-Output.Ns+1:end);
Output.Y = zeros(Full.Ne,Full.Ns);
for i = 1:size(Output.F,1)
    Output.Y(Output.jb(i),:) = Output.G(i,:);
end
% Reduced power flow matrices
Output.Br_hat = Full.B(Output.Full_Vs,:)*Output.T;
Output.Vr_hat = Full.V1(Output.Full_Vs,:)+Full.B(Output.Full_Vs,:)*Output.Y;

% Upper incidence matrix for reduced system
Output.M_upper = Full.M(Output.Full_Vs,:);
% Input Matrix for reduced system
Output.D = Full.D(Output.Full_Vs,:);
% Weighted incidence matrix for full system
Output.Mab = Full.Mab;
% Weighted incidence matrix corresponding to fast states
Output.Mfast = Full.Mab(Output.xf,:);
% Vertex capacitances
Output.Caps = Full.Caps(Output.Full_Vs);

% State Constraints
Output.x_max = Full.x_max(Output.Full_Vs);
Output.x_min = Full.x_min(Output.Full_Vs);

% Input Constraints
Output.u_max = Full.u_max;
Output.u_min = Full.u_min;

% Initial Conditions
Output.x0 = Full.x0(Output.Full_Vs);
Output.u0 = Full.u0;

end
```

## A.15 Nominal_Constraints.m

```
function Output = Nominal_Constraints(Input)

Output = Input;

% State Constraints
Output.Set_X = Polyhedron('lb',Output.x_min,'ub',Output.x_max);
% Input Constraints
Output.Set_U = Polyhedron('lb',Output.u_min,'ub',Output.u_max);

end
```

### A.16 Candidate_Controller.m

```
function [Output] = Candidate_Controller(Input)

Output = Input;

% Candidate control law
Output.K = -pinv(Output.B);

end
```

### A.17 Constraint_Tightening.m

```
function Output = Constraint_Tightening(Input)

Output = Input;

% Robust state set
Output.Set_X_robust = minus(Output.Set_X,Output.Set_E);
% Robust input set
Output.Set_U_robust = minus(Output.Set_U,...
                affineMap(Output.Set_dP,inv(Output.beta)));
Output.Set_U_robust = minus(Output.Set_U_robust,...
                affineMap(Output.Set_E,-inv(Output.beta)*Output.Mab_upper'));
Output.Set_U_robust = minus(Output.Set_U_robust,...
                affineMap(Output.Set_dXt,-inv(Output.beta)*Output.Mab_lower'));
Output.Set_X_robust.minHRep;    % Minimum representation of constraints
Output.Set_U_robust.minHRep;    % Minimum representation of constraints

% Input Set constraint reordering
indexes = [];
for i = 1:size(Output.Set_X_robust.H,2)-1
    indexes = [indexes;find(Output.Set_X_robust.H(:,i) == 1)];
end
for i = 1:size(Output.Set_X_robust.H,2)-1
    indexes = [indexes;find(Output.Set_X_robust.H(:,i) == -1)];
end
Output.Set_X_robust=Polyhedron('A',Output.Set_X_robust.H(indexes,1:end-1),...
                                'b',Output.Set_X_robust.H(indexes,end));

% Input Set constraint reordering
indexes = [];
for i = 1:size(Output.Set_U_robust.H,2)-1
    indexes = [indexes;find(Output.Set_U_robust.H(:,i) == 1)];
end
for i = 1:size(Output.Set_U_robust.H,2)-1
    indexes = [indexes;find(Output.Set_U_robust.H(:,i) == -1)];
end
Output.Set_U_robust=Polyhedron('A',Output.Set_U_robust.H(indexes,1:end-1),...
                                'b',Output.Set_U_robust.H(indexes,end));

end
```

### A.18 Tracking_Constraint_Matrices.m

```
function Output = Tracking_Constraint_Matrices(Input,Full,UpperReduced)

Output = Input;

% Find vertices contained in reduced upper level system
[trash,Output.Track] = ismember(intersect(Output.Sys_Vs,...
                                UpperReduced.Sys_Vs),Output.Sys_Vs);
% Generate state tracking matrix
Output.Zup = zeros(length(Output.Track),Output.Nv);
for i = 1:length(Output.Track)
    Output.Zup(i,Output.Track(i)) = 1;
end
% Generate outlet power tracking matrix
Output.Zout = zeros(Output.Nt,size(Output.T,1));
j = 1;
for i = 1:Full.Ne
    if Full.e(i,2) > Full.Nv
        Output.Zout(j,i) = 1;
        j = j + 1;
    end
end
% Generate fast state constraint matrix
Output.xlow = setdiff(1:Output.Nv,Output.Track);
Output.Zlow = zeros(length(Output.xlow),Output.Nv);
for i = 1:length(Output.xlow)
    Output.Zlow(i,Output.xlow(i)) = 1;
end

end
```

### A.19 Robust_Constraints.m

```
function Output = Robust_Constraints(Input,bound)

Output = Input;
% Robust state set
Output.Set_X_robust = Polyhedron('lb',Output.x_min_robust,...
                                 'ub',Output.x_max_robust);
% Robust input set
Output.Set_U_robust = Polyhedron('lb',Output.U_min_robust,...
                                 'ub',Output.U_max_robust);

% Additional input constraint tightening for lower level tracking error
% Upper bound lower level tracking error
Output.Dx_LL_max = bound*ones(length(Output.xf),1);
% Lower bound lower level tracking error
Output.Dx_LL_min = -Output.Dx_LL_max;
% Generate lower level tracking error set
Output.Set_Dx_LL = Polyhedron('lb',Output.Dx_LL_min,'ub',Output.Dx_LL_max);
% Tighten constraints
Output.Set_U_robust = minus(Output.Set_U_robust,...
                            affineMap(Output.Set_Dx_LL,-Output.Mfast'));
Output.Set_U_robust.minHRep;   % Minimum representation of constraints
```

```
% State Set constraint reordering
indexes = [];
for i = 1:size(Output.Set_X_robust.H,2)-1
    indexes = [indexes;find(Output.Set_X_robust.H(:,i) == 1)];
end
for i = 1:size(Output.Set_X_robust.H,2)-1
    indexes = [indexes;find(Output.Set_X_robust.H(:,i) == -1)];
end
Output.Set_X_robust=Polyhedron('A',Output.Set_X_robust.H(indexes,1:end-1),...
                                 'b',Output.Set_X_robust.H(indexes,end));

% Input Set constraint reordering
indexes = [];
for i = 1:size(Output.Set_U_robust.H,2)-1
    indexes = [indexes;find(Output.Set_U_robust.H(:,i) == 1)];
end
for i = 1:size(Output.Set_U_robust.H,2)-1
    indexes = [indexes;find(Output.Set_U_robust.H(:,i) == -1)];
end
Output.Set_U_robust=Polyhedron('A',Output.Set_U_robust.H(indexes,1:end-1),...
                                 'b',Output.Set_U_robust.H(indexes,end));

end
```

### A.20 State_Reordering.m

```
function Output = State_Reordering(Input,Full)

Output = Input;

% Initial reordering matrix
Output.Z = eye(Full.Nv);
% Identify order of states
[trash,reorder] = ismember([Output.Sys_Vs';...
                           setdiff(Full.Sys_Vs,Output.Sys_Vs)],Full.Sys_Vs);
% Reorder matrix
Output.Z = Output.Z(:,reorder);
% Add sink states to matrix
Output.Z = blkdiag(Output.Z,eye(Output.Nt));

end
```

### A.21 Level_1_Controller_Gen.m

```
function [ Output ] = Level_1_Controller_Gen(Input, weightings, constraints)

Output = Input;

% Use nominal or robust constraints based on flag
if constraints.robustOn
    Output.x_A = Output.Set_X_robust.H(:,1:end-1);
    Output.x_b = Output.Set_X_robust.H(:,end);
    Output.U_A = Output.Set_U_robust.H(:,1:end-1);
    Output.U_b = Output.Set_U_robust.H(:,end);
```

```
else
    Output.x_A = Output.Set_X.H(:,1:end-1);
    Output.x_b = Output.Set_X.H(:,end);
    Output.U_A = Output.Set_U.H(:,1:end-1);
    Output.U_b = Output.Set_U.H(:,end);
end
% Initialize Yalmip variables
x_ = sdpvar(repmat(Output.Nv,1,Output.horizon+1),...
            repmat(1,1,Output.horizon+1));
P_ = sdpvar(repmat(Output.Ne,1,Output.horizon),...
            repmat(1,1,Output.horizon));
Pin_ = sdpvar(repmat(Output.Ns,1,Output.horizon),...
            repmat(1,1,Output.horizon));
xt_ = sdpvar(repmat(Output.Nt,1,Output.horizon+1),...
            repmat(1,1,Output.horizon+1));
xlowDes_ = sdpvar(repmat(length(Output.xf),1,1),...
            repmat(1,1,1));
xlow_    = sdpvar(repmat(length(Output.xf),1,1),...
            repmat(1,1,1));
lambdas_ = sdpvar(repmat(Output.Ns,1,Output.horizon),...
            repmat(1,1,Output.horizon));
lambdat_ = sdpvar(repmat(Output.Nt,1,Output.horizon+1),...
            repmat(1,1,Output.horizon+1));

% Initalize objective function and constraints
objs = 0;
cons = [];

% Formulate optimization problem at each step in the prediction horizon
for k = 1:Output.horizon
    % Define full power flow vector
    P_full = Output.T*P_{k}+Output.Y*diag(lambdas_{k})*Pin_{k};
    % Nominal state tracking
    objs = objs + weightings.x*norm(x_{k+1} - Output.x0,2)^2;
    % Nominal power tracking
    objs = objs + weightings.u*norm(P_full-Output.Mab'*Output.Z*...
              [x_{k};xlowDes_;diag(lambdat_{k})*xt_{k}] - Output.u0,2)^2;
    % Minimize power
    objs = objs + weightings.u_eff*norm(P_full-Output.Mab'*Output.Z*...
              [x_{k};xlowDes_;diag(lambdat_{k})*xt_{k}] - Output.u_min,2)^2;
    % Minimize changes to desired disturbances
    objs = objs + 1e6*norm(lambdas_{k}-1,2)^2;
    objs = objs + 1e6*norm(lambdat_{k}-1,2)^2;
    % Constrain system dynamics
    cons = [cons, x_{k+1} == x_{k}+Output.Br_hat*P_{k}+...
                              Output.Vr_hat*diag(lambdas_{k})*Pin_{k}];
    % Constrain states
    cons = [cons, Output.x_A*x_{k} <= Output.x_b];
    % Constrain inputs
    cons = [cons, Output.U_A*(P_full-Output.Mab'*Output.Z*...
             [x_{k};xlowDes_;diag(lambdat_{k})*xt_{k}]) <= Output.U_b];
    % Constrain inputs for next timestep
    cons = [cons, Output.U_A*(P_full-Output.Mab'*Output.Z*...
             [x_{k+1};xlowDes_;diag(lambdat_{k})*xt_{k+1}]) <= Output.U_b];
    % If low state bounds on, constrain low states to be close to desired
    if constraints.lowerStateBoundsOn
        cons = [cons, -constraints.Dx_LL_max_value/2 <= ...
                     (xlow_ - xlowDes_) <= constraints.Dx_LL_max_value/2];
    end
    % Constrain last constant states at end of prediction horizon
    if k == Output.horizon
        cons = [cons, x_{k+1} == x_{k}];
    end
end
opts = sdpsettings('solver','+gurobi');
Output.Controller = optimizer(cons,objs,opts,{x_{1},Pin_{:},xt_{:},xlow_},...
                              [x_,P_,lambdas_,lambdat_,{xlowDes_}]);

end
```

## A.22 Level_i_Controller_Gen.m

```
function [ Output ] = Level_i_Controller_Gen(Input, weightings, constraints)

Output = Input;

% Use nominal or robust constraints based on flag
if constraints.robustOn
    Output.x_A = Output.Set_X_robust.H(:,1:end-1);
    Output.x_b = Output.Set_X_robust.H(:,end);
    Output.U_A = Output.Set_U_robust.H(:,1:end-1);
    Output.U_b = Output.Set_U_robust.H(:,end);
else
    Output.x_A = Output.Set_X.H(:,1:end-1);
    Output.x_b = Output.Set_X.H(:,end);
    Output.U_A = Output.Set_U.H(:,1:end-1);
    Output.U_b = Output.Set_U.H(:,end);
end
% Initialize Yalmip variables
x_ = sdpvar(repmat(Output.Nv,1,Output.horizon+1),...
            repmat(1,1,Output.horizon+1));
P_ = sdpvar(repmat(Output.Ne,1,Output.horizon),...
            repmat(1,1,Output.horizon));
Pin_ = sdpvar(repmat(Output.Ns,1,Output.horizon),...
            repmat(1,1,Output.horizon));
xt_ = sdpvar(repmat(Output.Nt,1,Output.horizon+1),...
            repmat(1,1,Output.horizon+1));
xdotDes_ = sdpvar(repmat(max(size(Output.Zup,1),1),1,Output.horizon),...
            repmat(1,1,Output.horizon));
PoutDes_ = sdpvar(repmat(max(size(Output.Zout,1),1),1,Output.horizon),...
            repmat(1,1,Output.horizon));
xlowDes_ = sdpvar(repmat(max(size(Output.Zlow,1),1),1,1),repmat(1,1,1));
xlow_    = sdpvar(repmat(max(length(Output.xf),1),1,1),repmat(1,1,1));

% Initalize objective function and constraints
objs = 0;
cons = [];

% Formulate optimization problem at each step in the prediction horizon
```

```
for k = 1:Output.horizon
    % Define full power flow vector
    P_full = Output.T*P_{k}+Output.Y*Pin_{k};
    % Nominal state tracking
    objs = objs + weightings.x*norm(x_{k+1} - Output.x0,2)^2;
    % Nominal power tracking
    objs = objs + weightings.u*norm(P_full-Output.Mab'*Output.Z*...
                                 [x_{k};xlow_;xt_{k}] - Output.u0,2)^2;
    % Minimize power
    objs = objs + weightings.u_eff*norm(P_full-Output.Mab'*Output.Z*...
                             [x_{k};xlow_;xt_{k}] - Output.u_min,2)^2;
    % Constrain system dynamics
    cons = [cons, x_{k+1} ==
x_{k}+Output.Br_hat*P_{k}+Output.Vr_hat*Pin_{k}];
    % Constrain states
    cons = [cons, Output.x_A*x_{k} <= Output.x_b];
    % Constrain inputs
    cons = [cons, Output.U_A*(P_full-Output.Mab'*Output.Z*...
                             [x_{k};xlow_;xt_{k}]) <= Output.U_b];
    % Constrain inputs for next timestep
    cons = [cons, Output.U_A*(P_full-Output.Mab'*Output.Z*...
                             [x_{k+1};xlow_;xt_{k+1}]) <= Output.U_b];
    % If state tracking on, constrain states to track desired trajectories
    if constraints.stateTrackingOn
        cons = [cons, Output.Zup*inv(diag(Output.Caps))*...
                  (-Output.M_upper*P_full+Output.D*Pin_{k}) == xdotDes_{k}];
    end
    % If outlet power tracking on, constrain outlet powers to track desired
    if constraints.pOutTrackingOn
        cons = [cons, Output.Zout*P_full == PoutDes_{k}];
    end
    % If low state bounds on, constrain low states to be close to desired
    if constraints.lowerStateBoundsOn
        cons = [cons, -constraints.Dx_LL_max_value/2 <= ...
          (Output.Zlow*x_{k+1} - xlowDes_) <= constraints.Dx_LL_max_value/2];
    end
end
opts = sdpsettings('solver','+gurobi');
Output.Controller = optimizer(cons,objs,opts,...
    {x_{1},Pin_{:},xt_{:},xdotDes_{:},PoutDes_{:},xlowDes_,xlow_},[x_,P_]);

end
```

## A.23 Level_N_Controller_Gen.m

```
function [ Output ] = Level_N_Controller_Gen( Input, weightings, constraints
)

Output = Input;

% Use nominal or robust constraints based on flag
if constraints.robustOn
    Output.x_A = Output.Set_X_robust.H(:,1:end-1);
    Output.x_b = Output.Set_X_robust.H(:,end);
    Output.U_A = Output.Set_U_robust.H(:,1:end-1);
    Output.U_b = Output.Set_U_robust.H(:,end);
else
    Output.x_A = Output.Set_X.H(:,1:end-1);
    Output.x_b = Output.Set_X.H(:,end);
    Output.U_A = Output.Set_U.H(:,1:end-1);
    Output.U_b = Output.Set_U.H(:,end);
end
% Initialize Yalmip variables
x_ = sdpvar(repmat(Output.Nv,1,Output.horizon+1),...
                            repmat(1,1,Output.horizon+1));
P_ = sdpvar(repmat(Output.Ne,1,Output.horizon),...
                            repmat(1,1,Output.horizon));
Pin_ = sdpvar(repmat(Output.Ns,1,Output.horizon),...
                            repmat(1,1,Output.horizon));
xt_ = sdpvar(repmat(Output.Nt,1,Output.horizon),...
                            repmat(1,1,Output.horizon));
xdotDes_ = sdpvar(repmat(max(size(Output.Zup,1),1),1,Output.horizon),...
                            repmat(1,1,Output.horizon));
PoutDes_ = sdpvar(repmat(max(size(Output.Zout,1),1),1,Output.horizon),...
                            repmat(1,1,Output.horizon));
xlowDes_ = sdpvar(repmat(max(size(Output.Zlow,1),1),1,1),repmat(1,1,1));

% Initalize objective function and constraints
objs = 0;
cons = [];

% Formulate optimization problem at each step in the prediction horizon
for k = 1:Output.horizon
    % Nominal state tracking
    objs = objs + weightings.x*norm(x_{k+1} - Output.x0,2)^2;
    % Nominal power tracking
    objs = objs + weightings.u*norm(P_{k}-Output.Mab'*...
                                    [x_{k};xt_{k}] - Output.u0,2)^2;
    % Minimize power
    objs = objs + weightings.u_eff*norm(P_{k}-Output.Mab'*...
                                    [x_{k};xt_{k}] - Output.u_min,2)^2;
    % Constrain system dynamics
    cons = [cons, x_{k+1} == x_{k}+Output.B*P_{k}+Output.V1*Pin_{k}];
    % Constrain states
    cons = [cons, Output.x_A*x_{k} <= Output.x_b];
    % Constrain inputs
    cons = [cons, Output.U_A*(P_{k}-Output.Mab'*...
                                    [x_{k};xt_{k}]) <= Output.U_b];
    % If state tracking on, constrain states to track desired trajectories
    if constraints.stateTrackingOn
        cons = [cons, Output.Zup*inv(diag(Output.Caps))*...
                  (-Output.M_upper*P_{k}+Output.D*Pin_{k}) == xdotDes_{k}];
    end
    % If outlet power tracking on, constrain outlet powers to track desired
    if constraints.pOutTrackingOn
        cons = [cons, Output.Zout*P_{k} == PoutDes_{k}];
    end
    % If low state bounds on, constrain low states to be close to desired
    if constraints.lowerStateBoundsOn
        cons = [cons, -constraints.Dx_LL_max_value/2 <= ...
```

```
            (Output.Zlow*x_{k+1} - xlowDes_) <= constraints.Dx_LL_max_value/2];
        end
    end
    opts = sdpsettings('solver','+gurobi');%,'verbose',2);
    Output.Controller = optimizer(cons,objs,opts,...
             {x_{1},Pin_{:},xt_{:},xdotDes_{:},PoutDes_{:},xlowDes_},[x_,P_]);

    end
```

## A.24 Call_Level_1_Controller.m

```
function [ Output ] = Call_Level_1_Controller( Input, plot, time )

Output = Input;

% Configure inputs to the controller
Inputs = cell(1,1+2*Output.horizon+1+1);
Inputs(1) = {Output.x0};
Inputs(2:Output.horizon+1) = ...
                    mat2cell(Output.Pin,Output.Ns,ones(1,Output.horizon));
Inputs(Output.horizon+2:2*Output.horizon+2) = ...
                    mat2cell(Output.xt,Output.Nt,ones(1,Output.horizon+1));
Inputs(2*Output.horizon+3) = {Output.xlow0};

% Start timing controller
if time == 1
    tic;
end
% Call controller
[Outputs,Output.feasible] = Output.Controller{Inputs};
% Record controller solve time
if time == 1
    Output.T_calc = toc;
end
% Display if optimization problem was infeasible
if Output.feasible ~= 0
    disp([Output.Name, ' infeasible at time = ', num2str(Output.Time),...
          ' with code = ', num2str(Output.feasible)])
end
% Output variables
Output.x = cell2mat(Outputs(:,1:Output.horizon+1));
Output.P = cell2mat(Outputs(:,Output.horizon+2:2*Output.horizon+1));
Output.lambdas = cell2mat(Outputs(:,2*Output.horizon+2:3*Output.horizon+1));
Output.lambdat = cell2mat(Outputs(:,3*Output.horizon+2:4*Output.horizon+2));
Output.xlowDes = cell2mat(Outputs(:,4*Output.horizon+3));
Output.x(:,1) = Output.x0;

if plot == 1
    figure;
    subplot(4,1,1);stairs(Output.x(:,1:end-1)')
    subplot(4,1,2);stairs(Output.P')
    subplot(4,1,3);stairs(Output.lambdas')
    subplot(4,1,4);stairs(Output.lambdat')
end
```

```
end
```

### A.25 Call_Level_i_Controller.m

```
function [ Output ] = Call_Level_i_Controller( Input, plot, time )

Output = Input;

% Configure inputs to the controller
Inputs = cell(1,1+4*Output.horizon+1+1+1);
Inputs(1) = {Output.x0};
Inputs(2:Output.horizon+1) = ...
                    mat2cell(Output.Pin,Output.Ns,ones(1,Output.horizon));
Inputs(Output.horizon+2:2*Output.horizon+2) = ...
                    mat2cell(Output.xt,Output.Nt,ones(1,Output.horizon+1));
Inputs(2*Output.horizon+3:3*Output.horizon+2) = ...
   mat2cell(Output.xdotDes,max(size(Output.Zup,1),1),ones(1,Output.horizon));
Inputs(3*Output.horizon+3:4*Output.horizon+2) = ...

mat2cell(Output.PoutDes,max(size(Output.Zout,1),1),ones(1,Output.horizon));
Inputs(4*Output.horizon+3) = {Output.xlowDes};
Inputs(4*Output.horizon+4) = {Output.xlow0};

% Start timing controller
if time == 1
    tic;
end
% Call controller
[Outputs,Output.feasible] = Output.Controller{Inputs};
% Record controller solve time
if time == 1
    Output.T_calc = toc;
end
% Display if optimization problem was infeasible
if Output.feasible ~= 0
    disp([Output.Name, ' infeasible at time = ', num2str(Output.Time),...
          ' with code = ', num2str(Output.feasible)])
end
% Output variables
Output.x = cell2mat(Outputs(:,1:Output.horizon+1));
Output.P = cell2mat(Outputs(:,Output.horizon+2:2*Output.horizon+1));
Output.x(:,1) = Output.x0;

if plot == 1
    figure;
    subplot(2,1,1);stairs(Output.x(:,1:end-1)')
    subplot(2,1,2);stairs(Output.P')
end

end
```

### A.26 Call_Level_N_Controller.m

```
function [ Output ] = Call_Level_N_Controller( Input, plot, time )

Output = Input;

% Configure inputs to the controller
Inputs = cell(1,1+4*Output.horizon+1);
Inputs(1) = {Output.x0};
Inputs(2:Output.horizon+1) = ...
                    mat2cell(Output.Pin,Output.Ns,ones(1,Output.horizon));
Inputs(Output.horizon+2:2*Output.horizon+1) = ...
                    mat2cell(Output.xt,Output.Nt,ones(1,Output.horizon));
Inputs(2*Output.horizon+2:3*Output.horizon+1) = ...
   mat2cell(Output.xdotDes,max(size(Output.Zup,1),1),ones(1,Output.horizon));
Inputs(3*Output.horizon+2:4*Output.horizon+1) = ...

mat2cell(Output.PoutDes,max(size(Output.Zout,1),1),ones(1,Output.horizon));
Inputs(4*Output.horizon+2) = {Output.xlowDes};
% Start timing controller
if time == 1
    tic;
end
% Call controller
[Outputs,Output.feasible] = Output.Controller{Inputs};
% Record controller solve time
if time == 1
    Output.T_calc = toc;
end
% Display if optimization problem was infeasible
if Output.feasible ~= 0
    disp([Output.Name, ' infeasible at time = ', num2str(Output.Time), ...
          ' with code = ', num2str(Output.feasible)])
end
% Output variables
Output.x = cell2mat(Outputs(:,1:Output.horizon+1));
Output.P = cell2mat(Outputs(:,Output.horizon+2:2*Output.horizon+1));
Output.x(:,1) = Output.x0;

if plot == 1
    figure;
    subplot(2,1,1);stairs(Output.x(:,1:end-1)')
    subplot(2,1,2);stairs(Output.P')
end

end
```

### A.27 Three_Level_Sim.slx

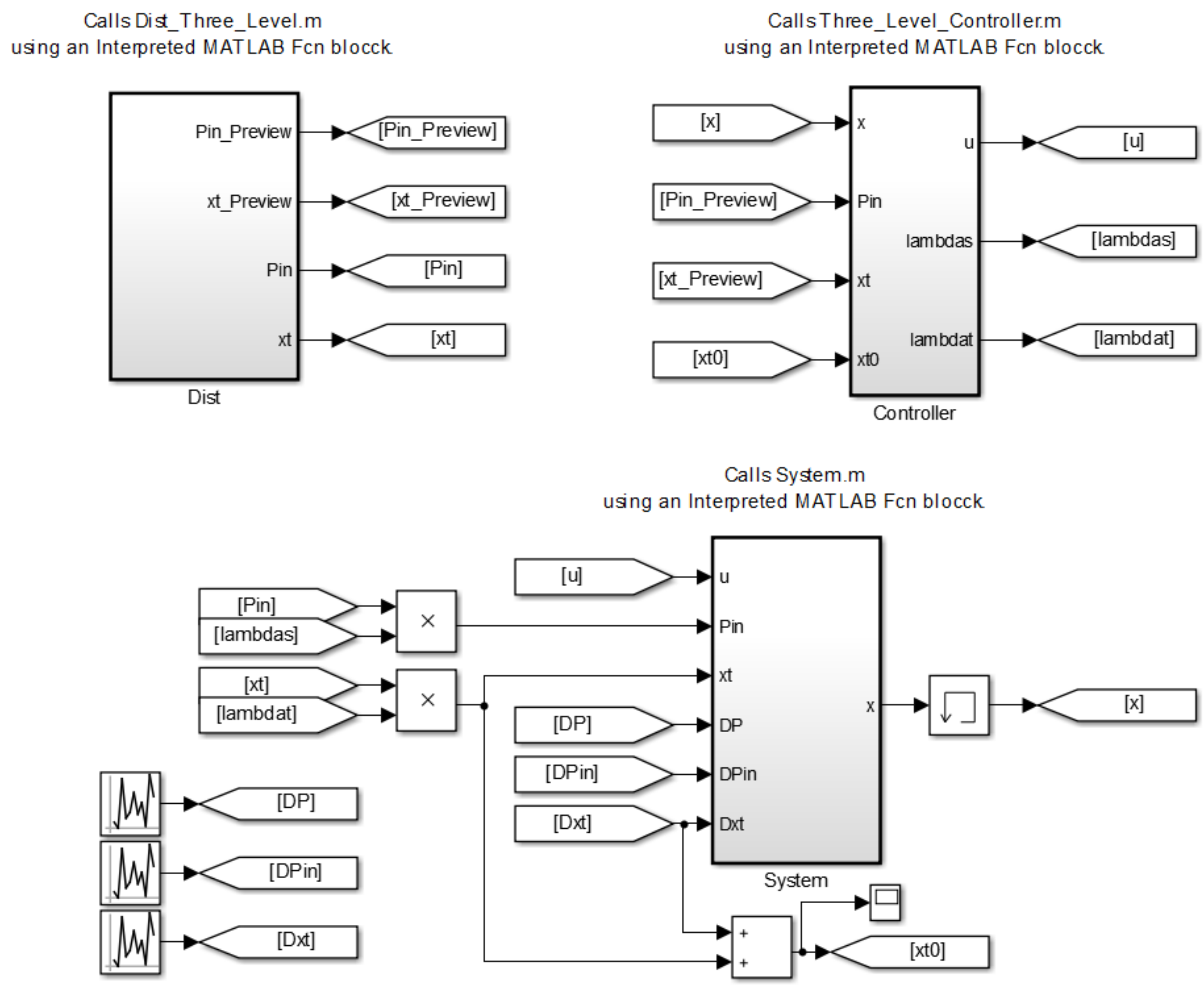


**Figure A.3 Simulink model used to simulate the robust hierarchical controller.**

## A.28 Three_Level_Controller.m

```
function out = Three_Level_Controller(in)

% Define persistent variables
persistent Sys SS1 SS2 SS3 SS4 S1r S2r Vehr x_pred plot_ time i_S i_SS...
           Pin_V xt_V Pfull_V x_V xdot_V...
           Pin_S1 xt_S1 Pfull_S1 x_S1 xdot_S1...
           Pin_S2 xt_S2 Pfull_S2 x_S2 xdot_S2

% Time
t = in(1);
% Load variables from work space
if t == 0
    Sys = evalin('base','Sys');
    SS1 = evalin('base','SS1');
    SS2 = evalin('base','SS2');
    SS3 = evalin('base','SS3');
    SS4 = evalin('base','SS4');
    S1r = evalin('base','S1r');
    S2r = evalin('base','S2r');
    Vehr = evalin('base','Vehr');
    x_pred = Sys.x0;
    plot_ = 0; time = 1;
    i_S = 0;
    i_SS = 0;
end
% Length of input vectors
l_x0 = Sys.Nv;
l_Pin = Sys.Ns*Vehr.horizon;
l_xt = Sys.Nt*(Vehr.horizon+1);
l_xt0 = Sys.Nt;
% Indices for each input vector
t1 = 2;     t2 = t1+l_x0-1;
t3 = t2+1;  t4 = t3+l_Pin-1;
t5 = t4+1;  t6 = t5+l_xt-1;
t7 = t6+1;  t8 = t7+l_xt0-1;
% Define inputs
x0 = in(t1:t2);
Pin = reshape(in(t3:t4),Vehr.horizon,Sys.Ns)';
xt = reshape(in(t5:t6),Vehr.horizon+1,Sys.Nt)';
xt0 = in(t7:t8);

% Call at Vehicle-level time step
if mod(t,Vehr.DT) == 0

    % Inputs to Vehr controller
    Vehr.x0 = x_pred(Vehr.Sys_Vs);
    Vehr.Pin = Pin;
    Vehr.xt = xt;
    Vehr.xlow0 = x_pred(Vehr.xf);
    % Call controller
    Vehr.Time = t;
    [ Vehr ] = Call_Level_1_Controller( Vehr, plot_, time );
    % Upsample trajectories to be used by lower level controllers
    [Pin_V,xt_V,Pfull_V,x_V,xdot_V] = Rate_Transition(Vehr,S1r);
    % Reinitialize index used by lower level controller
    i_S = 0;
end
% Call at System-level time step
if mod(t,S1r.DT) == 0
    % Increment index used to select values from upper level trajectories
    i_S = i_S+1;
    % Inputs to S1r controller
    S1r.x0 = x_pred(S1r.Sys_Vs);
    S1r.Pin = Pin_V(:,i_S:i_S+S1r.horizon-1);
    S1r.xt = x_V([4 3],i_S:i_S+S1r.horizon); % Vertices 11 and 7
    S1r.xdotDes = xdot_V([1],i_S:i_S+S1r.horizon-1);
    S1r.PoutDes = Pfull_V([8 9],i_S:i_S+S1r.horizon-1);
    S1r.xlowDes = Vehr.xlowDes([1 2 7]);  % Vertices 3, 4, and 10
    S1r.xlow0 = Vehr.xlowDes([5 6]);   % Vertices 8 and 9
    % Call controller
```

```
        S1r.Time = t;
        [ S1r ] = Call_Level_i_Controller( S1r, plot_, time );
        % Upsample trajectories to be used by lower level controllers
        [Pin_S1,xt_S1,Pfull_S1,x_S1,xdot_S1] = Rate_Transition(S1r,SS1);
            % Inputs to S2r controller
        S2r.x0 = x_pred(S2r.Sys_Vs);
        S2r.Pin = Pfull_V([8 9],i_S:i_S+S2r.horizon-1);
        S2r.xt = xt_V(:,i_S:i_S+S2r.horizon); % Vertices xt1 and xt2
        S2r.xdotDes = xdot_V([2 3 4],i_S:i_S+S2r.horizon-1);
        S2r.PoutDes = Pfull_V([13 18],i_S:i_S+S2r.horizon-1);
        S2r.xlowDes = Vehr.xlowDes([3 4]);  % Vertices 5 and 6
        S2r.xlow0 = Vehr.xlowDes([8]);   % Vertex 12
        % Call controller
        S2r.Time = t;
        [ S2r ] = Call_Level_i_Controller( S2r, plot_, time );
        % Upsample trajectories to be used by lower level controllers
        [Pin_S2,xt_S2,Pfull_S2,x_S2,xdot_S2] = Rate_Transition(S2r,SS3);
        % Reinitialize index used by lower level controller
        i_SS = 0;
    end
    % Call at Subsystem-level time step
    if mod(t,SS1.DT) == 0
        % Increment index used to select values from upper level trajectories
        i_SS = i_SS + 1;
        % Inputs to SS1 controller
        SS1.x0 = x_pred(SS1.Sys_Vs);
        SS1.Pin = Pin_S1(1,i_SS:i_SS+SS1.horizon-1);
        SS1.xt = [x_S1(4,i_SS:i_SS+SS1.horizon-1);...
                  xt_S1(1,i_SS:i_SS+SS1.horizon-1)]; % Vertices 10 and 11
        SS1.xdotDes = xdot_S1([2],i_SS:i_SS+SS1.horizon-1);
        SS1.PoutDes = Pfull_S1([4 8],i_SS:i_SS+SS1.horizon-1);
        SS1.xlowDes = Vehr.xlowDes([5 6]);  % Vertices 8 and 9
        % Call controller
        SS1.Time = t;
        [ SS1 ] = Call_Level_N_Controller( SS1, plot_, time );
        % Inputs to SS2 controller
        SS2.x0 = x_pred(SS2.Sys_Vs);
        SS2.Pin = [Pin_S1(2,i_SS:i_SS+SS2.horizon-1);...
                   Pfull_S1(4,i_SS:i_SS+SS2.horizon-1)];
        SS2.xt = xt_S1(2,i_SS:i_SS+SS2.horizon-1); % Vertex 7
        SS2.xdotDes = xdot_S1([1 3 4],i_SS:i_SS+SS2.horizon-1);
        SS2.PoutDes = Pfull_S1([9],i_SS:i_SS+SS2.horizon-1);
        SS2.xlowDes = 0;
        % Call controller
        SS2.Time = t;
        [ SS2 ] = Call_Level_N_Controller( SS2, plot_, time );
        % Inputs to SS3 controller
        SS3.x0 = x_pred(SS3.Sys_Vs);
        SS3.Pin = [Pin_S2(1,i_SS:i_SS+SS3.horizon-1);...
                   Pfull_S2(5,i_SS:i_SS+SS3.horizon-1)];
        SS3.xt = xt_S2(1,i_SS:i_SS+SS3.horizon-1); % Vertex xt1
        SS3.xdotDes = xdot_S2([1 2 5],i_SS:i_SS+SS3.horizon-1);
        SS3.PoutDes = Pfull_S2([4],i_SS:i_SS+SS3.horizon-1);
        SS3.xlowDes = 0;
        % Call controller
        SS3.Time = t;
        [ SS3 ] = Call_Level_N_Controller( SS3, plot_, time );
        % Inputs to SS4 controller
        SS4.x0 = x_pred(SS4.Sys_Vs);
        SS4.Pin = Pin_S2(2,i_SS:i_SS+SS4.horizon-1);
        SS4.xt = [x_S2(5,i_SS:i_SS+SS1.horizon-1);...
                  xt_S2(2,i_SS:i_SS+SS1.horizon-1)];    % Vertices 11 and xt2
        SS4.xdotDes = xdot_S2([3 4],i_SS:i_SS+SS4.horizon-1);
        SS4.PoutDes = Pfull_S2([5 9],i_SS:i_SS+SS4.horizon-1);
        SS4.xlowDes = Vehr.xlowDes([8]);  % Vertex 12
        % Call controller
        SS4.Time = t;
        [ SS4 ] = Call_Level_N_Controller( SS4, plot_, time );
    end
    % Calculate the desired power flows
    P1 = SS1.P(:,1) + SS1.K*(x0(SS1.Sys_Vs)-x_pred(SS1.Sys_Vs));
    P2 = SS2.P(:,1) + SS2.K*(x0(SS2.Sys_Vs)-x_pred(SS2.Sys_Vs));
    P3 = SS3.P(:,1) + SS3.K*(x0(SS3.Sys_Vs)-x_pred(SS3.Sys_Vs));
    P4 = SS4.P(:,1) + SS4.K*(x0(SS4.Sys_Vs)-x_pred(SS4.Sys_Vs));
    % Assemble the desired power flows
    P_bar  = [P1(1:4);P2(1:3);P1(5);P2(4);P3(1:4);P4(4);P4(1:3);P4(5)];
    % Calculate the control input
    u = inv(Sys.beta)*(P_bar-Sys.Mab'*[x0;xt0]);
    % Assemble predicted nominal states at the next time step
    x_pred = [SS2.x(1,2);SS3.x(1,2);SS1.x(1,2);SS2.x(2,2);SS3.x(2,2);...
              SS4.x(1,2);SS4.x(2,2);SS1.x(2,2);SS1.x(3,2);SS2.x(3,2);...
              SS3.x(3,2);SS4.x(3,2)];
    % Output inputs and throttling variables
    out = [u;Vehr.lambdas(:,1);Vehr.lambdat(:,1)];

    end
```

## A.29 Rate_Transition.m

```
function [Pin,xt,Pfull,x,xdot] = Rate_Transition(Upper,Lower)

% Determine Pin and xt over Lower-level prediction horizon based on
% values determined by the Upper-level controller
Pin = [];
xt = [];
% If signals are coming from the Vehicle level, calculate "throttled" values
if strcmp(Upper.Name,'Vehr')
 for i = 1:size(Upper.Pin,2)
  Pin = [Pin repmat(Upper.lambdas(:,i).*Upper.Pin(:,i),1,Upper.DT/Lower.DT)];
  xt = [xt repmat(Upper.lambdat(:,i).*Upper.xt(:,i),1,Upper.DT/Lower.DT)];
 end
  Pfull = [];
    for i = 1:size(Upper.P,2)
        P = Upper.T*Upper.P+Upper.Y*(Upper.lambdas.*Upper.Pin);
        Pfull = [Pfull repmat(P(:,i),1,Upper.DT/Lower.DT)];
    end
else
    for i = 1:size(Upper.Pin,2)
        Pin = [Pin repmat(Upper.Pin(:,i),1,Upper.DT/Lower.DT)];
        xt = [xt repmat(Upper.xt(:,i),1,Upper.DT/Lower.DT)];
    end
    Pfull = [];
    for i = 1:size(Upper.P,2)
        P = Upper.T*Upper.P+Upper.Y*Upper.Pin;
        Pfull = [Pfull repmat(P(:,i),1,Upper.DT/Lower.DT)];
    end
end
% Determine state trajectory
x = [];
for i = 1:size(Upper.x,2)-1
    x = [x repmat(Upper.x(:,i),1,Upper.DT/Lower.DT)+...
                 (Upper.x(:,i+1)-Upper.x(:,i))/(Upper.DT/Lower.DT)*...
                 [0:Upper.DT/Lower.DT-1]];
end
% Determine rate of change of states
xdot = inv(diag(Upper.Caps))*(-Upper.M_upper*Pfull+Upper.D*Pin);

end
```

### A.30 System.m

```
function out = System(in)

% Define persistent variables
persistent Sys

% Time
t = in(1);
% Load variables from work space
if t == 0
    Sys = evalin('base','Sys');
    Sys.x = Sys.x0;
end
% Length of input vectors
l_u  = size(Sys.u0,1);
l_Pin = Sys.Ns;
l_xt = Sys.Nt;
l_DP = Sys.Ne;
l_DPin = Sys.Ns;
l_Dxt = Sys.Nt;
% Indices for each input vector
t1 = 2;     t2 = t1+l_u-1;
t3 = t2+1;  t4 = t3+l_Pin-1;
t5 = t4+1;  t6 = t5+l_xt-1;
t7 = t6+1;  t8 = t7+l_DP-1;
t9 = t8+1;  t10 = t9+l_DPin-1;
t11 = t10+1;  t12 = t11+l_Dxt-1;
% Define inputs
Sys.u = in(t1:t2);
Sys.Pin = in(t3:t4);
Sys.xt = in(t5:t6);
Sys.DP = in(t7:t8);
Sys.DPin = in(t9:t10);
Sys.Dxt = in(t11:t12);
% System state at next time step
```

```
Sys.x = Sys.A*Sys.x+Sys.B*Sys.beta*Sys.u+...
    Sys.V1*(Sys.Pin+Sys.DPin)+Sys.V2*(Sys.xt+Sys.Dxt)+Sys.V3*Sys.DP;
% Output system state at next time step
out = [Sys.x];

end
```

### A.31 Dist_Three_Level.m

```
function out = Dist_Three_Level(in)

% Define persistent variables
persistent Vehr Sys xt Pin T_Pin T_xt

% Time
t = in(1);
% Load variables from work space
if t == 0
    Vehr = evalin('base','Vehr');
    Sys = evalin('base','Sys');

    Pin = 0*ones(Sys.Ns,Sys.Tsim+Vehr.DT*Vehr.horizon);
    xt = 0*ones(Sys.Nt,Sys.Tsim+Vehr.DT*(Vehr.horizon+1));

    Pin(1,201:300) = 3;
    Pin(2,401:500) = 3;
end
% Define time vector
if mod(t,Vehr.DT) == 0
    T_Pin = t+1:Vehr.DT:t+Vehr.DT*Vehr.horizon;
    T_xt = t+1:Vehr.DT:t+Vehr.DT*(Vehr.horizon+1);
end
% Output disturbance over prediction horizon and current values
if Sys.preview
    out = [reshape(Pin(:,T_Pin)',Sys.Ns*Vehr.horizon,1);...
           reshape(xt(:,T_xt)',Sys.Nt*(Vehr.horizon+1),1);...
           Pin(:,t+1);xt(:,t+1)];
else
    out =
[reshape(repmat(Pin(:,t+1),1,length(T_Pin))',Sys.Ns*Vehr.horizon,1);...

reshape(repmat(xt(:,t+1),1,length(T_xt))',Sys.Nt*(Vehr.horizon+1),1);...
           Pin(:,t+1);xt(:,t+1)];
end

end
```