

ABSTRACT

BOBINYEC, KAREN S. Observer Construction for Systems of Differential Algebraic Equations using Completions. (Under the direction of Dr. Stephen L. Campbell.)

This dissertation presents the results from researching observer construction for systems of differential algebraic equations using completions. Physical systems of interest in control theory are sometimes described by ordinary differential equations (ODEs) and algebraic equations, forming a system of differential algebraic equations (DAEs). Challenges from solving a system of DAEs come from the first derivative of its state vector having a singular coefficient matrix and from possibly implicit algebraic constraints defining its solution manifold. These challenges exist for observing a system of DAEs as well, leading to processes that require the system to meet certain assumptions.

Our approach of pairing a completion of a system of DAEs with observers for systems of ODEs has been successful in estimating the states of linear time-invariant and linear time-varying example systems of DAEs without requiring any structural assumptions about the systems. The material on completions and two of the observers is review, but observing a system of DAEs by observing a completion is a new approach. Two other observers included in this dissertation result from our research and take advantage of the constraints characterizing the solution manifold of the system of DAEs. In particular, our maximally reduced observer is shown experimentally to be the preferred observer when compared with traditional full-order and reduced-order observers. With the potential for our approach to be extended to nonlinear systems of DAEs, constructing observers for systems of DAEs using completions looks to be a general approach applicable to both linear and nonlinear systems.

© Copyright 2013 by Karen S. Bobinyec

All Rights Reserved

Observer Construction for Systems of Differential Algebraic Equations using Completions

by
Karen S. Bobinyec

A dissertation submitted to the Graduate Faculty of
North Carolina State University
in partial fulfillment of the
requirements for the Degree of
Doctor of Philosophy

Applied Mathematics

Raleigh, North Carolina

2013

APPROVED BY:

Dr. Robert H. Martin

Dr. Ralph C. Smith

Dr. Hien T. Tran

Dr. Stephen L. Campbell
Chair of Advisory Committee

DEDICATION

In honor of my parents and in memory of Dixie.

BIOGRAPHY

Karen S. Bobinyec was born in Dover, NJ; lived in Saugerties, NY, during her preschool years; and has lived in Cary, NC, since age 5 when her father was relocated to RTP by IBM. She graduated *summa cum laude* from Meredith College in Raleigh, NC, in 2004 with a BA in Political Studies, a BS in Mathematics, and minors in German and Physical Chemistry. The author also received an MS in Applied Mathematics from North Carolina State University in 2008. Her primary hobby is dance (African, ballet, clogging, jazz, and tap), currently enrolled in her 24th year of classes. Karen's favorite places to plan trips to and visit are Walt Disney World, Disneyland, Washington, DC, the Wildwoods in NJ, and New York City. Someday she hopes to return to England, Vancouver, and PEI and to finally travel to Germany.

ACKNOWLEDGEMENTS

I would like to acknowledge

- Dr. Stephen Campbell, who was willing to be my advisor;
- Dr. Robert Martin, for without his classes, I might not have remained in graduate school;
- Dr. Kimberly Drake for the internship at NAVSEA in Philadelphia, PA;
- my committee members and grad. school representative for being flexible with scheduling;
- my Meredith College professors in mathematics, history and politics, and the Honors Program for inspiring me to continue learning and for their dedication to their students;
- my parents John and Susan Bobinyec, my grandmother Berniece Stoops, and my friends Jennifer Rowland, Amanda Beasley, Sejal Patel, Martene Fair Stanberry, and Ellen Peterson Swanson for their support;
- friends, officemates Kim Spayd and Chuck Wessell for picking me to be their third wheel;
- and the Holly Springs School of Dance for the opportunity to dance.

TABLE OF CONTENTS

LIST OF TABLES	vii
LIST OF FIGURES	viii
Chapter 1 Introduction	1
Chapter 2 Completions	6
2.1 Least Squares Completion	7
2.2 Stabilized Least Squares Completion	8
2.3 Alternative Stabilized Completion	9
2.3.1 Linear Time-Invariant Case	10
2.3.2 Linear Time-Varying Case	14
2.4 Completion Stability	15
Chapter 3 Observers	17
3.1 Observable and Detectable	17
3.2 The Full-Order Observer	19
3.2.1 Linear Time-Invariant Case	20
3.2.2 Linear Time-Varying Case	22
3.3 The Reduced-Order Observer	25
3.3.1 Linear Time-Invariant Case	25
3.3.2 Linear Time-Varying Case	28
3.4 The Maximally Reduced Observer	34
3.5 DAE Manifold Observer	36
3.6 Completion Stability Revisited	38
Chapter 4 Theory	39
4.1 Completions and Observability	39
4.2 Full-order and Reduced-order Observers	44
4.3 Comments on Linear Time-Varying Theory	49
Chapter 5 Linear Time-Invariant Example	52
5.1 Mechanics Example Introduction	52
5.2 Observer Implementation	54
5.3 Observer Construction Code	83
Chapter 6 Linear Time-Varying Example	85
6.1 Circuit Example Introduction	85
6.2 Observer Implementation	91
6.2.1 Example with Positive Resistors	92
6.2.2 Example with Negative Resistors	143
6.3 Observer Construction Code	182

Chapter 7 Conclusions, Contributions, and Future Research Topics	184
7.1 Conclusions	184
7.2 Contributions	186
7.3 Future Research Topics	187
REFERENCES	189
APPENDICES	201
Appendix A Smooth Decomposition	202
A.1 Smooth Decomposition	202
A.2 First Derivatives	206
A.3 Alternatives to the Smooth Decomposition	211
Appendix B MATLAB Code	212
B.1 Chapter 5 Code	212
B.2 Chapter 6 Code	252
B.3 Code for both Examples	294

LIST OF TABLES

Table 6.1	For each Λ_{SL} , the number of $\tilde{A}(t)$'s eigenvalues with a negative real part.	98
Table 6.2	For each Λ_{SL} , the ranks of $W_o(t)$ and their time intervals for two output matrices.	98
Table 6.3	For each Λ_{SL} , the ranks of $W_o(t)$ and their time intervals for output matrix C_c	99
Table 6.4	For each Λ_A , the number of $\tilde{A}(t)$'s eigenvalues with a negative real part.	102
Table 6.5	For each Λ_A , the ranks of $W_o(t)$ and their time intervals for two output matrices.	103
Table 6.6	For each Λ_A , the ranks of $W_o(t)$ and their time intervals for output matrix C_c	103
Table 6.7	For each Λ_{SL} , the number of $\tilde{A}(t)$'s eigenvalues with a negative real part.	147
Table 6.8	For each Λ_{SL} , the ranks of $W_o(t)$ and their time intervals for two output matrices.	148
Table 6.9	For each Λ_{SL} , the ranks of $W_o(t)$ and their time intervals for output matrix C_f	148
Table 6.10	For each Λ_A , the number of $\tilde{A}(t)$'s eigenvalues with a negative real part.	151
Table 6.11	For each Λ_A , the ranks of $W_o(t)$ and their time intervals for two output matrices.	152
Table 6.12	For each Λ_A , the ranks of $W_o(t)$ and their time intervals for output matrix C_f	152

LIST OF FIGURES

Figure 5.1	$x(t)$, solution of system (5.4).	57
Figure 5.2	$\tilde{x}(t)$, completion solutions of system (5.4).	58
Figure 5.3	$x(t) - \tilde{x}(t)$, comparing exact solution with completion solutions.	58
Figure 5.4	SLSC/FO, ASC/FO observers plotted with solution $x(t)$. (C_a , $\lambda = 2$, $\rho = -1$)	63
Figure 5.5	$x(t) - \hat{x}(t)$ comparison. (C_a , $\lambda = 2$, $\rho = -1$)	64
Figure 5.6	LSC/MR observer's $x(t) - \hat{x}(t)$; the MR observer differences are SLSC – ASC and SLSC – LSC. Neither the completion nor λ influences the MR observer. (C_a , $\lambda = 0$, $\lambda = 2$, $\rho = -1$)	64
Figure 5.7	$x(t) - \hat{x}(t)$ progression as ρ is varied, 2nd component of $x_1(t)$. (C_a , $\lambda = 2$) .	66
Figure 5.8	$x(t) - \hat{x}(t)$ progression as ρ is varied, 1st component of $x_2(t)$. (C_a , $\lambda = 2$) .	66
Figure 5.9	$x(t) - \hat{x}(t)$ progression as ρ is varied, $x_3(t)$. (C_a , $\lambda = 2$)	67
Figure 5.10	$x(t) - \hat{x}(t)$ progression as λ is varied, 2nd component of $x_1(t)$. (C_a , $\rho = -1$) .	67
Figure 5.11	$x(t) - \hat{x}(t)$ progression as λ is varied, 1st component of $x_2(t)$. (C_a , $\rho = -1$) .	68
Figure 5.12	$x(t) - \hat{x}(t)$ progression as λ is varied, $x_3(t)$. (C_a , $\rho = -1$)	69
Figure 5.13	$q(t)$ (solid) and $\hat{q}(t)$ (dashed) for two extended output matrices. (C_a , $\rho = -1$)	70
Figure 5.14	MR observers' $q(t) - \hat{q}(t)$ for two extended output matrices. (C_a , $\rho = -1$)	70
Figure 5.15	$x(t)$ (solid) and $\hat{x}(t)$ (dashed) for two extended output matrices; the MR observer difference is $(C_\Xi, C_a) - (C_\Xi, \mathcal{GF})$. The results in terms of $x(t)$, $\hat{x}(t)$ are the same. (C_a , $\rho = -1$)	70
Figure 5.16	$x(t) - \hat{x}(t)$ comparison. (C_c , $\lambda = 2$, $\rho = -1$)	73
Figure 5.17	$x(t) - \hat{x}(t)$ progression as ρ is varied, 1st component of $x_1(t)$. (C_c , $\lambda = 2$) .	75
Figure 5.18	$x(t) - \hat{x}(t)$ progression as ρ is varied, 2nd component of $x_2(t)$. (C_c , $\lambda = 2$) .	75
Figure 5.19	$x(t) - \hat{x}(t)$ progression as ρ is varied, $x_3(t)$. (C_c , $\lambda = 2$)	76
Figure 5.20	$x(t) - \hat{x}(t)$ progression as λ is varied, 1st component of $x_1(t)$. (C_c , $\rho = -1$) .	76
Figure 5.21	$x(t) - \hat{x}(t)$ progression as λ is varied, 2nd component of $x_2(t)$. (C_c , $\rho = -1$) .	78
Figure 5.22	$x(t) - \hat{x}(t)$ progression as λ is varied, $x_3(t)$. (C_c , $\rho = -1$)	78
Figure 5.23	Solution $p(t)$ and DAE/M observer $\hat{p}(t)$ plotted together; DAE/M observer's $p(t) - \hat{p}(t)$. (C_a , $\rho = -1$)	80
Figure 5.24	Solution $x(t)$ and DAE/M observer's $\hat{x}(t)$ plotted together; DAE/M observer's $x(t) - \hat{x}(t)$. This observer's $e(t)$ is comparable to the $e(t)$ for the ASC/FO observer with C_c , $\lambda = 2$, $\rho = -1$. (C_a , $\rho = -1$)	81
Figure 5.25	$x(t) - \hat{x}(t)$ progression as ρ is varied, DAE/M observer. (C_a)	81
Figure 5.26	Solution $x(t)$ and DAE/M observer's $\hat{x}(t)$ plotted together; the DAE/M observer difference is $C_a - C_c$. The DAE/M observers are the same for Cases 1 and 3.	81
Figure 5.27	Solution $x(t)$ and DAE/M observer's $\hat{x}(t)$ plotted together; DAE/M observer's $x(t) - \hat{x}(t)$. (C_d , $\rho = -1$)	82
Figure 6.1	Circuit Example	85
Figure 6.2	$\ \Phi(t, 0)\ $ on two intervals of time. $\ \Phi(t, 0)\ \rightarrow 0$ for each choice of Λ_{SL}	96

Figure 6.3	$\ \tilde{A}(t)\ $ for five choices of Λ_{SL} . Each $\ \tilde{A}(t)\ $ is bounded.	96
Figure 6.4	$\ \tilde{B}(t)\ $ for five choices of Λ_{SL} . Each $\ \tilde{B}(t)\ $ is bounded.	97
Figure 6.5	$\ \Phi(t, 0)\ $ on two intervals of time. $\ \Phi(t, 0)\ \rightarrow 0$ for each choice of Λ_A .	100
Figure 6.6	$\ \tilde{A}(t)\ $ for five choices of Λ_A . Each $\ \tilde{A}(t)\ $ is bounded.	101
Figure 6.7	$\ \tilde{B}(t)v(t)\ $ for an SLSC and an ASC. $\ \tilde{B}(t)\ $ cannot be computed for the ASC, but comparing $\ \tilde{B}(t)v(t)\ $ for the stabilized completions suggests the ASC $\ \tilde{B}(t)\ $ is bounded.	101
Figure 6.8	$\ \Phi(t, 0)\ $ fails to converge to zero for the LSC (Λ_L).	104
Figure 6.9	$\ \tilde{A}(t)\ $ and $\ \tilde{B}(t)\ $ are bounded for the LSC (Λ_L).	104
Figure 6.10	SLSC solution ($\Lambda_{4\text{SL}}$) on two intervals of time. Figure 6.10b suggests system (6.9)'s solution is periodic.	106
Figure 6.11	The ASC solution (Λ_{4A}) is comparable to the SLSC solution in Figure 6.10. The solution difference is SLSC – ASC.	106
Figure 6.12	The LSC solution (Λ_L) is comparable to the SLSC solution in Figure 6.10. The solution difference is SLSC – LSC.	106
Figure 6.13	SLSC/FO observer's $\tilde{x}(t) - \hat{x}(t)$ on two intervals of time. $e(t) \rightarrow 0$ by $t = 135$. ($C_a, \Lambda_{4\text{SL}}, \delta = 0.1, \eta = 1, \xi = 0.1$)	108
Figure 6.14	SLSC/FO observer's $\tilde{x}(t) - \hat{x}(t)$ on two intervals of time; SLSC solution and FO observer plotted together. $e(t) \rightarrow 0$ by $t_f = 45$. ($C_a, \Lambda_{4\text{SL}}, \delta = 0.1, \eta = 100, \xi = 0.1$)	109
Figure 6.15	ASC/FO observer's $\tilde{x}(t) - \hat{x}(t)$ on two intervals of time; ASC solution and FO observer plotted together. $e(t) \rightarrow 0$ by $t_f = 45$. ($C_a, \Lambda_{4A}, \delta = 0.1, \eta = 100, \xi = 0.1$)	109
Figure 6.16	Observer comparison. SLSC/FO and ASC/FO observers plotted together; the observer difference is SLSC/FO – ASC/FO. ($C_a, \Lambda_{4\text{SL}}, \Lambda_{4A}, \delta = 0.1, \eta = 100, \xi = 0.1$)	109
Figure 6.17	$\tilde{x}(t) - \hat{x}(t)$ progression as η is varied, SLSC/FO observer. Shortened time intervals provide closer views of initial differences. ($C_a, \Lambda_{4\text{SL}}, \delta = 0.1, \xi = 0.1$)	110
Figure 6.18	SLSC/FO observers' $\tilde{x}(t) - \hat{x}(t)$ as δ and η are varied. These estimation errors appear to be the same. ($C_a, \Lambda_{4\text{SL}}, \xi = 0.1$)	111
Figure 6.19	$\delta = 0.01$ SLSC/FO and $\eta = 10$ SLSC/FO observers plotted together; the SLSC/FO observer difference is $(\delta = 0.01) - (\eta = 10)$. Shortened time interval shows observers are not the same using $i_v(t)$ estimates as example. ($C_a, \Lambda_{4\text{SL}}, \xi = 0.1$)	111
Figure 6.20	SLSC/FO observer's $\tilde{x}(t) - \hat{x}(t)$; the SLSC/FO observer difference is $(\xi = 0.01) - (\xi = 0.1)$. Decreasing ξ does not visibly affect SLSC/FO observer convergence. ($C_a, \Lambda_{4\text{SL}}$)	113
Figure 6.21	SLSC/FO observer's $\tilde{x}(t) - \hat{x}(t)$ for $\Lambda_{2\text{SL}}$; $\Lambda_{2\text{SL}}, \Lambda_{4\text{SL}}$ SLSC/FO observers plotted together. Effects from changing Λ_{SL} are visible on $t \in [0, 2]$. ($C_a, \delta = 0.1, \eta = 100, \xi = 0.1$)	113

Figure 6.22 SLSC/FO observers' $\tilde{x}(t) - \hat{x}(t)$ as δ and ξ are varied. ($C_a, \Lambda_{4SL}, \eta = 100$)	113
Figure 6.23 Solution comparison. The ASC solution difference is SDM – SYM. (Λ_{4A})	114
Figure 6.24 Observer comparison. ASC/FO SDM observer's $\tilde{x}(t) - \hat{x}(t)$; the ASC/FO observer difference is SDM – SYM. ($C_a, \Lambda_{4SL}, \delta = 0.1, \eta = 100, \xi = 0.1$)	115
Figure 6.25 LSC/FO observer's $\tilde{x}(t) - \hat{x}(t)$ on two intervals of time. $e(t)$ fails to converge to zero by $t = 90$. ($C_a, \Lambda_L, \delta = 0.0001, \eta = 100, \xi = 0.1$)	116
Figure 6.26 LSC/FO observers' $\tilde{x}(t) - \hat{x}(t)$ as η is varied. Increasing η does not result in $e(t) \rightarrow 0$ by $t_f = 45$. ($C_a, \Lambda_L, \delta = 0.1, \xi = 0.0001$)	117
Figure 6.27 LSC/FO observer's $\tilde{x}(t) - \hat{x}(t)$ on three intervals of time. $e(t)$ fails to converge to zero by $t = 90$. ($C_a, \Lambda_L, \delta = 0.1, \eta = 1000, \xi = 0.0001$)	117
Figure 6.28 SLSC/RO observer's $\tilde{x}(t) - \hat{x}(t)$ on two intervals of time; SLSC/RO observer's $q(t) - \hat{q}(t)$. Transformation matrix R is a permutation matrix. ($C_a, \Lambda_{4SL}, \delta = 0.1, \eta = 1, \xi = 0.1$)	119
Figure 6.29 ASC/RO observer's $\tilde{x}(t) - \hat{x}(t)$ on two intervals of time; ASC/RO observer's $q(t) - \hat{q}(t)$. The order of the observer is reduced from 6 to 2. ($C_a, \Lambda_{4A}, \delta = 0.1, \eta = 1, \xi = 0.1$)	119
Figure 6.30 ASC/RO observer's $\tilde{x}(t) - \hat{x}(t)$ on two intervals of time. Integration tolerance error occurs for SLSC/RO observer (Λ_{4SL}) and same δ, η , and ξ . ($C_a, \Lambda_{4A}, \delta = 1, \eta = 1, \xi = 1$)	121
Figure 6.31 $\ \Phi_R(t, 0)\ $ is bounded; LSC/RO observer's $\tilde{x}(t) - \hat{x}(t)$ on two intervals of time. $e(t) \rightarrow 0$ by $t_f = 45$ for an LSC/RO observer. ($C_a, \Lambda_L, \delta = 0.1, \eta = 1000, \xi = 0.0001$)	121
Figure 6.32 $\tilde{x}(t) - x(t)$; equation (6.13) computes $x(t)$ without observer. (C_a)	121
Figure 6.33 SLSC/FO observers' $\tilde{x}(t) - \hat{x}(t)$ for C_b and C_c . ($\Lambda_{4SL}, \delta = 0.1, \eta = 100, \xi = 0.1$)	122
Figure 6.34 SLSC/FO observers' $\tilde{x}(t) - \hat{x}(t)$ as η is varied. For output matrix C_c , increasing η may not improve observer convergence. ($\Lambda_{4SL}, \delta = 0.1, \xi = 0.1$)	123
Figure 6.35 $\tilde{x}(t) - \hat{x}(t)$ progression as η is varied, SLSC/FO observer with output matrix C_c . Effects on state variable estimates are not uniform when increasing η . ($\Lambda_{4SL}, \delta = 0.1, \xi = 0.1$)	123
Figure 6.36 SLSC/FO observers' $\tilde{x}(t) - \hat{x}(t)$ as η is varied. For output matrix C_b , increasing η leads to $e(t) \rightarrow 0$ monotonically. ($\Lambda_{4SL}, \delta = 0.1, \xi = 0.1$)	124
Figure 6.37 The SLSC/FO observer difference is $C_b - C_c$ as η is varied. The SLSC/FO observer with output matrix C_b converges to $\tilde{x}(t)$ faster. ($\Lambda_{4SL}, \delta = 0.1, \xi = 0.1$)	124
Figure 6.38 SLSC/RO observers' $q(t) - \hat{q}(t)$ for C_b and C_c . ($\Lambda_{4SL}, \delta = 0.1, \eta = 1, \xi = 0.1$)	126
Figure 6.39 SLSC/RO observers' $q(t) - \hat{q}(t)$ for C_b and C_c . Shortened time intervals for results from Figure 6.38 show each reduced-order observer is estimating just the unmeasurable components of its transformed state vector. ($\Lambda_{4SL}, \delta = 0.1, \eta = 1, \xi = 0.1$)	126
Figure 6.40 SLSC/RO observers' $\tilde{x}(t) - \hat{x}(t)$ for C_b and C_c . ($\Lambda_{4SL}, \delta = 0.1, \eta = 1, \xi = 0.1$)	126

Figure 6.41 SLSC/RO observers' $\tilde{x}(t) - \hat{x}(t)$ as η is varied. For output matrix C_c , increasing η exchanges an estimate's accuracy for faster convergence. (Λ_{4SL} , $\delta = 0.1$, $\xi = 0.1$)	127
Figure 6.42 SLSC/RO observers' $\tilde{x}(t) - \hat{x}(t)$ as η is varied. For output matrix C_b , a maximum η may exist for convergence to occur by a specific time. (Λ_{4SL} , $\delta = 0.1$, $\eta = 0.1$)	127
Figure 6.43 $\tilde{x}(t) - x(t)$, SLSC (Λ_{4SL}) and $C_x = C_b$, vs $\times 10^{-6}$; equation (6.13) computes $x(t)$ without observer.	128
Figure 6.44 SLSC/MR observers' $q(t) - \hat{q}(t)$ as η is varied. For output matrix C_c , the SLSC/MR observer estimates just one component of $q(t)$. (Λ_{4SL} , $\delta = 0.1$, $\xi = 0.1$)	129
Figure 6.45 $q_6(t)$ plotted with $\hat{q}_6(t)$ estimated by the SLSC/MR observer when $\eta = 1$; shortened time interval shows how much varying η affects $\hat{q}_6(t)$. (C_c , $\delta = 0.1$, $\xi = 0.1$)	129
Figure 6.46 SLSC/MR observers' $\tilde{x}(t) - \hat{x}(t)$ as η is varied. Without plotting the estimation errors together on a shortened interval, increasing η would not appear to affect convergence. (C_c , Λ_{4SL} , $\delta = 0.1$, $\xi = 0.1$)	129
Figure 6.47 Solution comparison. The ASC solution difference is $\Lambda_{4A} - \Lambda_{3A}$	131
Figure 6.48 ASC/FO observers' $\tilde{x}(t) - \hat{x}(t)$ for Λ_{4A} and Λ_{3A} ; the ASC/FO observer difference is $\Lambda_{4A} - \Lambda_{3A}$. (C_b , $\delta = 0.1$, $\eta = 100$, $\xi = 0.1$)	131
Figure 6.49 SLSC/FO and ASC/FO observers' $\tilde{x}(t) - \hat{x}(t)$ plotted separately and together. (C_d , Λ_{4SL} , Λ_{4A} , $\delta = 0.1$, $\eta = 100$, $\xi = 0.1$)	132
Figure 6.50 $\ \Phi_3(t, 0)\ \rightarrow 0$, so the stabilized completions are detectable. (C_d)	132
Figure 6.51 SLSC/RO and ASC/RO observers' $\tilde{x}(t) - \hat{x}(t)$ plotted separately and together. (C_d , Λ_{4SL} , Λ_{4A} , $\delta = 0.1$, $\eta = 100$, $\xi = 0.1$)	134
Figure 6.52 $\tilde{x}(t) - x(t)$; equation (6.13) computes $x(t)$ without observer. (C_d)	134
Figure 6.53 SLSC/FO observers' $\tilde{x}(t) - \hat{x}(t)$ for C_e and C_f ; the SLSC/FO observer difference is $C_e - C_f$. (Λ_{4SL} , $\delta = 0.1$, $\eta = 1000$, $\xi = 0.1$)	135
Figure 6.54 ASC/FO observers' $\tilde{x}(t) - \hat{x}(t)$ for C_e and C_f ; the ASC/FO observer difference is $C_e - C_f$. (Λ_{4A} , $\delta = 0.1$, $\eta = 1000$, $\xi = 0.1$)	135
Figure 6.55 LSC/FO observer's $\tilde{x}(t) - \hat{x}(t)$ for $\eta = 1000$; $\tilde{x}(t) - \hat{x}(t)$ progression in $i_{r_2}(t)$ and $i_v(t)$ as η is varied. (C_e , Λ_L , $\delta = 0.1$, $\xi = 0.0001$)	137
Figure 6.56 LSC/FO observer's $\tilde{x}(t) - \hat{x}(t)$ for $\eta = 1000$ on two intervals of time; $\tilde{x}(t) - \hat{x}(t)$ progression in $e_1(t)$ as η is varied. (C_f , Λ_L , $\delta = 0.1$, $\xi = 0.0001$)	137
Figure 6.57 $\ \Phi_3(t, 0)\ \rightarrow 0$ for the stabilized completions but not for the LSC. For output matrix C_e , the LSC is not detectable.	138
Figure 6.58 $\ \Phi_3(t, 0)\ \rightarrow 0$ for the stabilized completions but not for the LSC. For output matrix C_f , the LSC is not detectable.	138
Figure 6.59 SLSC/RO observers' $\tilde{x}(t) - \hat{x}(t)$ for C_e and C_f ; the SLSC/RO observer difference is $C_e - C_f$. (Λ_{4SL} , $\delta = 0.1$, $\eta = 100$, $\xi = 0.1$)	140
Figure 6.60 ASC/RO observers' $\tilde{x}(t) - \hat{x}(t)$ for C_e and C_f ; the ASC/RO observer difference is $C_e - C_f$. (Λ_{4A} , $\delta = 0.1$, $\eta = 100$, $\xi = 0.1$)	140
Figure 6.61 LSC/RO observers' $\tilde{x}(t) - \hat{x}(t)$ for C_e and C_f . $e(t)$ fails to converge to zero by $t_f = 45$. (Λ_L , $\delta = 0.1$, $\eta = 100$, $\xi = 10^{-5}$)	140

Figure 6.62 SLSC/MR observers' $\tilde{x}(t) - \hat{x}(t)$ for C_e and C_f ; the SLSC/MR observer difference is $C_e - C_f$. (Λ_{4SL} , $\delta = 0.1$, $\eta = 1000$, $\xi = 0.1$)	141
Figure 6.63 ASC/MR observer's $\tilde{x}(t) - \hat{x}(t)$; the observer difference is SLSC/MR – ASC/MR. (Case 4, Λ_{4A} , Λ_{4SL} , $\delta = 0.1$, $\eta = 1000$, $\xi = 0.1$)	142
Figure 6.64 LSC/MR observer's $\tilde{x}(t) - \hat{x}(t)$; the observer difference is SLSC/MR – LSC/MR. (Case 4, Λ_{4L} , Λ_{4SL} , $\delta = 0.1$, $\eta = 1000$, $\xi = 0.1$)	142
Figure 6.65 $\tilde{x}(t) - \hat{x}(t)$ progression in $i_l(t)$ as η is varied, MR observer. (Case 4, $\delta = 0.1$, $\xi = 0.1$)	142
Figure 6.66 $\ \Phi(t, 0)\ $, on two intervals of time, fails to converge to zero for the six Λ_{SL} options.	144
Figure 6.67 $\ \Phi(t, 0)\ $ comparison. Shortened time intervals provide closer views of norms' initial behaviors with and without result for component Λ_{5SL}	145
Figure 6.68 $\ \tilde{A}(t)\ $ for six choices of Λ_{SL} . Each $\ \tilde{A}(t)\ $ is bounded.	145
Figure 6.69 $\ \tilde{B}(t)\ $ for six choices of Λ_{SL} . Each $\ \tilde{B}(t)\ $ is bounded.	146
Figure 6.70 $\ \Phi(t, 0)\ $ comparison. $\ \Phi(t, 0)\ $ fails to converge to zero for the five Λ_A options, and a shortened time interval provides a closer view of norms' initial behaviors.	149
Figure 6.71 $\ \tilde{A}(t)\ $ for five choices of Λ_A . Each $\ \tilde{A}(t)\ $ is bounded.	150
Figure 6.72 $\ \tilde{B}(t)v(t)\ $ for an SLSC and an ASC. $\ \tilde{B}(t)\ $ cannot be computed for the ASC, but comparing $\ \tilde{B}(t)v(t)\ $ for the stabilized completions suggests the ASC $\ \tilde{B}(t)\ $ is bounded.	150
Figure 6.73 $\ \Phi(t, 0)\ $ fails to converge to zero for the LSC (Λ_L). vs $\times 10^4$	153
Figure 6.74 $\ \tilde{A}(t)\ $ and $\ \tilde{B}(t)\ $ are bounded for the LSC (Λ_L).	153
Figure 6.75 SLSC solution (Λ_{3SL}) on two intervals of time. The solution of the linear time-varying example with negative resistors is neither periodic nor bounded.	155
Figure 6.76 The ASC solution (Λ_{4A}) and the SLSC solution (Λ_{3SL}) grow apart in some state variables. The solution difference is SLSC – ASC.	155
Figure 6.77 The LSC solution (Λ_L) and the SLSC solution (Λ_{3SL}) grow apart in some state variables. The solution difference is SLSC – LSC.	155
Figure 6.78 SLSC/FO observer's $\tilde{x}(t) - \hat{x}(t)$ on two intervals of time. $e(t) \rightarrow 0$ by $t_f = 45$. (C_a , Λ_{3SL} , $\delta = 0.1$, $\eta = 100$, $\xi = 0.001$)	156
Figure 6.79 SLSC/FO observers' $\tilde{x}(t) - \hat{x}(t)$ as η is varied. $e(t) \rightarrow 0$ by $t_f = 45$ even as η is decreased. (C_a , Λ_{3SL} , $\delta = 0.1$, $\xi = 0.001$)	157
Figure 6.80 ASC/FO observers' $\tilde{x}(t) - \hat{x}(t)$ for Λ_{1A} and Λ_{4A} . These estimation errors appear to be the same. (C_a , $\delta = 0.1$, $\eta = 1$, $\xi = 0.001$)	157
Figure 6.81 Solution and observer comparisons. The ASC solution and ASC/FO observer differences are $\Lambda_{1A} - \Lambda_{4A}$. (C_a , $\delta = 0.1$, $\eta = 1$, $\xi = 0.001$)	157
Figure 6.82 SLSC/RO observer's $\tilde{x}(t) - \hat{x}(t)$; $e(t) \rightarrow 0$ by $t_f = 45$ but $\ \Phi_R(t, 0)\ $ does not go to zero by $t = 135$. (C_a , Λ_{3SL} , $\delta = 0.1$, $\eta = 1$, $\xi = 0.001$)	160

Figure 6.83 ASC/RO observer's $\tilde{x}(t) - \hat{x}(t)$ on two intervals of time; $\ \Phi_R(t, 0)\ \rightarrow 0$ by $t = 135$. ($C_a, \Lambda_{4A}, \delta = 0.1, \eta = 1, \xi = 0.001$)	160
Figure 6.84 $\tilde{x}(t) - x(t)$; equation (6.13) computes $x(t)$ without observer. (C_a)	160
Figure 6.85 SLSC reduced-order systems' rank ($W_o(t)$) examination for Λ_{3SL} and Λ_{2SL} . (C_b)	161
Figure 6.86 SLSC reduced-order systems' $\ \Phi_R(t, 0)\ $ for Λ_{3SL} and Λ_{2SL} . $\ \Phi_R(t, 0)\ $ fails to go to zero by $t = 135$ for both Λ_{SL} options. (C_b)	162
Figure 6.87 SLSC reduced-order systems' $\ \Phi_R(t, 0)\ $ for Λ_{3SL} and Λ_{2SL} . On $t \in [0, 45]$, $\ \Phi_R(t, 0)\ $ does not grow as quickly for Λ_{2SL} . (C_b)	162
Figure 6.88 SLSC $\underline{2}$ /RO observers' $\tilde{x}(t) - \hat{x}(t)$ as δ, η , and ξ are varied; the SLSC $\underline{2}$ /RO observer difference is $(\xi = 0.001) - (\delta = 0.001)$. Integration tolerance error occurs after $t = 20$. (C_b)	163
Figure 6.89 SLSC $\underline{2}$ /RO observers' $\tilde{x}(t) - \hat{x}(t)$ as δ, η , and ξ are varied. With the modifications to the gain matrix parameters, the SLSC $\underline{2}$ /RO observers are no longer the same. (C_b)	163
Figure 6.90 SLSC $\underline{2}$ /RO observers' $\tilde{x}(t) - \hat{x}(t)$ as η is varied. Increasing η improves observer convergence, but $e(45)$ is not within $(-10^{-3}, 10^{-3})$. ($C_b, \delta = 10^{-8}, \xi = 0.1$)	165
Figure 6.91 SLSC $\underline{2}$ /RO observers' $\tilde{x}(t) - \hat{x}(t)$ without and with moving horizon. $e(45)$ is within desired interval for moving horizon observer. ($C_b, \delta = 10^{-8}, \eta = 100, \xi = 0.1$)	165
Figure 6.92 ASC reduced-order system's $\ \Phi_R(t, 0)\ $ does not grow as quickly as results in Figure 6.86; ASC/RO observer's $\tilde{x}(t) - \hat{x}(t)$. This observer's convergence properties and rate are preferable to those for the SLSC $\underline{2}$ /RO observer's. ($C_b, \Lambda_{4A}, \delta = 0.1, \eta = 1, \xi = 10^{-6}$)	166
Figure 6.93 SLSC $\underline{2}$ /RO observers' $\tilde{x}(t) - \hat{x}(t)$ as RelTol is varied. The rank of $W_o(t)$ is affected by RelTol value. ($C_b, \delta = 0.001, \eta = 100, \xi = 0.1$)	166
Figure 6.94 Full-order observers' $\tilde{x}(t) - \hat{x}(t)$. ($C_b, \delta = 0.1, \eta = 100, \xi = 0.001$)	167
Figure 6.95 Full-order observers' $\tilde{x}(t) - \hat{x}(t)$ (results from Figure 6.94 on $t \in [0, 5]$). Shortened interval shows these completions' full-order observers have bet- ter convergence properties and rates than their respective reduced-order observers. ($C_b, \delta = 0.1, \eta = 100, \xi = 0.001$)	167
Figure 6.96 $\tilde{x}(t) - x(t)$; equation (6.13) computes $x(t)$ without observer. (C_b)	168
Figure 6.97 Full-order observers' $\tilde{x}(t) - \hat{x}(t)$. $e(t) \rightarrow 0$ by $t_f = 45$. ($C_d, \delta = 0.1,$ $\eta = 100, \xi = 0.001$)	169
Figure 6.98 Full-order observers' $\tilde{x}(t) - \hat{x}(t)$ (results from Figure 6.97 on $t \in [0, 10]$). Shortened interval provides closer view for estimation error comparison. ($C_d, \delta = 0.1, \eta = 100, \xi = 0.001$)	169
Figure 6.99 LSC/FO observer's $\tilde{x}(t) - \hat{x}(t)$. $\ \Phi_3(t, 0)\ $ fails to converge to zero for the LSC. ($C_d, \Lambda_L, \delta = 0.1, \eta = 100, \xi = 0.001$)	170
Figure 6.100 $\ \Phi_3(t, 0)\ \rightarrow 0$, so the stabilized completions are detectable. (C_d)	170
Figure 6.101 Solution and observer comparisons. The ASC solution and ASC/FO ob- server differences are SDM – SYM. ($C_d, \Lambda_{4A}, \delta = 0.1, \eta = 100, \xi = 0.001$)	171

Figure 6.102 Reduced-order observers' $\tilde{x}(t) - \hat{x}(t)$. The observers' estimates of $e_2(t)$ are visibly different. (C_d , $\delta = 0.1$, $\eta = 1$, $\xi = 0.1$)	172
Figure 6.103 Reduced-order observers' $\tilde{x}(t) - \hat{x}(t)$. Observers with results in Figures 6.103b and 6.103c have similar convergence rates. (C_d , $\delta = 0.1$, $\xi = 0.1$)	172
Figure 6.104 Reduced-order observers' $\tilde{x}(t) - \hat{x}(t)$. Results in Figures 6.103b and 6.103c extended to $t = 135$. (C_d , $\delta = 0.1$, $\xi = 0.1$)	172
Figure 6.105 $\ \Phi_R(t, 0)\ $ is bounded on $t \in [0, 135]$ for the stabilized completions. (C_d)	173
Figure 6.106 LSC/RO observers' $\tilde{x}(t) - \hat{x}(t)$ as η is varied. The reduced-order system is unobservable. (C_d , $\Lambda_{\underline{L}}$, $\delta = 0.1$, $\xi = 0.1$)	173
Figure 6.107 $\tilde{x}(t) - x(t)$; equation (6.13) computes $x(t)$ without observer. (C_d)	174
Figure 6.108 $\ \Phi_3(t, 0)\ \rightarrow 0$ for the stabilized completions but not for the LSC. The LSC is not detectable. (C_f)	175
Figure 6.109 SLSC/FO observers' $\tilde{x}(t) - \hat{x}(t)$ as η is varied. For these values of η , $e(45)$ is not within $(-10^{-3}, 10^{-3})$. (C_f , Λ_{3SL} , $\delta = 0.1$, $\xi = 0.001$)	176
Figure 6.110 SLSC/FO observers' $\tilde{x}(t) - \hat{x}(t)$ as η is varied. When $\eta = 50$, $e(45) \in (-10^{-3}, 10^{-3})$. (C_f , Λ_{3SL} , $\delta = 0.1$, $\xi = 0.001$)	176
Figure 6.111 ASC/FO observers' $\tilde{x}(t) - \hat{x}(t)$ as η is varied plotted separately on $t \in [0, 45]$ and together on $t \in [5, 15]$. (C_f , Λ_{4A} , $\delta = 0.1$, $\xi = 0.001$)	176
Figure 6.112 SLSC/RO observer's $\tilde{x}(t) - \hat{x}(t)$ on two intervals of time; $\ \Phi_R(t, 0)\ $ is bounded. (C_f , Λ_{3SL} , $\delta = 0.1$, $\eta = 100$, $\xi = 0.1$)	177
Figure 6.113 SLSC/RO observers' $\tilde{x}(t) - \hat{x}(t)$ as η is varied. When $\eta = 100$, $e(t) \in (-10^{-3}, 10^{-3})$. (C_f , Λ_{3SL} , $\delta = 0.1$, $\xi = 0.1$)	177
Figure 6.114 ASC/RO observers' $\tilde{x}(t) - \hat{x}(t)$ as η is varied. When $\eta = 10$, $e(45) \in (-10^{-3}, 10^{-3})$. (C_f , Λ_{4A} , $\delta = 0.1$, $\xi = 0.1$)	178
Figure 6.115 ASC/RO observer's $\tilde{x}(t) - \hat{x}(t)$ on the extended time interval; $\ \Phi_R(t, 0)\ $ is bounded. (C_f , Λ_{4A} , $\delta = 0.1$, $\eta = 10$, $\xi = 0.1$)	178
Figure 6.116 ASC/RO observer's $\tilde{x}(t) - \hat{x}(t)$ at two end times. The estimation error may appear to grow numerically as time progresses due to oscillation. (C_f , Λ_{4A} , $\delta = 0.1$, $\eta = 10$, $\xi = 0.1$)	179
Figure 6.117 $\tilde{x}(t) - \hat{x}(t)$ for LSC/FO and LSC/RO observers. FO and RO observers constructed using the LSC with output matrix C_f cannot be designed to estimate $\tilde{x}(t)$. ($\Lambda_{\underline{L}}$, $\delta = 0.1$)	179
Figure 6.118 $\ \Phi_{\bar{R}}(t, 0)\ \rightarrow 0.5$ for each completion. (C_f)	180
Figure 6.119 $\tilde{x}(t) - \hat{x}(t)$ for SLSC/MR, ASC/MR, and LSC/MR observers. (C_f , $\delta = 0.1$, $\eta = 1000$, $\xi = 0.1$)	181
Figure 6.120 The observer differences, SLSC – ASC and SLSC – LSC, confirm the construction of the MR observer distinguishes between neither completion nor Λ . $\tilde{x}(t) - \hat{x}(t)$ progression in $i_l(t)$; varying η affects the MR observer's convergence. (C_f , Λ_{3SL} , Λ_{4A} , $\Lambda_{\underline{L}}$, $\delta = 0.1$, $\xi = 0.1$)	181
Figure 6.121 Completion solution and MR observer estimate in $i_l(t)$ for two examples (negative, positive resistors). The estimation error difference is $(e(t), \text{Neg}) - (e(t), \text{Pos})$. The completions' solutions and observers are not the same but the observers' estimation errors are. (C_f , $\delta = 0.1$, $\eta = 1000$, $\xi = 0.1$)	181

Chapter 1

Introduction

The state of a physical system is not always available. Measurements may be hindered by cost or sensor inaccessibility, for example. But in some applications, knowing the state is important. In [122], Luenberger proposed a method for estimating the state so the estimate in lieu of a true value could be used in a feedback control law. This method designed an observer, which is a system of equations that utilizes the available measurements, or observations, on the inputs and outputs of the physical system to produce an asymptotic estimate of the state. The system of equations is constructed from a mathematical model of the physical system. Initial observer design focused on physical systems described by linear time-invariant systems of ordinary differential equations (ODEs). As research progressed, the systems became more general, incorporating time-varying coefficients and nonlinear structure. Research has also been extended to constructing observers for physical systems described by systems of differential algebraic equations (DAEs), the focus of this dissertation. Since the development of observers to estimate the state for use in a feedback control law, observers have been designed to estimate a system's unknown parameters [80]; applied to fault estimation, detection, and isolation [103], [117], [171] and disturbance detection [119]; implemented for tracking [162]; and included in power systems [67] and circuit theory [124].

A system $F(\dot{x}(t), x(t), u(t), t) = 0$ with a singular $\partial F / \partial \dot{x}(t)$ is a system of DAEs. These systems consist of both ODEs and algebraic equations so that $\dot{x}(t)$ cannot be solved for explicitly. The algebraic equations are identified as explicit constraints but implicit or hidden constraints exist if the index of the system of DAEs is greater than 1 [59]. The index is an identifier of the system of DAEs and two definitions found in [28] are presented.

Definition 1.1. *The index k of a system of DAEs is the minimum number of times the system is differentiated with respect to t in order to uniquely determine the first derivative of the state vector as a continuous function of the state vector and t .*

Definition 1.2. *The index k of a linear system of DAEs $E(t)\dot{x}(t) + F(t)x(t) = B(t)u(t)$ is the minimum number of times the system is differentiated with respect to t so that the coefficient matrices $\mathcal{E}(t)$, $\mathcal{F}(t)$ from the derivative array equations $\mathcal{E}(t)w(t) + \mathcal{F}(t)x(t) = \mathcal{B}(t)v(t)$ meet the following conditions:*

1. $\begin{bmatrix} \mathcal{E}(t) & \mathcal{F}(t) \end{bmatrix}$ is full row rank for all t ;
2. $\mathcal{E}(t)$ has constant rank; and
3. if $x(t)$ is n dimensional, then $\mathcal{E}(t)b(t) = 0$ for some vector $b(t)$ implies the first n entries of $b(t)$ are zero for all t .

Definition 1.2, a specific version of general Definition 1.1, includes checks easily implemented in an index-determining algorithm for a time-invariant system. Note for Definition 1.2, the derivative array equations are revisited in Chapter 2 and conditions 1 and 3 are related to the concept of 1-fullness. These definitions describe what may be clarified in the literature as the differentiation index [47]. Other indices associated with systems of DAEs such as the perturbation index [47], the differential algebraic index [140], the Kronecker index [148], the tractability index [148], or the geometric index [148] are unnecessary for the research in this dissertation; thus, the adjective differentiation is assumed when discussing the index. A system of ODEs is an index 0 system of DAEs, so the higher the index, the greater the challenge is for solving a system of DAEs compared with solving a system of ODEs.

A linear system of DAEs $E(t)\dot{x}(t) + F(t)x(t) = B(t)u(t)$ is solvable if, given $B(t)u(t)$, the system's solution is determined by a consistent initial condition [28]. These consistent initial conditions must satisfy all constraints, and if they do not, an incorrect system of DAEs is solved. Due to possibly hidden constraints, consistent initial conditions may not be obvious. Research on methods for finding consistent initial conditions includes [30], [57], [112], and [115], but the examples to which we apply our observer construction approach in Chapters 5 and 6 have consistent initial conditions that can be found by hand. We also assume the linear systems of DAEs are solvable, so methods such as those in [48] or [123] are not required for checking the solvability of a system. Note, if a linear system of DAEs is solvable, then matrix $\mathcal{E}(t)$ from the derivative array equations in Definition 1.2 has constant rank even though matrix $E(t)$ may not [56].

Theory for a linear time-invariant system of DAEs $E\dot{x}(t) + Fx(t) = Bu(t)$ incorporates the matrix pencil $sE + F$ [82]. A linear time-invariant system of DAEs is solvable if the matrix pencil is regular; that is, if there exists an s such that $\det(sE + F)$ does not equal zero. The values of s for which $\det(sE + F) = 0$ are the finite eigenvalues of the linear time-invariant system of DAEs and are referred to as the matrix pencil eigenvalues in this dissertation.

Two additional terms in this dissertation that require clarification are smooth and solution manifold. We assume the coefficient matrices of the linear systems of DAEs are smooth, meaning at least as many derivatives exist as are needed. The solution manifold, or the manifold of consistent initial conditions in [35], is characterized by the constraints of the system of DAEs and is where the solutions of a system of DAEs live for a consistent initial condition [41].

Other phrases in the literature synonymous with DAEs are descriptor [78], general or generalized state space [76], implicit [5], semistate [131], and singular [116]. This variety in terminology exists because of separate initial investigations into these systems by different disciplines. Examples of applications involving systems of DAEs include chemistry [33], [164]; circuit theory [130], [146]; constrained mechanical systems [21], [81]; gas networks [141] (discrete time and observer consideration); power systems [93]; and robotic motion [131]. Article [36] also provides an example overview on higher index systems of DAEs.

This introduction to systems of DAEs includes the information that is necessary for understanding the discussion in this dissertation. For those readers interested in learning more about systems of DAEs, suggested references include [6], [28], and [106] and for some earlier material consider [5], [44], [45], and [116].

Observers for linear time-invariant systems of DAEs are numerous. Article [129] designs full-order and reduced-order observers for systems in descriptor standard form, a form dependent on the matrix pencil. Another form in [163], [165] from which to construct a reduced-order observer is the generalized staircase form. The authors express concern, though, about the computational efficiency of the algorithms required to transform a linear time-invariant system of DAEs into this special form. The observers in [95] are derived from Luenberger observer theory but the systems to be observed should be in normal form. The construction of the full-order and reduced-order observers in [70] requires the computation of more matrices than the one gain matrix commonly found in the design of observers for linear time-invariant systems of ODEs. The observers in [76] and [78] incorporate singular value decompositions, while article [150] focuses on generalized inverses. Proportional integral observers [169] and functional observers [68] are also proposed.

One observer for a linear time-invariant system of DAEs is not a system of ODEs but a system of DAEs and its construction is based on eigenvalue placement. In [22], the first derivative of the observer's state vector has the same singular coefficient matrix as the system to be observed. One of the requirements for this observer to asymptotically estimate the state is dependent on the regularity of a modified matrix pencil. The author of [153] also considers DAE observers for linear time-invariant systems of DAEs. Dependences on eigenvalue placement, matrix pencils, and special forms make it difficult to generalize these observers for estimating linear time-varying and nonlinear systems of DAEs.

An observer in [134] applies a reduction algorithm to a linear time-invariant system of DAEs in order to reveal a system that can be observed by a system of ODEs. Our observer construction approach for estimating the state of a linear system of DAEs also constructs an observer for a system of ODEs and utilizes a Luenberger observer (described in Chapter 3). Even though the eigenvalues of a linear time-varying system of ODEs do not provide any indication of the system's stability [170], eigenvalue placement techniques are used to design the observers in [61], [114], and [120]. The author of [114] writes to improve rather than promote this construction technique since eigenvalue placement is implemented for linear time-varying engineering applications.

The observers for linear time-varying systems of ODEs in [133] are also designed with an eigenvalue placement technique. This article's approach assumes the system to be observed is lexicographic, meaning the system is observable and has an observability matrix with the same rows linearly independent for all time. Article [132] extends the research in [133] to reduced-order observers, and a lexicographic assumption can also be found in [152]. In response to this restriction, the authors of [62] propose forming an augmented system that is lexicographic when the original one is not. Section 3.1 describes why computing, and thereby constructing an observer dependent on, the observability matrix is avoided for our time-varying observers. An observable assumption is also limiting, commented on by the authors of [94]. Other attempts at generalizing linear time-invariant observer designs for linear time-varying systems of ODEs incorporate canonical forms [149], [151].

Previous work in [18], [19], [56] constructs an observer for linear time-varying systems of DAEs similar to our approach, but our design takes advantage of recent developments in stabilized completions. Described in detail in Chapter 2, a completion of a system of DAEs is a system of ODEs and is one of many suggestions on how to solve these systems of differential and algebraic equations.

The research on solving systems of DAEs using iterative methods designed for systems of ODEs, commonly backward difference formulas, is expansive. A first attempt at solving linear time-invariant systems of DAEs iteratively implements a backward Euler method [84], [85], [86]. This method is reconsidered along with other numerical methods in [155]. Index 1 assumptions are required in [32] for an implicit Runge-Kutta method; in [102] to treat the system as a stiff ODE; and in [6], [142] for the computer program DASSL.

Reformulations also exist for systems of higher index so iterative methods designed for systems with a lower index (usually index 1) can be applied. However, reformulation stability and discretization efficiency are examined in [7], [9]. Stability is a concern because if reformulation produces a perturbed system or the iterative method produces a solution that fails to converge [110], the resulting solutions are for a system other than the original system of DAEs. An observer constructed from a perturbed system would then fail to estimate the true state. Even

for a system that is not initially perturbed, the solution of the system being solved may drift away from the true solution if the constraints characterizing the solution manifold of the system of DAEs are not satisfied. The authors of [8] suggest combining forward Euler discretization with backward Euler stabilization to minimize drift. More recently, three general integrators identified as explicit integration, implicit coordinate partitioning, and index one integration are solved using a Gauss-Newton iteration designed to satisfy any system constraints [55], [58], [173]. Index one integration is modified to become the completion considered in Section 2.3. Other articles on solving systems of DAEs include [72], [91], [143], [144].

Common structural assumptions on a system's index [37] (and references listed earlier) or on the system being in Hessenberg form (defined in Section 5.1) [39], [40], [66] may simplify the processes for solving linear systems of DAEs but do not lend themselves to developing a general observer. Our observer construction approach constructs observers for linear systems of DAEs using completions, which requires no assumptions on a system's structure and addresses the constraints concern expressed for systems of DAEs.

After reviewing three completions in Chapter 2 and two observers in Sections 3.2 and 3.3, observers resulting from this PhD research are described in Sections 3.4 and 3.5. Theoretical contributions are presented in Sections 4.1 and 4.3, while Section 4.2 includes established theory helpful for example analysis. The results from applying our observer construction approach to two examples, a linear time-invariant system of DAEs and a linear time-varying system of DAEs, are analyzed in Chapters 5 and 6, respectively. In Chapter 7 we draw conclusions, list articles and presentations in which the contributions from this research appear, and propose future research topics. MATLAB code is attached in Appendix B. At times, particularly in Chapter 6, the time derivative traditionally designated as $\dot{g}(t)$ may be designated as $g(t)'$ to improve readability.

Chapter 2

Completions

A completion of an index k linear system of DAEs

$$E(t)\dot{x}(t) + F(t)x(t) = B(t)u(t) \quad (2.1)$$

is a linear system of ODEs

$$\dot{\tilde{x}}(t) = \tilde{A}(t)\tilde{x}(t) + \tilde{B}(t) \begin{bmatrix} u(t) \\ \dot{u}(t) \\ \vdots \\ u^{(k)}(t) \end{bmatrix} = \tilde{A}(t)\tilde{x}(t) + \tilde{B}(t)v(t) \quad (2.2)$$

designed so the solutions of equation (2.2) contain those of equation (2.1). For an n -dimensional state vector $x(t)$, a linear system of DAEs is defined on the less than n -dimensional solution manifold. Equation (2.2) is described as a completion of (2.1) since it is defined on a complete n -dimensional space that has the solution manifold as a submanifold [46], [138]. Theory in [46] identifies a general form relating all possible completions of a linear system of DAEs.

The completion evolved from the i th order, j -block method ((i, j) -method) presented in [42]. The (i, j) -method uses the expansions from an implicit backward Euler method to assemble a j -block dimensional system that is then solved in the least squares sense. Articles written responding to the (i, j) -method include [11], [12]. The larger dimensional system in [42] is replaced in [50] by the derivative array equations from which an underlying ODE is constructed. Although the term completion is not used in [50], the authors define “an underlying ODE [as] one whose solutions include those of the DAE” ([50], pg. 21). The derivative array equations, to be discussed further in Sections 2.1 and 2.2, form a $(k+1)$ -block dimensional system consisting of the linear system of DAEs and its 1st through k th derivatives [34]. Once this system of equations is assembled, the approach in [34] solves for the first derivative of the state vector

using a singular Newton method. A noted benefit in [34] of working from the derivative array equations is the derivatives are of the original linear system of DAEs and not of numerically computed values.

Our research focuses on three completions of linear systems of DAEs: the least squares completion (LSC), the stabilized least squares completion (SLSC), and the alternative stabilized completion (ASC). An early article on the least squares completion is [46]. This completion uses a generalized inverse to find the unique minimum norm least squares solution of the derivative array equations. The least squares completion is unique since the solution manifold can be determined from the derivative array equations [46]. A concern of [50] revisited in [138], [139] is that consistent initial conditions may not be enough to keep the numerical solution of the least squares completion from drifting off the solution manifold. The desire is to affect the additional dynamics from completing the manifold so the completion's solutions satisfy the constraints of the linear system of DAEs and converge to the solution manifold if perturbed. The processes proposed in [138] of finding the least squares completion and an alternative least squares completion are each modified in [139] with stabilized differentiation to produce the stabilized least squares completion and the alternative stabilized completion, respectively.

Section 2.1 details the least squares completion; Section 2.2 describes how the process in Section 2.1 is modified to produce the stabilized least squares completion; and Section 2.3 develops the alternative stabilized completion. Each section's completion is constructed for a solvable index k linear system of DAEs with an $n \times 1$ state vector $x(t)$, a square matrix $E(t)$, and smooth coefficient matrices. This chapter concludes with a discussion in Section 2.4 on the stability of these three completions.

2.1 Least Squares Completion

Three steps are required for finding the least squares completion of the linear system of DAEs, equation (2.1). The first step is to construct the derivative array equations

$$\mathcal{E}(t)w(t) + \mathcal{F}(t)x(t) = \mathcal{B}(t)v(t). \quad (2.3)$$

For this completion, the structure of (2.3) consists of

$$\mathcal{E}(t) = \begin{bmatrix} E(t) & 0 & 0 & \dots \\ \dot{E}(t) + F(t) & E(t) & 0 & \ddots \\ \ddot{E}(t) + 2\dot{F}(t) & 2\dot{E}(t) + F(t) & E(t) & \ddots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix}, \quad \mathcal{B}(t) = \begin{bmatrix} B(t) & 0 & 0 & \dots \\ \dot{B}(t) & B(t) & 0 & \ddots \\ \ddot{B}(t) & 2\dot{B}(t) & B(t) & \ddots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix},$$

$$\mathcal{F}(t) = \begin{bmatrix} F(t) \\ \dot{F}(t) \\ \vdots \\ F^{(k)}(t) \end{bmatrix}, \quad w(t) = \begin{bmatrix} \dot{x}(t) \\ \ddot{x}(t) \\ \vdots \\ x^{(k+1)}(t) \end{bmatrix}, \quad v(t) = \begin{bmatrix} u(t) \\ \dot{u}(t) \\ \vdots \\ u^{(k)}(t) \end{bmatrix},$$

where $\mathcal{E}(t)$, $\mathcal{B}(t)$ are $(k+1) \times (k+1)$ block matrices and vectors $v(t)$ in equations (2.2) and (2.3) are equal. The first block row of this system is the linear system of DAEs and the subsequent block rows are the 1st through k th derivatives of equation (2.1), which are found by differentiating with differential polynomial $\frac{d}{dt}$. Having assumed matrices $E(t)$, $F(t)$, and $B(t)$ from the linear system of DAEs are smooth, matrices $\mathcal{E}(t)$, $\mathcal{F}(t)$, and $\mathcal{B}(t)$ are smooth as well. Additionally, $\mathcal{E}(t)$ has constant rank from the solvable system assumption [56].

The second step is to solve for a vector $\bar{w}(t)$ in the least squares sense using $\mathcal{E}(t)^\dagger$, the Moore-Penrose pseudoinverse of $\mathcal{E}(t)$:

$$\bar{w}(t) = \begin{bmatrix} \dot{x}(t) \\ * \end{bmatrix} = -\mathcal{E}(t)^\dagger \mathcal{F}(t)x(t) + \mathcal{E}(t)^\dagger \mathcal{B}(t)v(t). \quad (2.4)$$

Equation (2.4) is a least squares solution [128] of

$$\mathcal{E}(t)\bar{w}(t) = \mathcal{E}(t)\mathcal{E}(t)^\dagger(-\mathcal{F}(t)x(t) + \mathcal{B}(t)v(t)),$$

which is derived from the derivative array equations,

$$\begin{aligned} & \mathcal{E}(t)w(t) + \mathcal{F}(t)x(t) = \mathcal{B}(t)v(t) \\ \implies & \mathcal{E}(t)\mathcal{E}(t)^\dagger\mathcal{E}(t)w(t) = \mathcal{E}(t)\mathcal{E}(t)^\dagger(-\mathcal{F}(t)x(t) + \mathcal{B}(t)v(t)) \\ \implies & \mathcal{E}(t)w(t) = \mathcal{E}(t)\mathcal{E}(t)^\dagger(-\mathcal{F}(t)x(t) + \mathcal{B}(t)v(t)). \end{aligned}$$

The final step is to define the completion's coefficient matrices $\tilde{A}(t)$ and $\tilde{B}(t)$ from equation (2.4). By construction, the first n rows of $\bar{w}(t)$ equal $\dot{x}(t)$ from the linear system of DAEs. Therefore, matrix $\tilde{A}(t)$ is taken as the first n rows of $-\mathcal{E}(t)^\dagger \mathcal{F}(t)$ and matrix $\tilde{B}(t)$ is taken as the first n rows of $\mathcal{E}(t)^\dagger \mathcal{B}(t)$. Both matrices $\tilde{A}(t)$ and $\tilde{B}(t)$ are smooth because they are defined from smooth matrices; $\mathcal{E}(t)^\dagger$ is smooth since $\mathcal{E}(t)$ is smooth and is assumed to have constant rank [77].

2.2 Stabilized Least Squares Completion

The three steps outlined in Section 2.1 also describe the process for finding the stabilized least squares completion of equation (2.1) except for one modification to the first step. For this

completion, the k derivatives in the derivative array equations come from differentiating the linear system of DAEs using differential polynomial $\frac{d}{dt} + \lambda$, resulting in

$$\mathcal{E}(t) = \begin{bmatrix} E(t) & 0 & 0 & \dots \\ \dot{E}(t) + F(t) + \lambda E(t) & E(t) & 0 & \ddots \\ \ddot{E}(t) + 2\dot{F}(t) + 2\lambda\dot{E}(t) + 2\lambda F(t) + \lambda^2 E(t) & 2\dot{E}(t) + F(t) + 2\lambda E(t) & E(t) & \ddots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix},$$

$$\mathcal{B}(t) = \begin{bmatrix} B(t) & 0 & 0 & \dots \\ \dot{B}(t) + \lambda B(t) & B(t) & 0 & \ddots \\ \ddot{B}(t) + 2\lambda\dot{B}(t) + \lambda^2 B(t) & 2\dot{B}(t) + 2\lambda B(t) & B(t) & \ddots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix},$$

$$\mathcal{F}(t) = \begin{bmatrix} F(t) \\ \dot{F}(t) + \lambda F(t) \\ \ddot{F}(t) + 2\lambda\dot{F}(t) + \lambda^2 F(t) \\ \vdots \end{bmatrix}, \quad w(t) = \begin{bmatrix} \dot{x}(t) \\ \ddot{x}(t) \\ \vdots \\ x^{(k+1)}(t) \end{bmatrix}, \quad v(t) = \begin{bmatrix} u(t) \\ \dot{u}(t) \\ \vdots \\ u^{(k)}(t) \end{bmatrix}.$$

$\frac{d}{dt} + \lambda$ represents stabilized differentiation and works to stabilize the additional dynamics of the completion. Stabilization parameter λ is discussed further in Section 2.4.

2.3 Alternative Stabilized Completion

The process for finding the alternative stabilized completion originated with a method from [105] that constructs a canonical form for a linear system of DAEs. Articles [107], [108], and [111] expand on [105], extracting an index 1 linear system of DAEs from a higher index one without altering the system's solutions. This extraction is motivated by the availability of numerical methods for solving index 1 and not higher index linear systems of DAEs. Reference [106] provides a comprehensive overview of the material in [105], [107], [108], [111].

This extraction technique is incorporated by the authors of [138] to generate a system, the alternative least squares completion, for which $\dot{x}(t)$ is defined explicitly. Article [139] extends the linear time-invariant focus of [138] to linear time-varying systems and introduces stabilized differentiation into the process of finding the alternative least squares completion. The resulting alternative stabilized LSC is now referred to as the alternative stabilized completion [23], [24], [25], [26]. Dissertation [137] includes additional details about the methods in [138], [139].

The first step for finding the alternative stabilized completion of an index k linear system of DAEs is to construct equation (2.3), the derivative array equations using differential polynomial $\frac{d}{dt}$. From this point on, the derivations differ for the linear time-invariant and time-varying cases, so their developments are presented separately.

2.3.1 Linear Time-Invariant Case

For a linear time-invariant system of DAEs $E\dot{x}(t) + Fx(t) = Bu(t)$, the coefficient matrices of equation (2.3) simplify to

$$\mathcal{E} = \begin{bmatrix} E & 0 & 0 & \dots \\ F & E & 0 & \ddots \\ 0 & F & E & \ddots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix}, \quad \mathcal{F} = \begin{bmatrix} F \\ 0 \\ 0 \\ \vdots \end{bmatrix}, \quad \mathcal{B} = \begin{bmatrix} B & 0 & 0 & \dots \\ 0 & B & 0 & \ddots \\ 0 & 0 & B & \ddots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix}.$$

Following the construction of the derivative array equations, the first k block rows of matrices \mathcal{E} , \mathcal{F} , and $f_v(t) = \mathcal{B}v(t)$ are denoted by $\tilde{\mathcal{E}}$, $\tilde{\mathcal{F}}$, and $\tilde{f}_v(t)$, respectively.

The process continues with the calculation of three orthonormal bases to define matrices Z_2^T , T_2 , and Z_1^T . Their developments are presented in terms of singular value decompositions to reflect how the bases are calculated in our numerical algorithm. Matrix Z_2^T is determined such that $Z_2^T \tilde{\mathcal{E}} = 0$. Since the columns of Z_2 form an orthonormal basis for $N(\tilde{\mathcal{E}}^T)$, Z_2^T is taken as the transpose of the last $kn - r_1$ columns of U_1 from the singular value decomposition $\tilde{\mathcal{E}} = U_1 \begin{bmatrix} (D_1)_{r_1 \times r_1} & 0 \\ 0 & 0 \end{bmatrix} V_1^T$. In block column notation $Z_2^T = [Z_{2,0}^T \dots Z_{2,k-1}^T]$, where each of the k blocks has n columns. Next matrix T_2 is determined such that $Z_2^T \tilde{\mathcal{F}} T_2 = 0$. The last $n - r_2$ columns of V_2 from the singular value decomposition $Z_2^T \tilde{\mathcal{F}} = U_2 \begin{bmatrix} (D_2)_{r_2 \times r_2} & 0 \\ 0 & 0 \end{bmatrix} V_2^T$ are an orthonormal basis for $N(Z_2^T \tilde{\mathcal{F}})$ and are used to define T_2 . Finally, matrix Z_1 is defined such that its columns form an orthonormal basis for $R(ET_2)$. Matrix Z_1^T is taken as the transpose of the first r_3 columns of U_3 from the singular value decomposition $ET_2 = U_3 \begin{bmatrix} (D_3)_{r_3 \times r_3} & 0 \\ 0 & 0 \end{bmatrix} V_3^T$.

By construction, matrix $\begin{bmatrix} Z_1^T E \\ Z_1^T F \\ Z_{2,0}^T F \end{bmatrix}$ is nonsingular.

Matrices Z_2^T and Z_1^T are used to build the $(k+1)n \times (k+1)n$ matrix

$$\Gamma = \begin{bmatrix} Z_1^T & 0_{\kappa \times kn} \\ Z_2^T & 0_{d \times n} \\ 0_{d \times n} & Z_2^T \\ \hline & Z_3^T \end{bmatrix},$$

where d is the rank of Z_2^T , $\kappa = n - d$, and Z_3^T is a $kn \times (k + 1)n$ matrix selected so that Γ is nonsingular. The block structure

$$\Gamma = \begin{bmatrix} \begin{bmatrix} Z_1^T \\ Z_{2,0}^T \end{bmatrix} & \begin{bmatrix} 0_{\kappa \times n} \\ Z_{2,1}^T \end{bmatrix} & \dots & \begin{bmatrix} 0_{\kappa \times n} \\ Z_{2,k-1}^T \end{bmatrix} & \begin{bmatrix} 0_{n \times n} \\ 0_{n \times n} \end{bmatrix} \\ \begin{bmatrix} 0_{d \times n} \\ Z_{3,0}^T(1 : \kappa) \end{bmatrix} & \begin{bmatrix} Z_{2,0}^T \\ Z_{3,1}^T(1 : \kappa) \end{bmatrix} & \dots & \begin{bmatrix} Z_{2,k-2}^T \\ Z_{3,k-1}^T(1 : \kappa) \end{bmatrix} & \begin{bmatrix} Z_{2,k-1}^T \\ Z_{3,k}^T(1 : \kappa) \end{bmatrix} \\ * & * & \dots & * & * \\ \vdots & \vdots & \dots & \vdots & \vdots \end{bmatrix}$$

is helpful in the following development, and $*$ is used to designate a term irrelevant to the calculation of the alternative stabilized completion. Multiplying $\mathcal{E}w(t) + \mathcal{F}x(t) = f_v(t)$ on the left by Γ produces $\Gamma\mathcal{E} =$

$$\begin{bmatrix} \begin{bmatrix} Z_1^T \\ Z_{2,0}^T \end{bmatrix} E + \begin{bmatrix} 0_{\kappa \times n} \\ Z_{2,1}^T \end{bmatrix} F & \begin{bmatrix} 0_{\kappa \times n} \\ Z_{2,1}^T \end{bmatrix} E + \begin{bmatrix} 0_{\kappa \times n} \\ Z_{2,2}^T \end{bmatrix} F & \dots & \begin{bmatrix} 0_{\kappa \times n} \\ Z_{2,k-1}^T \end{bmatrix} E + \begin{bmatrix} 0_{n \times n} \\ 0_{n \times n} \end{bmatrix} F & \begin{bmatrix} 0_{n \times n} \\ 0_{n \times n} \end{bmatrix} E \\ \begin{bmatrix} 0_{d \times n} \\ * \end{bmatrix} E + \begin{bmatrix} Z_{2,0}^T \\ * \end{bmatrix} F & \begin{bmatrix} Z_{2,0}^T \\ * \end{bmatrix} E + \begin{bmatrix} Z_{2,1}^T \\ * \end{bmatrix} F & \dots & \begin{bmatrix} Z_{2,k-2}^T \\ * \end{bmatrix} E + \begin{bmatrix} Z_{2,k-1}^T \\ * \end{bmatrix} F & \begin{bmatrix} Z_{2,k-1}^T \\ * \end{bmatrix} E \\ * & * & \dots & * & * \\ \vdots & \vdots & \dots & \vdots & \vdots \end{bmatrix},$$

which simplifies to

$$\Gamma\mathcal{E} = \begin{bmatrix} Z_1^T E & 0_{\kappa \times n} & \dots & 0_{\kappa \times n} & 0_{\kappa \times n} \\ 0_{d \times n} & 0_{d \times n} & \dots & 0_{d \times n} & 0_{d \times n} \\ Z_{2,0}^T F & 0_{d \times n} & \dots & 0_{d \times n} & 0_{d \times n} \\ * & * & \dots & * & * \end{bmatrix};$$

$$\Gamma\mathcal{F} = \begin{bmatrix} \begin{bmatrix} Z_1^T \\ Z_{2,0}^T \end{bmatrix} F \\ \begin{bmatrix} 0_{d \times n} \\ * \end{bmatrix} F \\ * \\ \vdots \end{bmatrix} = \begin{bmatrix} Z_1^T F \\ Z_{2,0}^T F \\ 0_{d \times n} \\ * \end{bmatrix};$$

and for $f(t) = Bu(t)$, $\Gamma f_v(t) =$

$$\begin{bmatrix} \begin{bmatrix} Z_1^T \\ Z_{2,0}^T \end{bmatrix} f(t) + \begin{bmatrix} 0_{\kappa \times n} \\ Z_{2,1}^T \end{bmatrix} \dot{f}(t) + \dots + \begin{bmatrix} 0_{\kappa \times n} \\ Z_{2,k-1}^T \end{bmatrix} f^{(k-1)}(t) + \begin{bmatrix} 0_{n \times n} \\ Z_{2,k-1}^T \end{bmatrix} f^{(k)}(t) \\ \begin{bmatrix} 0_{d \times n} \\ * \end{bmatrix} f(t) + \begin{bmatrix} Z_{2,0}^T \\ * \end{bmatrix} \dot{f}(t) + \dots + \begin{bmatrix} Z_{2,k-2}^T \\ * \end{bmatrix} f^{(k-1)}(t) + \begin{bmatrix} Z_{2,k-1}^T \\ * \end{bmatrix} f^{(k)}(t) \\ \vdots \end{bmatrix},$$

which simplifies to

$$\Gamma f_v(t) = \begin{bmatrix} Z_1^T f(t) \\ Z_{2,0}^T f(t) + Z_{2,1}^T \dot{f}(t) + \dots + Z_{2,k-1}^T f^{(k-1)}(t) \\ Z_{2,0}^T \dot{f}(t) + \dots + Z_{2,k-2}^T f^{(k-1)}(t) + Z_{2,k-1}^T f^{(k)}(t) \\ * \end{bmatrix} = \begin{bmatrix} Z_1^T f(t) \\ Z_2^T \tilde{f}_v(t) \\ Z_2^T (\tilde{f}_v(t))' \\ * \end{bmatrix}.$$

The augmented form of $\Gamma \mathcal{E}w(t) = -\Gamma \mathcal{F}x(t) + \Gamma f_v(t)$ is

$$[\Gamma \mathcal{E} \| -\Gamma \mathcal{F} | \Gamma f_v(t)] = \left[\begin{array}{c|ccc|c} Z_1^T E & 0 & \dots & 0 & -Z_1^T F & Z_1^T f(t) \\ 0 & 0 & \dots & 0 & -Z_{2,0}^T F & Z_2^T \tilde{f}_v(t) \\ Z_{2,0}^T F & 0 & \dots & 0 & 0 & Z_2^T (\tilde{f}_v(t))' \\ * & * & \dots & * & * & * \end{array} \right]. \quad (2.5)$$

The third row in (2.5) equals the first derivative of the second row:

$$\frac{d}{dt} (0 = -Z_{2,0}^T Fx(t) + Z_2^T \tilde{f}_v(t)) \implies Z_{2,0}^T F \dot{x}(t) = Z_2^T (\tilde{f}_v(t))'.$$

If stabilized differentiation is applied to the second row of (2.5) instead,

$$\begin{aligned} & \left(\frac{d}{dt} + \lambda \right) (0 = -Z_{2,0}^T Fx(t) + Z_2^T \tilde{f}_v(t)) \\ \implies & Z_{2,0}^T F \dot{x}(t) = -\lambda Z_{2,0}^T Fx(t) + Z_2^T (\tilde{f}_v(t))' + \lambda Z_2^T \tilde{f}_v(t). \end{aligned}$$

The augmented form then becomes

$$\left[\begin{array}{c|cccc} Z_1^T E & 0 & \cdots & 0 & -Z_1^T F \\ 0 & 0 & \cdots & 0 & -Z_{2,0}^T F \\ Z_{2,0}^T F & 0 & \cdots & 0 & -\lambda Z_{2,0}^T F \\ \hline * & * & \cdots & * & * \end{array} \right] \left[\begin{array}{c} Z_1^T f(t) \\ Z_2^T \tilde{f}_v(t) \\ Z_2^T (\tilde{f}_v(t))' + \lambda Z_2^T \tilde{f}_v(t) \\ * \end{array} \right]. \quad (2.6)$$

The first and third rows of (2.6) can be considered together to form the system

$$\left[\begin{array}{c} Z_1^T E \\ Z_{2,0}^T F \end{array} \right] \dot{\tilde{x}}(t) = - \left[\begin{array}{c} Z_1^T F \\ \lambda Z_{2,0}^T F \end{array} \right] \tilde{x}(t) + \left[\begin{array}{c} Z_1^T f(t) \\ Z_2^T (\tilde{f}_v(t))' + \lambda Z_2^T \tilde{f}_v(t) \end{array} \right], \quad (2.7)$$

from which $\dot{\tilde{x}}(t)$ can be solved for explicitly,

$$\dot{\tilde{x}}(t) = - \left[\begin{array}{c} Z_1^T E \\ Z_{2,0}^T F \end{array} \right]^{-1} \left[\begin{array}{c} Z_1^T F \\ \lambda Z_{2,0}^T F \end{array} \right] \tilde{x}(t) + \left[\begin{array}{c} Z_1^T E \\ Z_{2,0}^T F \end{array} \right]^{-1} \left[\begin{array}{c} Z_1^T f(t) \\ Z_2^T (\tilde{f}_v(t))' + \lambda Z_2^T \tilde{f}_v(t) \end{array} \right], \quad (2.8)$$

since its coefficient matrix in equation (2.7) is nonsingular. Equation (2.8) is the alternative stabilized completion. If \tilde{A} is defined as

$$-\left[\begin{array}{c} Z_1^T E \\ Z_{2,0}^T F \end{array} \right]^{-1} \left[\begin{array}{c} Z_1^T F \\ \lambda Z_{2,0}^T F \end{array} \right]$$

and if \tilde{B} is defined as

$$\left[\begin{array}{c} Z_1^T E \\ Z_{2,0}^T F \end{array} \right]^{-1} \left[\begin{array}{ccccc} Z_1^T B & 0 & \cdots & 0 & 0 \\ \lambda Z_{2,0}^T B & (Z_{2,0}^T + \lambda Z_{2,1}^T) B & \cdots & (Z_{2,k-2}^T + \lambda Z_{2,k-1}^T) B & Z_{2,k-1}^T B \end{array} \right],$$

then equation (2.8) can be written as $\dot{\tilde{x}}(t) = \tilde{A}\tilde{x}(t) + \tilde{B}v(t)$.

The derivation of the alternative stabilized completion includes more steps than are required in a numerical algorithm. As with the completions described in Sections 2.1 and 2.2, the process for finding the alternative stabilized completion can be grouped into three steps:

STEP 1: Construct the derivative array equations with differential polynomial $\frac{d}{dt}$ and denote the first k block rows of \mathcal{E} and \mathcal{F} as $\tilde{\mathcal{E}}$ and $\tilde{\mathcal{F}}$, respectively.

STEP 2: Define matrices $Z_2^T = [Z_{2,0}^T \dots Z_{2,k-1}^T]$ (n columns per block), T_2 , and Z_1^T so the columns of Z_2 are an orthonormal basis for $N(\tilde{\mathcal{E}}^T)$, the columns of T_2 are an orthonormal basis for $N(Z_2^T \tilde{\mathcal{F}})$, and the columns of Z_1 are an orthonormal basis for $R(ET_2)$.

STEP 3: Calculate the completion's coefficient matrices \tilde{A} and \tilde{B} :

$$-\begin{bmatrix} Z_1^T E \\ Z_{2,0}^T F \end{bmatrix}^{-1} \begin{bmatrix} Z_1^T F \\ \lambda Z_{2,0}^T F \end{bmatrix},$$

$$\begin{bmatrix} Z_1^T E \\ Z_{2,0}^T F \end{bmatrix}^{-1} \begin{bmatrix} Z_1^T B & 0 & \dots & 0 & 0 \\ \lambda Z_{2,0}^T B & (Z_{2,0}^T + \lambda Z_{2,1}^T) B & \dots & (Z_{2,k-2}^T + \lambda Z_{2,k-1}^T) B & Z_{2,k-1}^T B \end{bmatrix}.$$

2.3.2 Linear Time-Varying Case

In Subsection 2.3.1, the alternative stabilized completion was derived from a subsystem of the derivative array equations that did not include the k th derivative of the linear system of DAEs. The information provided by an additional differentiation is superfluous in the linear time-invariant case but necessary in the linear time-varying case.

Following the construction of the derivative array equations, a matrix $Z_2(t)^T$ is defined such that the columns of $Z_2(t)$ form an orthonormal basis for $N(\mathcal{E}(t)^T)$. Multiplying equation (2.3) on the left by $Z_2(t)^T$ and then moving all remaining terms to the right-hand side results in

$$0 = -Z_2(t)^T \mathcal{F}(t)x(t) + Z_2(t)^T \mathcal{B}(t)v(t). \quad (2.9)$$

Applying stabilized differentiation once to equation (2.9) produces

$$\begin{aligned} 0 &= \left(\frac{d}{dt} + \lambda \right) (-Z_2(t)^T \mathcal{F}(t)x(t) + Z_2(t)^T \mathcal{B}(t)v(t)) \\ 0 &= -\dot{Z}_2(t)^T \mathcal{F}(t)x(t) - Z_2(t)^T \dot{\mathcal{F}}(t)x(t) - Z_2(t)^T \mathcal{F}(t)\dot{x}(t) - \lambda Z_2(t)^T \mathcal{F}(t)x(t) \\ &\quad + (Z_2(t)^T \mathcal{B}(t)v(t))' + \lambda Z_2(t)^T \mathcal{B}(t)v(t) \end{aligned}$$

from which

$$\begin{aligned} Z_2(t)^T \mathcal{F}(t)\dot{x}(t) &= -\left((Z_2(t)^T \mathcal{F}(t))' + \lambda Z_2(t)^T \mathcal{F}(t) \right) x(t) \\ &\quad + \left((Z_2(t)^T \mathcal{B}(t)v(t))' + \lambda Z_2(t)^T \mathcal{B}(t)v(t) \right). \end{aligned} \quad (2.10)$$

Next matrices $T_2(t)$ and $Z_{1,0}(t)^T$ are determined so the columns of $T_2(t)$ are an orthonormal basis for $N(Z_2(t)^T \mathcal{F}(t))$ and the columns of $Z_{1,0}(t)$ form an orthonormal basis for $R(E(t)T_2(t))$.

By construction, matrix $\begin{bmatrix} Z_{1,0}(t)^T E(t) \\ Z_2(t)^T \mathcal{F}(t) \end{bmatrix}$ is nonsingular. Therefore,

$$Z_{1,0}(t)^T E(t)\dot{x}(t) = -Z_{1,0}(t)^T F(t)x(t) + Z_{1,0}(t)^T B(t)u(t)$$

considered together with equation (2.10) produces system

$$\begin{bmatrix} Z_{1,0}(t)^T E(t) \\ Z_2(t)^T \mathcal{F}(t) \end{bmatrix} \dot{\tilde{x}}(t) = - \begin{bmatrix} Z_{1,0}(t)^T F(t) \\ (Z_2(t)^T \mathcal{F}(t))' + \lambda Z_2(t)^T \mathcal{F}(t) \end{bmatrix} \tilde{x}(t) \\ + \begin{bmatrix} Z_{1,0}(t)^T B(t)u(t) \\ (Z_2(t)^T \mathcal{B}(t)v(t))' + \lambda Z_2(t)^T \mathcal{B}(t)v(t) \end{bmatrix}$$

whence $\dot{\tilde{x}}(t)$ can be solved for explicitly. The alternative stabilized completion is

$$\begin{aligned} \dot{\tilde{x}}(t) &= - \begin{bmatrix} Z_{1,0}(t)^T E(t) \\ Z_2(t)^T \mathcal{F}(t) \end{bmatrix}^{-1} \begin{bmatrix} Z_{1,0}(t)^T F(t) \\ (Z_2(t)^T \mathcal{F}(t))' + \lambda Z_2(t)^T \mathcal{F}(t) \end{bmatrix} \tilde{x}(t) \\ &+ \begin{bmatrix} Z_{1,0}(t)^T E(t) \\ Z_2(t)^T \mathcal{F}(t) \end{bmatrix}^{-1} \begin{bmatrix} Z_{1,0}(t)^T B(t)u(t) \\ (Z_2(t)^T \mathcal{B}(t)v(t))' + \lambda Z_2(t)^T \mathcal{B}(t)v(t) \end{bmatrix}. \end{aligned} \quad (2.11)$$

The structure of equation (2.11) is reminiscent of $\dot{\tilde{x}}(t) = \tilde{A}(t)\tilde{x}(t) + \tilde{B}(t)v(t)$, but the first derivative of vector $v(t)$ keeps a matrix $\tilde{B}(t)$ from being defined independent of the input $u(t)$ and its $k+1$ derivatives.

The process of finding the alternative stabilized completion requires the computation of three orthonormal bases to define matrices $Z_2(t)^T$, $T_2(t)$, and $Z_{1,0}(t)^T$. In a numerical algorithm, a singular value decomposition could be executed at each time t if the bases did not need to be smooth. However, the first derivative of $Z_2(t)^T$ is required to calculate the alternative stabilized completion. A method presented in [109] and [147] finds a smooth decomposition of a linear time-varying matrix by performing an orthogonal transformation on a non-smooth decomposition. Article [109] also describes a process for determining the first derivatives of the smooth decomposition factors. For convenience, Appendix A includes steps detailing the smooth decomposition and the first derivative definitions.

2.4 Completion Stability

The motivation behind stabilizing the least squares completion and developing the alternative stabilized completion was to improve the stability of the additional dynamics so a completion's numerical solution would converge to the solution manifold if a perturbation caused drift. The implementation in [139] of stabilized differentiation with differential polynomial $\frac{d}{dt} + \lambda$ during completion development was influenced by the work in [14]. Stabilization parameter λ is generally defined as a constant scalar. The effects from selecting such a λ are known for linear time-invariant completions due to the relationship between the eigenvalues and the stability of a linear time-invariant system of ODEs [4].

By design, the matrix pencil eigenvalues of a linear time-invariant system of DAEs are also eigenvalues of matrix \tilde{A} . A completion's other eigenvalues come from its additional dynamics and are referred to as the additional dynamics eigenvalues in this dissertation. If drift occurs but the additional dynamics eigenvalues have a negative real part, the completion's numerical solution will converge to the solution manifold due to the asymptotically stable additional dynamics. For a constant scalar λ , the additional dynamics eigenvalues equal $-\lambda$. Therefore, the desired negative real part results when $\lambda > 0$. [137], [139]

If $\lambda = 0$, the stabilized least squares completion reduces to the least squares completion and its additional dynamics eigenvalues equal 0. The Jordan canonical form of \tilde{A} from the least squares completion reveals the 0 eigenvalues are contained in a $k \times k$ Jordan block. For an index 1 linear time-invariant system of DAEs, the additional dynamics of its least squares completion are stable. Thus, a perturbation's drift will not grow but the completion's numerical solution will remain off the solution manifold. For a higher index system, if a perturbation occurs, the resulting drift will grow as linear combinations formed with terms t^j , $0 \leq j \leq k - 1$, from the additional dynamics affect the completion's numerical solution. Matrix \tilde{A} from the alternative least squares completion (the alternative stabilized completion when $\lambda = 0$) has a different Jordan canonical form. Instead of a $k \times k$ Jordan block, there is a 1×1 Jordan block for each 0 eigenvalue. No matter the index, the additional dynamics are stable since the only polynomial terms that appear in the linear combinations are constants. When $\lambda > 0$, the linear combinations formed with terms $t^j e^{-\lambda t}$ for the stabilized least squares completion and with terms $e^{-\lambda t}$ for the alternative stabilized completion are asymptotically stable. [137], [138], [139]

A completion's additional dynamics are not well-understood in the time-varying case. Thus, the linear time-invariant system theory acts as guidelines when selecting a large enough constant scalar $\lambda > 0$ to find a stabilized completion of a linear time-varying system of DAEs [137], [139]. Theory is discussed in [137] for linear time-varying stabilization parameters and remains to be formalized for stabilizing with a matrix instead of a scalar [23].

When reading [25], the $\lambda (Z_2^T \mathcal{F}x - Z_2^T \mathcal{B}v)$ term in equation (7b) should be subtracted and not added since λ is defined ≥ 0 .

Chapter 3

Observers

Completions allow us to construct observers for systems of DAEs using design techniques associated with systems of ODEs. The development of the observers being considered assumes a completion of a system of DAEs with output equation has already been found:

$$\dot{\tilde{x}}(t) = \tilde{A}(t)\tilde{x}(t) + \tilde{B}(t)v(t) \quad (3.1a)$$

$$\tilde{y}(t) = C(t)\tilde{x}(t). \quad (3.1b)$$

This assumption is necessary in the linear time-varying case to reduce the number of times matrices we construct are differentiated. Additionally, $\tilde{A}(t)$, $\tilde{B}(t)$, and $C(t)$ should be smooth matrix functions and are assumed bounded. Unless otherwise noted, $\tilde{x}(t)$ is $n \times 1$, $\tilde{y}(t)$ is $m \times 1$, and $C(t)$ is full row rank.

Section 3.1 describes observability checks for linear systems of ODEs. Sections 3.2 and 3.3 review derivations of a full-order observer and a reduced-order observer, respectively; Section 3.4 presents our observer, the maximally reduced observer; and Section 3.5 introduces the DAE manifold observer for linear time-invariant systems of DAEs. Section 3.6 concludes this chapter by reconsidering completion stability with observability in mind.

3.1 Observable and Detectable

Not every system of ordinary differential equations can be observed. In order for the system to be observable, the information provided through the output needs to make the state distinguishable from the zero solution.

For the linear time-invariant system $\dot{\tilde{x}}(t) = \tilde{A}\tilde{x}(t) + \tilde{B}v(t)$ with output equation $\tilde{y}(t) = C\tilde{x}(t)$, this relationship between the output and the state can be verified by calculating the rank of the observability matrix

$$\mathcal{O} = \begin{bmatrix} C \\ C\tilde{A} \\ \vdots \\ C\tilde{A}^{n-1} \end{bmatrix}.$$

The entries of \mathcal{O} are the coefficient matrices of $\tilde{x}(t)$ from $\tilde{y}(t)$ and its first $n - 1$ derivatives. If matrix \mathcal{O} is full column rank, then the system, as well as the pair (\tilde{A}, C) , is observable and an observer can be designed to asymptotically estimate the state.

If the rank of \mathcal{O} is $n_o < n$, the possibility of observing the linear time-invariant system becomes dependent on the eigenvalues of \tilde{A} . The similarity transformation matrix $Q = [Q_{n_o} \ v_1 \ \dots \ v_{n-n_o}]$ is assembled from the basis vectors of $N(\mathcal{O})$ ($\{v_1, \dots, v_{n-n_o}\}$) and from n_o vectors chosen to make Q nonsingular (Q_{n_o}). Matrices $\mathcal{A}_u = Q^{-1}\tilde{A}Q = \begin{bmatrix} A_1 & 0 \\ A_2 & A_3 \end{bmatrix}$ and $\mathcal{C}_u = CQ = \begin{bmatrix} C_1 & 0 \end{bmatrix}$ reveal the standard form for the unobservable system. The eigenvalues of block A_1 are observable, while the eigenvalues of block A_3 are unobservable. Although the unobservable eigenvalues cannot be affected to make the observer converge to the true state, if all eigenvalues of block A_3 have a negative real part, then the observer can still be designed to asymptotically estimate the state. These systems for which (A_1, C_1) is the observable pair and the unobservable eigenvalues are stable (negative real part) are referred to as detectable. If C_1 is not full row rank, it should be redefined with only its linearly independent rows. [4]

It is possible to check the observability of a linear time-varying system of ODEs using the observability matrix [154]. However, for system (3.1) the $n - 1$ derivatives of $\tilde{y}(t)$ needed to construct $\mathcal{O}(t)$ require knowing the first $n - 2$ derivatives of $\tilde{A}(t)$ and the first $n - 1$ derivatives of $C(t)$. Unless matrix $\tilde{A}(t)$ can be defined symbolically in terms of quantities with known derivatives, differentiating a computed matrix may introduce numerical error into the observability matrix and possibly produce an incorrect rank. Derivatives of $\tilde{A}(t)$ can be computed using larger derivative arrays, but that approach is not presently considered.

One alternative method for checking the observability of a linear time-varying system of ODEs solves a Lyapunov differential equation

$$\frac{\partial W_o(t)}{\partial t} = -\tilde{A}(t)^T W_o(t) - W_o(t)\tilde{A}(t) - C(t)^T C(t) \quad (3.2)$$

by integrating backwards in time with $W_o(t_f, t_f) = 0$ [60]. The observability Gramian $W_o(t_0, t_f)$ is defined

$$W_o(t_0, t_f) = \int_{t_0}^{t_f} \Phi(\tau, t_0)^T C(\tau)^T C(\tau) \Phi(\tau, t_0) d\tau, \quad (3.3)$$

where $\Phi(t, t_0)$ is the state transition matrix ($\dot{\Phi}(t, t_0) = \tilde{A}(t)\Phi(t, t_0)$ and $\Phi(t_0, t_0) = \Phi(t_0) = I$). The method implemented for the linear time-varying example in Chapter 6 integrates forward in time with $W_o(t_0, t_0)$ and solves the system

$$\frac{d}{dt}W_o(t) = \Phi(t)^T C^T(t)C(t)\Phi(t) \quad (3.4a)$$

$$\frac{d}{dt}\Phi(t) = \tilde{A}(t)\Phi(t). \quad (3.4b)$$

If there exists a time t_f such that $\text{rank}(W_o(t_0, t_f)) = n$ for $t_0 < t_f < \infty$, then system (3.1) is observable and $(\tilde{A}(t), C(t))$ is its observable pair [4]. If system (3.1) is not observable, a similarity transformation can also be applied in the linear time-varying case to identify the standard form for the unobservable system. The transformation matrix should be smooth and bounded and have a bounded inverse [160]. As long as these conditions are met, the structure of $Q(t)$ can remain the same as Q without having found the observability matrix since $N(\mathcal{O}) = N(W_o(t_0, t_f))$ [4]. From [159], [160], if the unobservable system $\dot{x}_3(t) = A_3(t)x_3(t)$ is uniformly asymptotically stable, then system (3.1) is detectable. For our numerical algorithm, a system is detectable if the norm of its unobservable system's state transition matrix goes to zero as time goes to infinity. An alternative definition for detectable is given in Subsection 3.2.2.

3.2 The Full-Order Observer

The full-order observer

$$\dot{\tilde{x}}(t) = \tilde{A}(t)\tilde{x}(t) + \tilde{B}(t)v(t) + L(t)(\tilde{y}(t) - \hat{y}(t)) \quad (3.5a)$$

$$\hat{y}(t) = C(t)\tilde{x}(t) \quad (3.5b)$$

is an example of a Luenberger observer because of its use of known inputs and outputs from the physical system to be observed and because of its assignment of observable eigenvalues to the left half-plane in the linear time-invariant case [121], [122]. Observer (3.5) uses a correction term to compare the plant output $\tilde{y}(t)$ with the observer output $\hat{y}(t)$ (the output resulting from the observer calculated state). If data exists for the plant output, then these results from the physical system can be substituted into (3.5a) for $\tilde{y}(t)$.

A measure of the observer's state estimate is $e(t) = \tilde{x}(t) - \hat{x}(t)$, the difference between the solution of the completion and its estimated value. The first derivative of estimation error $e(t)$,

$$\begin{aligned} \dot{e}(t) &= \dot{\tilde{x}}(t) - \dot{\hat{x}}(t) \\ &= \tilde{A}(t)\tilde{x}(t) + \tilde{B}(t)v(t) - \left(\tilde{A}(t)\hat{x}(t) + \tilde{B}(t)v(t) + L(t)(\tilde{y}(t) - \hat{y}(t)) \right) \\ &= \tilde{A}(t)(\tilde{x}(t) - \hat{x}(t)) - L(t)C(t)(\tilde{x}(t) - \hat{x}(t)) \\ &= (\tilde{A}(t) - L(t)C(t))e(t), \end{aligned} \quad (3.6)$$

indicates the choice of gain matrix $L(t)$ influences whether or not $e(t) \rightarrow 0$ as $t \rightarrow \infty$. The methods chosen for constructing the gain matrix differ for the linear time-invariant and time-varying cases, so their developments are presented separately.

3.2.1 Linear Time-Invariant Case

In the linear time-invariant case, $e(t) \rightarrow 0$ as $t \rightarrow \infty$ if the eigenvalues of $\tilde{A} - LC$ have negative real parts. Gain matrix L is designed to assign the observable eigenvalues of \tilde{A} to the left half-plane. The algorithm implemented for constructing gain matrix L comes from [4] and utilizes the controller form of the linear time-invariant system's observable pair.

The gain matrix algorithm receives three inputs. If system (3.1) is observable, then \tilde{A} and C are two of the three inputs. If the system is detectable, A_1 and C_1 are input instead. The third input is an appropriately sized, user-defined matrix D with eigenvalues equaling the ones to which the user wants to assign the observable eigenvalues. One way to guarantee matrix D has the desired eigenvalues for use in controller form is to define it as a diagonal matrix so its entries are the eigenvalues.

The algorithm begins by finding the dual (A_D, B_D) of the input observable pair ($A_D = \tilde{A}^T$ and $B_D = C^T$, for example). Recall, A_D is an $n_o \times n_o$ matrix where n_o is the rank of observability matrix \mathcal{O} . Working with the controller form, the goal becomes constructing a matrix F_D so that $e_c(t)$ from $\dot{e}_c(t) = (A_D + B_D F_D) e_c(t) = D e_c(t)$ goes to zero as time goes to infinity.

If B_D has a single column ($m = 1$), the algorithm assembles the controllability matrix $\mathcal{C} = \begin{bmatrix} B_D & A_D B_D & \dots & A_D^{n_o-1} B_D \end{bmatrix}$, calculates its inverse, and then defines vector q as the n_o th row of \mathcal{C}^{-1} . After calculating the similarity transformation matrix

$$P = \begin{bmatrix} q^T & (q A_D)^T & \dots & (q A_D^{n_o-1})^T \end{bmatrix}^T,$$

the transformation of $D = A_D + B_D F_D$ to $PDP^{-1} = PA_D P^{-1} + PB_D F_D P^{-1}$ reveals the controller form of (A_D, B_D) ,

$$A_C = PA_D P^{-1} = \begin{bmatrix} 0 & 1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 1 \\ -\alpha_0 & -\alpha_1 & \dots & -\alpha_{n_o-1} \end{bmatrix}, \quad B_C = PB_D = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix},$$

and the characteristic polynomial of A_C (and of A_D),

$$\alpha(s) = \det(sI - A_C) = s^{n_o} + \alpha_{n_o-1}s^{n_o-1} + \dots + \alpha_1s + \alpha_0.$$

The eigenvalues of a matrix are the roots of its characteristic polynomial. Since $A_F = PDP^{-1}$ is also in companion form, its final row provides the coefficients of the characteristic

polynomial $\beta(s) = \det(sI - A_F) = s^{n_o} + \beta_{n_o-1}s^{n_o-1} + \dots + \beta_1s + \beta_0$ with roots equal to the desired eigenvalues. Additionally, $A_F = A_C + B_C F_C$,

$$\begin{bmatrix} 0 & 1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 1 \\ -\beta_0 & -\beta_1 & \dots & -\beta_{n_o-1} \end{bmatrix} = \begin{bmatrix} 0 & 1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 1 \\ -\alpha_0 & -\alpha_1 & \dots & -\alpha_{n_o-1} \end{bmatrix} + \begin{bmatrix} 0 & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 0 \\ f_0 & f_1 & \dots & f_{n_o-1} \end{bmatrix},$$

reveals the components of vector $F_C = F_D P^{-1} = [f_0 \dots f_{n_o-1}]$ equal the difference between the actual and the desired characteristic polynomial coefficients. After the algorithm calculates $f_i = \alpha_i - \beta_i$ for $i = 0, \dots, n_o - 1$, gain matrix L is defined in terms of the dual matrix $F_D = F_C P$. If system (3.1) is observable, then $L = -F_D^T$. If the system is detectable, $L_d = -F_D^T$ is the gain matrix for $A_1 - L_d C_1$. The additional calculation $L = Q \begin{bmatrix} L_d & 0 \\ 0 & 0 \end{bmatrix}$ is required for use in the observer, where Q is the transformation matrix defined in Section 3.1.

When $m \geq 2$, the process for building the similarity transformation matrix P requires some additional steps. After the gain matrix algorithm assembles the controllability matrix

$$\begin{aligned} \mathcal{C} &= \left[B_D \quad A_D B_D \quad \dots \quad A_D^{n_o-1} B_D \right] \\ &= \left[b_1 \quad b_2 \quad \dots \quad b_m \mid Ab_1 \quad Ab_2 \quad \dots \quad Ab_m \mid \dots \mid A^{n_o-1} b_1 \quad A^{n_o-1} b_2 \quad \dots \quad A^{n_o-1} b_m \right], \end{aligned}$$

it identifies the first n_o linearly independent columns of \mathcal{C} (moving from left to right). These n_o linearly independent columns are saved in

$$\bar{\mathcal{C}} = \left[b_1 \quad Ab_1 \quad \dots \quad A^{\mu_1-1} b_1 \mid b_2 \quad Ab_2 \quad \dots \quad A^{\mu_2-1} b_2 \mid \dots \mid b_m \quad Ab_m \quad \dots \quad A^{\mu_m-1} b_m \right].$$

Using a counter, the algorithm keeps track of the controllability indices μ_i , the number of linearly independent columns associated with each b_i , $i = 1, \dots, m$. As a check, $\sum_{i=1}^m \mu_i = n_o$. After calculating $(\bar{\mathcal{C}})^{-1}$ and defining q_k as the σ_k th row of $(\bar{\mathcal{C}})^{-1}$, where $\sigma_k = \sum_{i=1}^k \mu_i$ for $k = 1, \dots, m$, these q_k vectors are used to calculate the similarity transformation matrix

$$P = \left[q_1^T \quad (q_1 A_D)^T \quad \dots \quad (q_1 A_D^{\mu_1-1})^T \mid \dots \mid q_m^T \quad (q_m A_D)^T \quad \dots \quad (q_m A_D^{\mu_m-1})^T \right]^T.$$

The transformation $PDP^{-1} = PA_D P^{-1} + PB_D F_D P^{-1}$ reveals the controller form of (A_D, B_D) ,

$$A_{\mathcal{C}} = PA_D P^{-1} = \begin{bmatrix} A_{ij} \end{bmatrix}, \quad B_{\mathcal{C}} = PB_D = \begin{bmatrix} B_1 \\ B_2 \\ \vdots \\ B_m \end{bmatrix},$$

where

$$A_{ii} = \begin{bmatrix} 0 & 1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 1 \\ * & * & \dots & * \end{bmatrix}, \quad A_{ij} = \begin{bmatrix} 0 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & 0 \\ * & \dots & * \end{bmatrix}, \quad B_i = \begin{bmatrix} 0 & \dots & 0 & 0 & 0 & \dots & 0 \\ \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & \dots & 0 & 1 & * & \dots & * \end{bmatrix}.$$

Submatrix A_{ii} is $\mu_i \times \mu_i$, A_{ij} is $\mu_i \times \mu_j$, and B_i is $\mu_i \times m$, where $*$ is a placeholder for example specific entries. The i th column of B_i is a unit vector with a 1 in the μ_i th row.

The gain matrix algorithm constructs

$$W_F = \begin{bmatrix} 0 & 1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 1 \\ -\omega_0 & -\omega_1 & \dots & -\omega_{n_o-1} \end{bmatrix},$$

a matrix in companion form with ω_i entries equal to the coefficients of matrix D 's characteristic polynomial. After the algorithm picks out the σ_k rows of $A_{\mathcal{C}}$, $B_{\mathcal{C}}$, and W_F ($A_{\mathcal{C}}(\sigma_k, :)$ is the σ_k th row of $A_{\mathcal{C}}$, for example), it assembles

$$A_{\sigma} = \begin{bmatrix} A_{\mathcal{C}}(\sigma_1, :) \\ A_{\mathcal{C}}(\sigma_2, :) \\ \vdots \\ A_{\mathcal{C}}(\sigma_m, :) \end{bmatrix}, \quad B_{\sigma} = \begin{bmatrix} B_{\mathcal{C}}(\sigma_1, :) \\ B_{\mathcal{C}}(\sigma_2, :) \\ \vdots \\ B_{\mathcal{C}}(\sigma_m, :) \end{bmatrix}, \quad W_{\sigma} = \begin{bmatrix} W_F(\sigma_1, :) \\ W_F(\sigma_2, :) \\ \vdots \\ W_F(\sigma_m, :) \end{bmatrix}$$

and calculates

$$F_{\mathcal{C}} = F_D P^{-1} = B_{D_{\sigma}}^{-1} (A_{D_{\sigma}} - W_{D_{\sigma}}).$$

Then gain matrix L is defined in terms of $F_D = F_{\mathcal{C}} P$ the same way it was when $m = 1$.

3.2.2 Linear Time-Varying Case

The method chosen for constructing the gain matrix in the linear time-varying case comes from the development presented in [98], [135], [136]. From the theory of Lyapunov, if $V_L(t) =$

$e(t)^T P_L(t) e(t)$, if $-\dot{V}_L(t) = e(t)^T Q_L(t) e(t)$, and if $(\tilde{A}(t), C(t))$ is uniformly completely observable, then for $P_L(t)$, $Q_L(t)$ real symmetric, positive definite matrices, the observer will asymptotically estimate the state. The pair $(\tilde{A}(t), C(t))$ is uniformly completely observable if for every t , its observability Gramian $W_o(t, t_f)$ is positive definite (an alternative definition for observable) and $0 < a(\chi)I < W_o(t, t + \chi) \leq b(\chi)I$ for constants χ , $a(\chi)$, $b(\chi)$ [101].

The definition of the gain matrix comes from the derivative of $V_L(t) = e(t)^T P_L(t) e(t)$:

$$\begin{aligned}\dot{V}_L(t) &= \dot{e}(t)^T P_L(t) e(t) + e(t)^T \dot{P}_L(t) e(t) + e(t)^T P_L(t) \dot{e}(t) \\ -e(t)^T Q_L(t) e(t) &= \left((\tilde{A}(t) - L(t)C(t)) e(t) \right)^T P_L(t) e(t) + e(t)^T \dot{P}_L(t) e(t) \\ &\quad + e(t)^T P_L(t) \left((\tilde{A}(t) - L(t)C(t)) e(t) \right) \\ -e(t)^T Q_L(t) e(t) &= e(t)^T \tilde{A}(t)^T P_L(t) e(t) - e(t)^T C(t)^T L(t)^T P_L(t) e(t) + e(t)^T \dot{P}_L(t) e(t) \\ &\quad + e(t)^T P_L(t) \tilde{A}(t) e(t) - e(t)^T P_L(t) L(t) C(t) e(t),\end{aligned}$$

which implies

$$\begin{aligned}-Q_L(t) &= \tilde{A}(t)^T P_L(t) - C(t)^T L(t)^T P_L(t) + \dot{P}_L(t) + P_L(t) \tilde{A}(t) - P_L(t) L(t) C(t) \\ \dot{P}_L(t) &= -P_L(t) \tilde{A}(t) - \tilde{A}(t)^T P_L(t) - Q_L(t) + C(t)^T L(t)^T P_L(t) + P_L(t) L(t) C(t) \\ \dot{P}_L(t) &= -P_L(t) \tilde{A}(t) - \tilde{A}(t)^T P_L(t) - Q_L(t) + C(t)^T \left(\frac{1}{2} M(t) C(t) P_L(t)^{-1} \right) P_L(t) \\ &\quad + P_L(t) \left(\frac{1}{2} P_L(t)^{-1} C(t)^T M(t) \right) C(t) \\ \dot{P}_L(t) &= -P_L(t) \tilde{A}(t) - \tilde{A}(t)^T P_L(t) - Q_L(t) + C(t)^T M(t) C(t)\end{aligned}\tag{3.7}$$

if $L(t) = \frac{1}{2} P_L(t)^{-1} C(t)^T M(t)$ for $M(t)$ a real symmetric, positive definite matrix. However, not every choice of $P_L(t_0)$ and $M(t)$ may work for the chosen $Q_L(t)$. Resources [98], [135], [136] did not include a description of how $P_L(t_0)$, $M(t)$, and $Q_L(t)$ should be defined. The following results on $M(t)$, $Q_L(t)$, and a Riccati equation derived in terms of $S_L(t) = P_L(t)^{-1}$ are from our analysis of this process for constructing the gain matrix.

Multiplying (3.7) on the right by state transition matrix $\Phi(t, t_0)$ and on the left by its transpose produces

$$\begin{aligned}\Phi(t, t_0)^T \dot{P}_L(t) \Phi(t, t_0) &= -\Phi(t, t_0)^T P_L(t) \tilde{A}(t) \Phi(t, t_0) - \Phi(t, t_0)^T \tilde{A}(t)^T P_L(t) \Phi(t, t_0) \\ &\quad - \Phi(t, t_0)^T Q_L(t) \Phi(t, t_0) + \Phi(t, t_0)^T C(t)^T M(t) C(t) \Phi(t, t_0) \\ \Phi(t, t_0)^T \dot{P}_L(t) \Phi(t, t_0) &= -\Phi(t, t_0)^T P_L(t) \dot{\Phi}(t, t_0) - \dot{\Phi}(t, t_0)^T P_L(t) \Phi(t, t_0) \\ &\quad - \Phi(t, t_0)^T Q_L(t) \Phi(t, t_0) + \Phi(t, t_0)^T C(t)^T M(t) C(t) \Phi(t, t_0) \\ \left(\Phi(t, t_0)^T P_L(t) \Phi(t, t_0) \right)' &= -\Phi(t, t_0)^T Q_L(t) \Phi(t, t_0) + \Phi(t, t_0)^T C(t)^T M(t) C(t) \Phi(t, t_0),\end{aligned}$$

which results in $\Phi(t, t_0)^T P_L(t) \Phi(t, t_0) =$

$$P_L(t_0) - \int_{t_0}^t \Phi(s, t_0)^T Q_L(s) \Phi(s, t_0) ds + \int_{t_0}^t \Phi(s, t_0)^T C(s)^T M(s) C(s) \Phi(s, t_0) ds.$$

The rightmost integral is related to observability Gramian (3.3), which is symmetric and positive definite. Thus, for some $P_L(t_0)$ and $Q(t)$, $P_L(t)$ will remain positive definite if $M(t)$ is chosen large enough.

The positive definiteness of $P_L(t)$ may come into question when one of the eigenvalues of $P_L(t)$ becomes very large. Even though the smallest eigenvalue of $P_L(t)$ may not be close to zero, it may appear to be, causing MATLAB's inverse computation of $P_L(t)$ for $L(t)$ to be ill-conditioned. The solution implemented in our numerical algorithms is to use $P_L(t)^{-1}$ rather than $P_L(t)$ to construct the Riccati equation. If $S_L(t) = P_L(t)^{-1}$, then

$$\begin{aligned}\dot{S}_L(t) &= -P_L(t)^{-1} \dot{P}_L(t) P_L(t)^{-1} \\ &= -S_L(t) \left(-P_L(t) \tilde{A}(t) - \tilde{A}(t)^T P_L(t) - Q_L(t) + C(t)^T M(t) C(t) \right) S_L(t) \\ &= \tilde{A}(t) S_L(t) + S_L(t) \tilde{A}(t)^T + S_L(t) (Q_L(t) - C(t)^T M(t) C(t)) S_L(t).\end{aligned}$$

By letting $\dot{\Theta}(t, t_0) = -\tilde{A}(t)^T \Theta(t, t_0)$ with $\Theta(t_0, t_0) = I$,

$$\begin{aligned}\Theta(t, t_0)^T \dot{S}_L(t) \Theta(t, t_0) &= \Theta(t, t_0)^T \tilde{A}(t) S_L(t) \Theta(t, t_0) + \Theta(t, t_0)^T S_L(t) \tilde{A}(t)^T \Theta(t, t_0) \\ &\quad + \Theta(t, t_0)^T S_L(t) (Q_L(t) - C(t)^T M(t) C(t)) S_L(t) \Theta(t, t_0) \\ \Theta(t, t_0)^T \dot{S}_L(t) \Theta(t, t_0) &= -\dot{\Theta}(t, t_0)^T S_L(t) \Theta(t, t_0) - \Theta(t, t_0)^T S_L(t) \dot{\Theta}(t, t_0) \\ &\quad + \Theta(t, t_0)^T S_L(t) (Q_L(t) - C(t)^T M(t) C(t)) S_L(t) \Theta(t, t_0) \\ \left(\Theta(t, t_0)^T \dot{S}_L(t) \Theta(t, t_0) \right)' &= \Theta(t, t_0)^T S_L(t) (Q_L(t) - C(t)^T M(t) C(t)) S_L(t) \Theta(t, t_0).\end{aligned}$$

If $Q_L(t) = \bar{Q}(t) + C(t)^T M(t) C(t)$ for some real symmetric, positive definite matrix $\bar{Q}(t)$, then

$$\Theta(t, t_0)^T \dot{S}_L(t) \Theta(t, t_0) = S_L(t_0) + \int_{t_0}^t \Theta(s, t_0)^T S_L(s) \bar{Q}(s) S_L(s) \Theta(s, t_0) ds$$

indicates $S_L(t)$ will remain real symmetric, positive definite if $S_L(t_0)$ is real symmetric, positive definite. The gain matrix becomes $L(t) = \frac{1}{2} S_L(t) C(t)^T M(t)$ for the Riccati equation with $S_L(t)$.

The only requirements on $\tilde{A}(t)$, $\tilde{B}(t)$, and $C(t)$ mentioned in [98], [135], [136] were that they be continuous and bounded and that $(\tilde{A}(t), C(t))$ be uniformly completely observable. However, articles [2] (Theorem 5) and [3] also require that $\tilde{A}(t)$ be exponentially asymptotically stable if a matrix $P(t)$ similar to $P_L(t)$ in equation (3.7) is to exist. This stability requirement is revisited in Chapter 6, including a look at its effect on the positive definiteness of $S_L(t)$. Furthermore,

the results in Chapter 6 reveal the full-order observer also asymptotically estimates the state if the condition on $(\tilde{A}(t), C(t))$ is relaxed to detectable. In addition to the detectable definition based on the standard form for the unobservable system presented in Section 3.1, a linear time-varying system of ODEs is detectable if there exists a bounded gain matrix $L(t)$ such that equation (3.6) is uniformly asymptotically stable [159], [160] or exponentially stable [117]. Thus, if $(\tilde{A}(t), C(t))$ is not observable but an observer is still able to estimate the state, then the pair is detectable.

3.3 The Reduced-Order Observer

The construction of the reduced-order observer, proposed in [121], [122] and designed in [88], takes into consideration information about the state vector provided through the output equation. If a linear system of ODEs with output equation, system (3.1) for example, has an $n \times 1$ state vector $\tilde{x}(t)$, $m \times 1$ output vector $\tilde{y}(t)$, and a full row rank output matrix $C(t)$, then m state variables are known through the output equation. These m components of the state vector are described as measurable while the remaining $n - m$ components are referred to as unmeasurable [100], [121], [157]. The description reduced-order comes from constructing an observer to estimate only the unmeasurable state variables rather than the full state vector.

The construction of the reduced-order observer for the two linear cases is presented in separate subsections.

3.3.1 Linear Time-Invariant Case

The development of the reduced-order observer in this subsection is based on derivations described in [15], [100], [157]. The structure of the output equation is important in the construction of the reduced-order observer. The output should clearly identify m measurable components, which means no linear combinations of any of the n state variables should be included. As an example, $y(t) = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix}$ outputs $x_1(t)$ but $y(t) = \begin{bmatrix} 1 & 1 \end{bmatrix} \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix}$ does not identify a single measurable component. The desired structure for $\tilde{y}(t)$ can be obtained by applying an appropriate transformation to the system to be observed.

If R is a nonsingular transformation matrix such that $CR^{-1} = \begin{bmatrix} I & 0 \end{bmatrix}$, where I is an

$m \times m$ identity matrix, then for $\tilde{x}(t) = R^{-1}q(t)$,

$$\tilde{y}(t) = C\tilde{x}(t) = CR^{-1}q(t) = \begin{bmatrix} I & 0 \end{bmatrix} \begin{bmatrix} q_1(t) \\ \vdots \\ q_m(t) \\ q_{m+1}(t) \\ \vdots \\ q_n(t) \end{bmatrix} = \begin{bmatrix} q_1(t) \\ \vdots \\ q_m(t) \end{bmatrix} = q_m(t).$$

After substituting into equation (3.1a) for $\tilde{x}(t)$ and $\dot{\tilde{x}}(t) = R^{-1}\dot{q}(t)$,

$$\begin{aligned} \dot{\tilde{x}}(t) &= \tilde{A}\tilde{x}(t) + \tilde{B}v(t) \\ \implies R^{-1}\dot{q}(t) &= \tilde{A}R^{-1}q(t) + \tilde{B}v(t) \\ \implies \dot{q}(t) &= A_R q(t) + B_R v(t), \end{aligned} \quad (3.8)$$

where $A_R = R\tilde{A}R^{-1}$ and $B_R = R\tilde{B}$. Output equation $\tilde{y}(t) = q_m(t)$ equals the measurable components of transformed system (3.8).

Three methods were considered for constructing transformation matrix R . The first method, constructed for a $1 \times n$ output matrix, tested the components of C ($C(i)$) in order to define the nonzero elements of the transformation matrix as $\pm \frac{1}{C(i)}$. This method was abandoned without being extended to an $m \times n$ output matrix due to the possibility of MATLAB dividing by a number it did not realize was zero. The second method for determining R calculated the singular value decomposition of $R_{m \times n} = C_{m \times n} = U \cdot D \cdot V^T$; defined the next row of R as the transpose of the last column of V ; and repeated this algorithm for $R_{(m+1) \times n}$ and until R was $n \times n$. In consideration of the linear time-varying case, this method was abandoned due to the repeated computations of the singular value decomposition at one unit of time. Instead, the method chosen for constructing $R = \begin{bmatrix} C \\ C_R \end{bmatrix}$ comes from [15] and defines C_R as the transpose of the basis vectors for $N(C)$ found from calculating the singular value decomposition of C .

Transformed system (3.8) can be rewritten so it appears in terms of its measurable and unmeasurable components:

$$\dot{q}(t) = \begin{bmatrix} \dot{q}_m(t) \\ \dot{q}_u(t) \end{bmatrix} = \begin{bmatrix} A_{R_{11}} & A_{R_{12}} \\ A_{R_{21}} & A_{R_{22}} \end{bmatrix} \begin{bmatrix} q_m(t) \\ q_u(t) \end{bmatrix} + \begin{bmatrix} B_{R_1} \\ B_{R_2} \end{bmatrix} v(t)$$

$$\implies \dot{q}_m(t) = A_{R_{11}}q_m(t) + A_{R_{12}}q_u(t) + B_{R_1}v(t) \quad (3.9a)$$

$$\dot{q}_u(t) = A_{R_{21}}q_m(t) + A_{R_{22}}q_u(t) + B_{R_2}v(t). \quad (3.9b)$$

Equation (3.9b) is then used with a correction term to construct a reduced-order observer. Unlike full-order observer (3.5), the reduced-order observer is unable to incorporate $L(\tilde{y}(t) - \hat{y}(t))$ as its correction term since

$$\tilde{y}(t) - \hat{y}(t) = \tilde{y}(t) - \begin{bmatrix} I & 0 \end{bmatrix} \begin{bmatrix} q_m(t) \\ \hat{q}_u(t) \end{bmatrix} = \tilde{y}(t) - q_m(t) = 0.$$

Instead, the correction term comes from

$$\dot{\tilde{y}}(t) = \dot{q}_m(t) = A_{R_{11}}q_m(t) + A_{R_{12}}q_u(t) + B_{R_1}v(t).$$

This equation for the first derivative of the output can be rearranged by grouping measurable and known terms on one side and unmeasurable terms on the other:

$$\dot{\tilde{y}}(t) - A_{R_{11}}q_m(t) - B_{R_1}v(t) = A_{R_{12}}q_u(t).$$

Then for the correction term, $\dot{\tilde{y}}(t) - A_{R_{11}}q_m(t) - B_{R_1}v(t)$ represents the plant output and $A_{R_{12}}\hat{q}_u(t)$ represents the observer output, resulting in

$$\begin{aligned} \dot{\hat{q}}_u(t) &= A_{R_{22}}\hat{q}_u(t) + (A_{R_{21}}q_m(t) + B_{R_2}v(t)) \\ &\quad + L_R(\dot{\tilde{y}}(t) - A_{R_{11}}q_m(t) - B_{R_1}v(t) - A_{R_{12}}\hat{q}_u(t)). \end{aligned} \quad (3.10)$$

A change of variable is required to keep $\dot{\tilde{y}}(t)$ from appearing explicitly in equation (3.10):

$$\begin{aligned} \dot{\hat{q}}_u(t) - L_R\dot{\tilde{y}}(t) &= A_{R_{22}}\hat{q}_u(t) + (A_{R_{21}}q_m(t) + B_{R_2}v(t)) - L_R(A_{R_{11}}q_m(t) + B_{R_1}v(t) + A_{R_{12}}\hat{q}_u(t)) \\ &= (A_{R_{22}} - L_R A_{R_{12}})\hat{q}_u(t) + (A_{R_{21}} - L_R A_{R_{11}})q_m(t) + (B_{R_2} - L_R B_{R_1})v(t) \\ &= (A_{R_{22}} - L_R A_{R_{12}})\hat{q}_u(t) - (A_{R_{22}} - L_R A_{R_{12}})L_R\tilde{y}(t) \\ &\quad + (A_{R_{22}} - L_R A_{R_{12}})L_R\tilde{y}(t) + (A_{R_{21}} - L_R A_{R_{11}})q_m(t) + (B_{R_2} - L_R B_{R_1})v(t) \\ &= (A_{R_{22}} - L_R A_{R_{12}})(\hat{q}_u(t) - L_R\tilde{y}(t)) \\ &\quad + ((A_{R_{22}} - L_R A_{R_{12}})L_R + (A_{R_{21}} - L_R A_{R_{11}}))\tilde{y}(t) + (B_{R_2} - L_R B_{R_1})v(t). \end{aligned} \quad (3.11)$$

By letting $z(t) = \hat{q}_u(t) - L_R\tilde{y}(t)$ and substituting $z(t)$ and $\dot{z}(t) = \dot{\hat{q}}_u(t) - L_R\dot{\tilde{y}}(t)$ into equation (3.11) gives

$$\begin{aligned} \dot{z}(t) &= (A_{R_{22}} - L_R A_{R_{12}})z(t) \\ &\quad + ((A_{R_{22}} - L_R A_{R_{12}})L_R + (A_{R_{21}} - L_R A_{R_{11}}))\tilde{y}(t) + (B_{R_2} - L_R B_{R_1})v(t). \end{aligned}$$

After defining $D_R = A_{R22} - L_R A_{R12}$, $F_R = D_R L_R + (A_{R21} - L_R A_{R11})$, and $G_R = B_{R2} - L_R B_{R1}$, the reduced-order observer for the transformed system is

$$\dot{z}(t) = D_R z(t) + F_R \tilde{y}(t) + G_R v(t) \quad (3.12a)$$

$$\hat{q}_u(t) = z(t) + L_R \tilde{y}(t). \quad (3.12b)$$

The transformation $\hat{x}(t) = R^{-1} \hat{q}(t)$ is required to study the reduced-order observer results in terms of the original state vector.

The gain matrix L_R should be constructed so the difference between the unmeasurable components of the transformed system and the state variables being estimated by the reduced-order observer, $e_R(t) = q_u(t) - \hat{q}_u(t)$, goes to zero as $t \rightarrow \infty$. By considering

$$\begin{aligned} \dot{e}_R(t) &= \dot{q}_u(t) - \dot{\hat{q}}_u(t) \\ &= A_{R22} q_u(t) + (A_{R21} q_m(t) + B_{R2} v(t)) - A_{R22} \hat{q}_u(t) \\ &\quad - (A_{R21} q_m(t) + B_{R2} v(t)) - L_R (\tilde{y}(t) - A_{R11} q_m(t) - B_{R1} v(t) - A_{R12} \hat{q}_u(t)) \\ &= A_{R22} (q_u(t) - \hat{q}_u(t)) - L_R (A_{R12} q_u(t) - A_{R12} \hat{q}_u(t)) \\ &= A_{R22} (q_u(t) - \hat{q}_u(t)) - L_R A_{R12} (q_u(t) - \hat{q}_u(t)) \\ &= (A_{R22} - L_R A_{R12}) e_R(t) \end{aligned}$$

it becomes clear the eigenvalues of $A_{R22} - L_R A_{R12}$ can be placed so that $e_R(t) \rightarrow 0$ as $t \rightarrow \infty$ if (A_{R22}, A_{R12}) is observable or detectable. The observability and detectability checks for this reduced-order observer follow the linear time-invariant approaches outlined in Section 3.1 with A_{R22} substituted in for \tilde{A} and A_{R12} substituted in for C . Using these same matrix substitutions, the method for constructing gain matrix L_R follows the approach for constructing the full-order observer gain matrix outlined in Subsection 3.2.1.

3.3.2 Linear Time-Varying Case

The following process for constructing a reduced-order observer when the linear system of ODEs is time-varying comes from [135], [136], [174]. Similar to the linear time-invariant case, the system to be observed is transformed when necessary to affect the structure of the output equation. Again, letting transformation matrix $R(t)$ equal $\begin{bmatrix} C(t) \\ C_R(t) \end{bmatrix}$, where $C_R(t)$ is defined so $R(t)$ is nonsingular, results in $C(t)R(t)^{-1} = \begin{bmatrix} I & 0 \end{bmatrix}$. The derivative of $\tilde{x}(t) = R(t)^{-1}q(t)$ is

$$\dot{\tilde{x}}(t) = (R(t)^{-1})' q(t) + R(t)^{-1} \dot{q}(t) = \left(-R(t)^{-1} \dot{R}(t) R(t)^{-1} \right) q(t) + R(t)^{-1} \dot{q}(t).$$

Substituting into equation (3.1a) for $\tilde{x}(t)$ and $\dot{\tilde{x}}(t)$ produces

$$\begin{aligned} \left(-R(t)^{-1}\dot{R}(t)R(t)^{-1}\right)q(t) + R(t)^{-1}\dot{q}(t) &= \tilde{A}(t)R(t)^{-1}q(t) + \tilde{B}(t)v(t) \\ \implies R(t)^{-1}\dot{q}(t) &= \left(\tilde{A}(t)R(t)^{-1} - \left(-R(t)^{-1}\dot{R}(t)R(t)^{-1}\right)\right)q(t) + \tilde{B}(t)v(t) \\ \implies \dot{q}(t) &= \left(R(t)\tilde{A}(t)R(t)^{-1} + \dot{R}(t)R(t)^{-1}\right)q(t) + R(t)\tilde{B}(t)v(t), \end{aligned}$$

and with $\tilde{y}(t) = q_m(t)$ from

$$\tilde{y}(t) = C(t)\tilde{x}(t) = C(t)R(t)^{-1}q(t) = \begin{bmatrix} I & 0 \end{bmatrix} \begin{bmatrix} q_m(t) \\ q_u(t) \end{bmatrix},$$

the transformed system is

$$\dot{q}(t) = A_R(t)q(t) + B_R(t)v(t) \quad (3.14a)$$

$$\tilde{y}(t) = q_m(t) \quad (3.14b)$$

for $A_R(t) = R(t)\tilde{A}(t)R(t)^{-1} + \dot{R}(t)R(t)^{-1}$ and $B_R(t) = R(t)\tilde{B}(t)$.

If the output matrix is time-invariant, the transformation matrix is also constant. However, for a time-varying output matrix $C(t)$, the first derivative of transformation matrix $R(t)$ is needed to calculate the coefficient matrix of state vector $q(t)$ in equation (3.14a). Output matrix $C(t)$ is assumed smooth, but $C_R(t)$ needs to be smooth as well. In the time-invariant case, we chose to define C_R as the transpose of the basis vectors for $N(C)$ from a singular value decomposition of output matrix C . A related process called a smooth decomposition described in [109], [147] and outlined in Appendix A can be implemented in the time-varying case to find smooth basis vectors for $N(C(t))$. The process is extended in [109] to calculate the first derivatives of the basis vectors found during the smooth decomposition. An alternative method for determining a smooth $C_R(t)$ and its first derivative is to utilize either MATLAB's or Maple's commands for finding a nullspace and for taking a derivative with t defined symbolically.

Unlike the linear time-invariant reduced-order observer in Subsection 3.3.1, the derivation of this subsection's reduced-order observer for transformed system (3.14) begins by defining

$$z(t) = T(t)\hat{q}(t) = \begin{bmatrix} T_1(t) & T_2(t) \end{bmatrix} \begin{bmatrix} q_m(t) \\ \hat{q}_u(t) \end{bmatrix}. \quad (3.15)$$

Equation (3.15) represents an output for the observer-estimated unmeasurable components and

when considered together with output equation (3.14b),

$$\begin{bmatrix} \tilde{y}(t) \\ z(t) \end{bmatrix} = \begin{bmatrix} q_m(t) \\ T(t)\hat{q}(t) \end{bmatrix} = \begin{bmatrix} I & 0 \\ T_1(t) & T_2(t) \end{bmatrix} \begin{bmatrix} q_m(t) \\ \hat{q}_u(t) \end{bmatrix}. \quad (3.16)$$

System (3.16) reveals $T(t)$ should be chosen so $\hat{q}(t)$ can be solved for explicitly. If

$$\begin{bmatrix} I & 0 \\ T_1(t) & T_2(t) \end{bmatrix}^{-1} = \begin{bmatrix} V_1(t) & P_1(t) \\ V_2(t) & P_2(t) \end{bmatrix},$$

where $V_1(t)$ is $m \times m$ and $P_2(t)$ is $(n-m) \times (n-m)$, then

$$\begin{bmatrix} I & 0 \\ T_1(t) & T_2(t) \end{bmatrix} \begin{bmatrix} V_1(t) & P_1(t) \\ V_2(t) & P_2(t) \end{bmatrix} = \begin{bmatrix} I & 0 \\ 0 & I \end{bmatrix}$$

implies $V_1(t)$ is an identity matrix, $P_1(t)$ is a zero matrix, $T_1(t) + T_2(t)V_2(t) = 0$, and $P_2(t)$ needs to be nonsingular since $T_2(t)P_2(t) = I$. Choosing $T(t) = \begin{bmatrix} -P_2(t)^{-1}V_2(t) & P_2(t)^{-1} \end{bmatrix}$ appropriately is now dependent on the choices of matrices $V_2(t)$ and $P_2(t)$.

After substituting into equation (3.15) for $T(t)$,

$$z(t) = T(t)\hat{q}(t) = -P_2(t)^{-1}V_2(t)q_m(t) + P_2(t)^{-1}\hat{q}_u(t),$$

the derivative of $z(t)$ is

$$\begin{aligned} \dot{z}(t) &= -\dot{P}_2(t)^{-1}V_2(t)q_m(t) - P_2(t)^{-1}\dot{V}_2(t)q_m(t) - P_2(t)^{-1}V_2(t)\dot{q}_m(t) \\ &\quad + \dot{P}_2(t)^{-1}\hat{q}_u(t) + P_2(t)^{-1}\dot{\hat{q}}_u(t) \\ &= -\left(-P_2(t)^{-1}\dot{P}_2(t)P_2(t)^{-1}\right)V_2(t)q_m(t) - P_2(t)^{-1}\dot{V}_2(t)q_m(t) \\ &\quad - P_2(t)^{-1}V_2(t)(A_{R_{11}}(t)q_m(t) + A_{R_{12}}(t)\hat{q}_u(t) + B_{R_1}(t)v(t)) \\ &\quad + \left(-P_2(t)^{-1}\dot{P}_2(t)P_2(t)^{-1}\right)\hat{q}_u(t) + P_2(t)^{-1}(A_{R_{21}}(t)q_m(t) + A_{R_{22}}(t)\hat{q}_u(t) + B_{R_2}(t)v(t)) \\ &= P_2(t)^{-1}\left(A_{R_{22}}(t) - V_2(t)A_{R_{12}}(t) - \dot{P}_2(t)P_2(t)^{-1}\right)\hat{q}_u(t) \\ &\quad + P_2(t)^{-1}\left(A_{R_{21}}(t) - V_2(t)A_{R_{11}}(t) + \dot{P}_2(t)P_2(t)^{-1}V_2(t) - \dot{V}_2(t)\right)q_m(t) \\ &\quad + P_2(t)^{-1}(-V_2(t)B_{R_1}(t) + B_{R_2}(t))v(t), \end{aligned}$$

where $\dot{q}_m(t)$ and $\dot{\hat{q}}_u(t)$ come from equation (3.14a) written in terms of its measurable and unmeasurable state variables (the linear time-varying version of (3.9)). The explicit solution of $\hat{q}(t)$ provides $q_m(t) = \tilde{y}(t)$ and $\hat{q}_u(t) = V_2(t)\tilde{y}(t) + P_2(t)z(t)$ so $\dot{z}(t)$ becomes

$$\begin{aligned}
\dot{z}(t) = & \left(P_2(t)^{-1} (A_{R22}(t) - V_2(t)A_{R12}(t)) P_2(t) - P_2(t)^{-1} \dot{P}_2(t) \right) z(t) \\
& + P_2(t)^{-1} \left(A_{R21}(t) - V_2(t)A_{R11}(t) + A_{R22}(t)V_2(t) - V_2(t)A_{R12}(t)V_2(t) - \dot{V}_2(t) \right) \tilde{y}(t) \\
& + P_2(t)^{-1} (-V_2(t)B_{R1}(t) + B_{R2}(t)) v(t).
\end{aligned}$$

After defining

$$D_R(t) = P_2(t)^{-1} (A_{R22}(t) - V_2(t)A_{R12}(t)) P_2(t) - P_2(t)^{-1} \dot{P}_2(t),$$

$$F_R(t) = P_2(t)^{-1} \left(A_{R21}(t) - V_2(t)A_{R11}(t) + A_{R22}(t)V_2(t) - V_2(t)A_{R12}(t)V_2(t) - \dot{V}_2(t) \right),$$

$$G_R(t) = P_2(t)^{-1} (-V_2(t)B_{R1}(t) + B_{R2}(t)),$$

the reduced-order observer for the transformed system is

$$\dot{z}(t) = D_R(t)z(t) + F_R(t)\tilde{y}(t) + G_R(t)v(t) \quad (3.17a)$$

$$\hat{q}_u(t) = V_2(t)\tilde{y}(t) + P_2(t)z(t). \quad (3.17b)$$

The transformation $\hat{x}_R(t) = R(t)^{-1}\hat{q}(t)$ returns the reduced-order observer results in terms of the original state vector.

For reduced-order observer (3.17), the convergence of $e(t) = q(t) - \hat{q}(t)$ to zero as time goes to infinity is dependent on the choice of $V_2(t)$. The second error equation $\epsilon(t) = T(t)q(t) - z(t)$, a measure of how well the reduced-order observer is estimating the unmeasurable components, is required to show this dependence. The first derivative of $\epsilon(t)$ is

$$\begin{aligned}
\dot{\epsilon}(t) = & \dot{T}(t)q(t) + T(t)\dot{q}(t) - \dot{z}(t) \\
= & \dot{T}(t)q(t) + T(t)(A_R(t)q(t) + B_R(t)v(t)) - (D_R(t)z(t) + F_R(t)\tilde{y}(t) + G_R(t)v(t)) \\
= & D_R(t)T(t)q(t) - D_R(t)z(t) - D_R(t)T(t)q(t) + T(t)A_R(t)q(t) + \dot{T}(t)q(t) \\
& - F_R(t)C(t)R(t)^{-1}q(t) + (T(t)B_R(t) - G_R(t))v(t) \\
= & D_R(t)\epsilon(t) - \left(D_R(t)T(t) - T(t)A_R(t) - \dot{T}(t) + F_R(t)C(t)R(t)^{-1} \right) q(t) \\
& + (T(t)B_R(t) - G_R(t))v(t).
\end{aligned}$$

The coefficient of $q(t)$ equals zero,

$$\begin{aligned}
& D_R(t)T(t) - T(t)A_R(t) \\
= & D_R(t)P_2(t)^{-1} \left[\begin{array}{cc} -V_2(t) & I \end{array} \right] + P_2(t)^{-1} \left[\begin{array}{cc} V_2(t)A_{R11}(t) - A_{R21}(t) & V_2(t)A_{R12}(t) - A_{R22}(t) \end{array} \right] \\
= & \left[\begin{array}{cc} -(P_2(t)^{-1})'V_2(t) & (P_2(t)^{-1})' \end{array} \right] + \left[\begin{array}{cc} -P_2(t)^{-1}\dot{V}_2(t) & 0 \end{array} \right] - \left[\begin{array}{cc} -P_2(t)^{-1}\dot{V}_2(t) & 0 \end{array} \right] \\
& + \left[\begin{array}{cc} -P_2(t)^{-1}(A_{R21}(t) - V_2(t)A_{R11}(t) + A_{R22}(t)V_2(t) - V_2(t)A_{R12}(t)V_2(t)) & 0 \end{array} \right] \\
= & \left[\begin{array}{cc} -(P_2(t)^{-1}V_2(t))' & (P_2(t)^{-1})' \end{array} \right] - F_R(t) \left[\begin{array}{cc} I & 0 \end{array} \right] \\
= & \dot{T}(t) - F_R(t)C(t)R(t)^{-1},
\end{aligned}$$

as does the coefficient of $v(t)$,

$$G_R(t) = \begin{bmatrix} -P_2(t)^{-1}V_2(t) & P_2(t)^{-1} \end{bmatrix} \begin{bmatrix} B_{R_1}(t) \\ B_{R_2}(t) \end{bmatrix} = T(t)B_R(t),$$

so the equation for $\dot{\epsilon}(t)$ simplifies to $\dot{\epsilon}(t) = D_R(t)\epsilon(t)$. Matrix $P_2(t)$ can be taken constant as long as it is nonsingular, resulting in $\dot{\epsilon}(t) = P_2^{-1}(A_{R_{22}}(t) - V_2(t)A_{R_{12}}(t))P_2\epsilon(t)$. This equation shows the choice of matrix $V_2(t)$ affects whether or not $\epsilon(t) \rightarrow 0$ at $t \rightarrow \infty$.

If $e(t)$ is expressed as $\begin{bmatrix} e_1(t) \\ e_2(t) \end{bmatrix}$, where the row dimensions of $e_1(t)$ and $e_2(t)$ equal the number of measurable and unmeasurable components, respectively, then

$$\begin{aligned} e(t) &= \begin{bmatrix} e_1(t) \\ e_2(t) \end{bmatrix} = q(t) - \hat{q}(t) \\ &= \begin{bmatrix} 0 \\ -V_2(t)q_m(t) + q_u(t) - P_2z(t) \end{bmatrix} \\ &= \begin{bmatrix} 0 \\ P_2 \end{bmatrix} \left(\begin{bmatrix} -P_2^{-1}V_2(t) & P_2^{-1} \end{bmatrix} \begin{bmatrix} q_m(t) \\ q_u(t) \end{bmatrix} - z(t) \right) \\ &= \begin{bmatrix} 0 \\ P_2 \end{bmatrix} \epsilon(t) \end{aligned}$$

reveals $e_1(t) = 0$ and $e_2(t) = P_2\epsilon(t)$. Thus, the derivative of $e(t)$,

$$\begin{aligned} \dot{e}(t) &= \begin{bmatrix} 0 \\ P_2\dot{\epsilon}(t) \end{bmatrix} \\ &= \begin{bmatrix} 0 \\ P_2(P_2^{-1}(A_{R_{22}}(t) - V_2(t)A_{R_{12}}(t))P_2)\epsilon(t) \end{bmatrix} \\ &= \begin{bmatrix} 0 & 0 \\ 0 & A_{R_{22}}(t) - V_2(t)A_{R_{12}}(t) \end{bmatrix} \begin{bmatrix} e_1(t) \\ e_2(t) \end{bmatrix}, \end{aligned}$$

demonstrates that for an appropriate $V_2(t)$ (and a constant, nonsingular P_2), $e(t) \rightarrow 0$ as $t \rightarrow \infty$ and reduced-order observer (3.17) converges to the unmeasurable state variables.

The construction of gain matrix $V_2(t)$ follows the process outlined for the construction of gain matrix $L(t)$ in Subsection 3.2.2 with $A_{R_{22}}(t)$ and $A_{R_{12}}(t)$ inserted in for $\tilde{A}(t)$ and $C(t)$, respectively. The resulting Riccati equation is

$$\dot{S}_L(t) = A_{R_{22}}(t)S_L(t) + S_L(t)A_{R_{22}}(t)^T + S_L(t)(Q_L(t) - A_{R_{12}}(t)^TM(t)A_{R_{12}}(t))S_L(t)$$

and is used to compute $V_2(t) = \frac{1}{2}S_L(t)A_{R_{12}}(t)^T M(t)$. Additionally, the first derivative of $V_2(t)$,

$$\dot{V}_2(t) = \frac{1}{2} \left(\dot{S}_L(t)A_{R_{12}}(t)^T M(t) + S_L(t)(A_{R_{12}}(t)^T)' M(t) + S_L(t)A_{R_{12}}(t)^T \dot{M}(t) \right),$$

is required to determine $F_R(t)$, the coefficient matrix of $\tilde{y}(t)$ in equation (3.17a). The calculations of $\dot{S}_L(t)$ and $\dot{M}(t)$ are straightforward. The Riccati equation provides the first derivative of $S_L(t)$. The user determines $M(t)$ and therefore has control over constructing a matrix for which $\dot{M}(t)$ exists and is known without differentiating any numerically computed quantities.

Recall, matrix $A_{R_{12}}(t)$ is a submatrix of $A_R(t) = R(t)\tilde{A}(t)R(t)^{-1} + \dot{R}(t)R(t)^{-1}$, so knowing the first derivative of $\tilde{A}(t)$ and the second derivative of $R(t)$ when the output matrix is time-varying is necessary for calculating $(A_{R_{12}}(t)^T)'$. The process from [109] mentioned earlier in this subsection for finding $C_R(t)$ and its first derivative has yet to be extended to finding higher order derivatives of matrices computed using the smooth decomposition method. Instead, the symbolic results from finding $C_R(t)$ and both its first and second derivatives using either MATLAB or Maple are implemented in our numerical algorithms when defining $R(t)$ and its derivatives.

To avoid introducing numerical error by differentiating the computed value of $\tilde{A}(t)$ at each time t , $(\tilde{A}(t))'$ can be defined in terms of matrices with known derivatives for all three completions. For both the least squares completion and the stabilized least squares completion, $\tilde{A}(t)$ is taken as the first n rows of $-\mathcal{E}(t)^\dagger \mathcal{F}(t)$. Thus, $(\tilde{A}(t))'$ is taken as the first n rows of $- (\mathcal{E}(t)^\dagger)' \mathcal{F}(t) - \mathcal{E}(t)^\dagger \dot{\mathcal{F}}(t)$, where

$$\begin{aligned} (\mathcal{E}(t)^\dagger)' &= -\mathcal{E}(t)^\dagger \dot{\mathcal{E}}(t) \mathcal{E}(t)^\dagger + (I - \mathcal{E}(t)^\dagger \mathcal{E}(t)) (\mathcal{E}(t)^T)' (\mathcal{E}(t)^\dagger)^T \mathcal{E}(t)^\dagger \\ &\quad + \mathcal{E}(t)^\dagger (\mathcal{E}(t)^\dagger)^T (\mathcal{E}(t)^T)' (I - \mathcal{E}(t) \mathcal{E}(t)^\dagger). \end{aligned} \quad [51]$$

Recall, for the alternative stabilized completion,

$$\tilde{A}(t) = - \begin{bmatrix} Z_{1,0}(t)^T E(t) \\ Z_2(t)^T \mathcal{F}(t) \end{bmatrix}^{-1} \begin{bmatrix} Z_{1,0}(t)^T F(t) \\ (Z_2(t)^T \mathcal{F}(t))' + \lambda Z_2(t)^T \mathcal{F}(t) \end{bmatrix}.$$

The first derivative of $\tilde{A}(t)$ requires the second derivative of matrix $Z_2(t)^T$. However, if $Z_2(t)^T$ is calculated from a smooth decomposition as suggested in Subsection 2.3.2, then only the first derivative of the computed matrix is known. Alternatively, either MATLAB's or Maple's commands for determining range and nullspace bases can be used to symbolically define $Z_2(t)^T$ as well as $T_2(t)$ (used to find $Z_{1,0}(t)^T$ and $Z_{1,0}(t)^T$). Thus, the alternative stabilized completion's coefficient matrix $\tilde{A}(t)$ can be computed symbolically, allowing MATLAB or Maple to return the first derivative of $\tilde{A}(t)$ as an expression in terms of t .

Once $(\tilde{A}(t))'$, $R(t)$, and the necessary derivatives of $R(t)$ are defined, submatrix $\dot{A}_{R_{12}}(t)$ is taken from

$$\begin{aligned}\dot{A}_R(t) &= \dot{R}(t)\tilde{A}(t)R(t)^{-1} + R(t)(\tilde{A}(t))'R(t)^{-1} + R(t)\tilde{A}(t)(R(t)^{-1})' \\ &\quad + \ddot{R}(t)R(t)^{-1} + \dot{R}(t)(R(t)^{-1})',\end{aligned}$$

where $(R(t)^{-1})' = -R(t)^{-1}\dot{R}(t)R(t)^{-1}$. If the output matrix is time-invariant, the derivative of $A_R(t)$ simplifies to

$$\dot{A}_R(t) = R(\tilde{A}(t))'R^{-1}.$$

Transposing $\dot{A}_{R_{12}}(t)$ produces the desired matrix $(A_{R_{12}}(t)^T)'$ for calculating $\dot{V}_2(t)$ without differentiating a numerically computed quantity.

3.4 The Maximally Reduced Observer

One goal of designing a reduced-order observer is to reduce the number of state variables the observer must estimate by utilizing the information the output provides about the physical system. A new observer we introduced in [25] uses the constraints defining the solution manifold of a system of DAEs to provide additional information about previously unmeasurable components so the order of the observer is reduced even more. We refer to these observers as maximally reduced observers.

The derivative array equations $\mathcal{E}(t)w(t) + \mathcal{F}(t)x(t) = \mathcal{B}(t)v(t)$ found without using stabilized differentiation (see Section 2.1) can be manipulated to become a system of algebraic equations characterizing the solution manifold:

$$0 = -\mathcal{G}(t)\mathcal{F}(t)x(t) + \mathcal{G}(t)\mathcal{B}(t)v(t), \quad (3.18)$$

where matrix $\mathcal{G}(t)$ has maximal rank and is determined so that $\mathcal{G}(t)\mathcal{E}(t) = 0$. In the linear time-invariant case, two methods were considered for determining \mathcal{G} . The first method defined $\mathcal{G}_* = (I - \mathcal{E}\mathcal{E}^\dagger)$ since $(I - \mathcal{E}\mathcal{E}^\dagger)\mathcal{E} = \mathcal{E} - \mathcal{E}\mathcal{E}^\dagger\mathcal{E} = 0$ from the properties of the Moore-Penrose pseudoinverse [128]. However, the number of rows of \mathcal{G}_* equaled those of \mathcal{E} , so an additional step formed matrix \mathcal{G} from the linearly independent rows of \mathcal{G}_* . In order to simplify the algorithm in both the linear time-invariant and time-varying cases, the chosen method defines $\mathcal{G}(t)$ from the results of a singular value decomposition of $\mathcal{E}(t)$. Wanting $\mathcal{G}(t)\mathcal{E}(t)$ to equal zero,

$$(\mathcal{G}(t)\mathcal{E}(t))^T = 0^T \implies \mathcal{E}(t)^T\mathcal{G}(t)^T = 0^T$$

indicates the columns of $\mathcal{G}(t)^T$ are in the nullspace of $\mathcal{E}(t)^T$. If $\mathcal{E}(t)$ has rank r and the index of the linear system of DAEs is k , the last $(k+1)n-r$ columns of $U(t)$ from the singular value decomposition of $\mathcal{E}(t)$, $\mathcal{E}(t) = U(t) \cdot D(t) \cdot V(t)^T$, form an orthonormal basis for $N(\mathcal{E}(t)^T)$. Thus, $\mathcal{G}(t)$ is defined as the transpose of these columns.

Knowing matrices $\mathcal{G}(t)$ and $\mathcal{B}(t)$ and vector $v(t)$, product $\mathcal{G}(t)\mathcal{B}(t)v(t)$ behaves as another output, permitting the formation of

$$y_{\Xi}(t) = \begin{bmatrix} y(t) \\ \mathcal{G}(t)\mathcal{B}(t)v(t) \end{bmatrix}_{\Xi} = \begin{bmatrix} C(t) \\ \mathcal{G}(t)\mathcal{F}(t) \end{bmatrix}_{\Xi} x(t) = C_{\Xi}(t)x(t), \quad (3.19)$$

where $C_{\Xi}(t)$ is full row rank. We refer to equation (3.19) as the extended output equation and to $C_{\Xi}(t)$ as the extended output matrix. The structure of the extended output equation is not unique. For example, the user may decide to save all of output matrix $C(t)$ and then complete the extension with the rows of $\mathcal{G}(t)\mathcal{F}(t)$ linearly independent from the rows of $C(t)$. Alternatively, all of $\mathcal{G}(t)\mathcal{F}(t)$ could be saved and then extended with the appropriate rows of output matrix $C(t)$. However the extension occurs, it is necessary to keep track of the rows of $C(t)$ and $\mathcal{G}(t)\mathcal{F}(t)$ used to form extended output matrix $C_{\Xi}(t)$ so the correct rows of $y(t)$ and $\mathcal{G}(t)\mathcal{B}(t)v(t)$ are selected to define $y_{\Xi}(t)$.

Once $C_{\overline{R}}(t)$ is determined so transformation matrix $\overline{R}(t) = \begin{bmatrix} C_{\Xi}(t) \\ C_{\overline{R}}(t) \end{bmatrix}$ is nonsingular, the construction of the maximally reduced observer follows the process outlined in Section 3.3 for the construction of the reduced-order observer. For example, in the linear time-invariant case, the construction process produces maximally reduced observer

$$\begin{aligned} \dot{z}(t) &= D_{\overline{R}}z(t) + F_{\overline{R}}\tilde{y}_{\Xi}(t) + G_{\overline{R}}v(t) \\ \hat{q}_u(t) &= z(t) + L_{\overline{R}}\tilde{y}_{\Xi}(t), \end{aligned}$$

where $D_{\overline{R}} = A_{\overline{R}_{22}} - L_{\overline{R}}A_{\overline{R}_{12}}$; $F_{\overline{R}} = D_{\overline{R}}L_{\overline{R}} + (A_{\overline{R}_{21}} - L_{\overline{R}}A_{\overline{R}_{11}})$; $G_{\overline{R}} = B_{\overline{R}_2} - L_{\overline{R}}B_{\overline{R}_1}$; $\tilde{y}_{\Xi}(t) = \begin{bmatrix} \tilde{y}(t) \\ \mathcal{G}\mathcal{B}v(t) \end{bmatrix}_{\Xi}$; and $L_{\overline{R}}$ is an appropriately defined gain matrix. Transformation $\hat{x}(t) = (\overline{R})^{-1}\hat{q}(t)$ returns the observer results for pair $(A_{\overline{R}_{22}}, A_{\overline{R}_{12}})$ in terms of the original state vector. Similar modifications result for the maximally reduced observer in the linear time-varying case. Based on the observer development in Subsection 3.3.2, the first and second derivatives of $\overline{R}(t) = \begin{bmatrix} C_{\Xi}(t) \\ C_{\overline{R}}(t) \end{bmatrix}$ are required for the construction of the maximally reduced observer. Thus, for our numerical algorithms, we again select either MATLAB's or Maple's symbolic commands to define $\mathcal{G}(t)\mathcal{F}(t)$, $C_{\overline{R}}(t)$, and their necessary derivatives.

3.5 DAE Manifold Observer

An additional observer we developed for a linear time-invariant system of DAEs

$$E\dot{x}(t) + Fx(t) = Bu(t) \quad (3.20)$$

with output equation $y(t) = Cx(t)$ can be constructed without using a completion. The process combines information from the canonical form of a matrix pencil with the constraints defining the solution manifold of the linear time-invariant system of DAEs. We refer to this observer as the DAE manifold observer. Coefficient matrix E is $n \times n$, full row rank output matrix C is $m \times n$, and the solution manifold is d -dimensional. The extension of this observer to linear time-varying systems of DAEs is discussed in Chapter 7.

From [28] (Theorem 2.3.2), if matrix pencil $sE + F$ of equation (3.20) is regular, then nonsingular matrices \mathcal{P} and \mathcal{Q} exist such that

$$\mathcal{P}E\mathcal{Q} = \begin{bmatrix} I & 0 \\ 0 & N \end{bmatrix}, \quad \mathcal{P}F\mathcal{Q} = \begin{bmatrix} \Omega & 0 \\ 0 & I \end{bmatrix}.$$

Matrix N is nilpotent with degree of nilpotency equal to the index of the linear time-invariant system of DAEs. Recall, matrix pencil $sE + F$ is regular if for at least one value of s , the determinant of $sE + F$ does not equal zero. Substituting into equation (3.20) for $x(t) = \mathcal{Q}p(t)$ and $\dot{x}(t) = \mathcal{Q}\dot{p}(t)$ and then multiplying the equation on the left by \mathcal{P} produces

$$\mathcal{P}E\mathcal{Q}\dot{p}(t) + \mathcal{P}F\mathcal{Q}p(t) = \mathcal{P}Bu(t). \quad (3.21)$$

Equation (3.21) can be rewritten as

$$\begin{bmatrix} I & 0 \\ 0 & N \end{bmatrix} \begin{bmatrix} \dot{p}_1(t) \\ \dot{p}_2(t) \end{bmatrix} + \begin{bmatrix} \Omega & 0 \\ 0 & I \end{bmatrix} \begin{bmatrix} p_1(t) \\ p_2(t) \end{bmatrix} = \begin{bmatrix} \mathcal{P}B_1 \\ \mathcal{P}B_2 \end{bmatrix} u(t)$$

and then separated into

$$\dot{p}_1(t) + \Omega p_1(t) = \mathcal{P}B_1 u(t) \quad (3.22a)$$

$$N\dot{p}_2(t) + p_2(t) = \mathcal{P}B_2 u(t) \quad (3.22b)$$

for $p(t) = \begin{bmatrix} [p_1(t)]_{d \times 1} \\ [p_2(t)]_{(n-d) \times 1} \end{bmatrix}$. Let

$$\mathcal{G}\mathcal{F}x(t) = \mathcal{G}\mathcal{B}v(t) \quad (3.23)$$

characterize equation (3.20)'s solution manifold (see Subsection 3.4 for explanation of equation

(3.23)) and assume \mathcal{G} is full row rank. Substituting into equation (3.23) for $x(t) = \mathcal{Q}p(t)$ reveals

$$\mathcal{G}\mathcal{F}x(t) = \mathcal{G}\mathcal{F}\mathcal{Q}p(t) = \begin{bmatrix} 0 & \mathbf{G} \end{bmatrix} \begin{bmatrix} p_1(t) \\ p_2(t) \end{bmatrix} = \mathcal{G}\mathcal{B}v(t).$$

The multiplication of $\mathcal{G}\mathcal{F}$ on the right by \mathcal{Q} causes a column compression and forms an invertible $(n-d) \times (n-d)$ matrix \mathbf{G} . Thus, $p_2(t)$ can be solved for explicitly,

$$p_2(t) = \mathbf{G}^{-1}\mathcal{G}\mathcal{B}v(t), \quad (3.24)$$

and does not need to be estimated.

With $p_2(t)$ a known value, a modified output equation is defined for observing equation (3.22a). In terms of block matrices, output equation $y(t) = C\mathcal{Q}p(t)$ has structure

$$y(t) = \begin{bmatrix} C_{11} & C_{12} \end{bmatrix} \begin{bmatrix} \mathcal{Q}_{11} & \mathcal{Q}_{12} \\ \mathcal{Q}_{21} & \mathcal{Q}_{22} \end{bmatrix} \begin{bmatrix} p_1(t) \\ p_2(t) \end{bmatrix},$$

where C_{11} is $m \times d$, \mathcal{Q}_{11} is $d \times d$, and \mathcal{Q}_{22} is $(n-d) \times (n-d)$. Matrix multiplication produces

$$y(t) = (C_{11}\mathcal{Q}_{11} + C_{12}\mathcal{Q}_{21})p_1(t) + (C_{11}\mathcal{Q}_{12} + C_{12}\mathcal{Q}_{22})p_2(t),$$

which, along with the available information on $p_2(t)$, allows the modification

$$\bar{y}(t) = y(t) - (C_{11}\mathcal{Q}_{12} + C_{12}\mathcal{Q}_{22})p_2(t) = (C_{11}\mathcal{Q}_{11} + C_{12}\mathcal{Q}_{21})p_1(t).$$

The system to be observed becomes

$$\dot{p}_1(t) = -\Omega p_1(t) + \mathcal{P}B_1 u(t) \quad (3.25a)$$

$$\bar{y}(t) = (C_{11}\mathcal{Q}_{11} + C_{12}\mathcal{Q}_{21})p_1(t). \quad (3.25b)$$

If the output matrix in equation (3.25b) is not full row rank, the output equation should be redefined accordingly. The full-order observer for system (3.25) is

$$\dot{\hat{p}}_1(t) = -\Omega\hat{p}_1(t) + \mathcal{P}B_1 u(t) + L(\bar{y}(t) - \hat{y}(t)),$$

requiring transformation

$$\hat{x}(t) = \mathcal{Q} \begin{bmatrix} \hat{p}_1(t) \\ p_2(t) \end{bmatrix}$$

to study the observer results in terms of original state vector $x(t)$.

3.6 Completion Stability Revisited

The discussion on completion stability in Section 2.4 described the effects a constant scalar λ has on a linear time-invariant completion's additional dynamics. These effects influence the observability of the completion. If the only unobservable eigenvalues of a completion come from its additional dynamics eigenvalues, which equal $-\lambda$, the completion is detectable when $\lambda > 0$ and unobservable when $\lambda \leq 0$ (recall, $\lambda = 0$ for the least squares completion).

Although the additional dynamics also influence the observability of a linear time-varying completion, the consequences from choosing a particular λ are not as apparent. For a constant scalar λ in the linear time-invariant case, the Jordan canonical form of a completion's coefficient matrix \tilde{A} reveals the potential for a repeated additional dynamics eigenvalue to have linearly dependent eigenvectors. In numerical algorithms, stabilized differentiation may be implemented using λ times an appropriately sized identity matrix. Thus, one proposal to reduce the number of linearly dependent eigenvectors and thereby increase the probability of having an observable system is to stabilize using a stabilization parameter matrix Λ that is a constant diagonal matrix with distinct entries. We suspect this modification to stabilized differentiation $(\frac{d}{dt} + \Lambda)$ also improves the chances for observability in the linear time-varying case. The results from implementing $\frac{d}{dt} + \Lambda$ for a linear time-varying example are analyzed in Chapter 6.

Chapter 4

Theory

4.1 Completions and Observability

A theoretical contribution of our research is a proof showing a matrix pencil eigenvalue is unobservable for all completions if it is unobservable for one completion. The introduction of Chapter 2 mentioned all completions of a linear system of DAEs are related by a general form. Two completions of a homogeneous linear time-invariant system of DAEs $E\dot{x}(t) + Fx(t) = 0$ can be written as $\dot{\tilde{x}}(t) = \tilde{A}\tilde{x}(t)$ and $\dot{\check{x}}(t) = (\check{A} + \Delta M)\check{x}(t)$ [46]. Recall from Section 3.4, the solution manifold of a linear system of DAEs can be characterized from the derivative array equations found with differential polynomial $\frac{d}{dt}$; notationally, $\mathcal{E}w(t) + \mathcal{F}x(t) = 0$ for system $E\dot{x}(t) + Fx(t) = 0$. For a maximal rank (preferably full row rank) matrix \mathcal{G} such that $\mathcal{G}\mathcal{E} = 0$, the nullspace of matrix M , $N(M)$ such that $M = \mathcal{G}\mathcal{F}$, characterizes the solution manifold. Matrix Δ describes the relationship between the completions' coefficient matrices \check{A} and \tilde{A} ; that is, \check{A} equals \tilde{A} plus a multiple Δ of M .

Theorem 4.1. *For some linear time-invariant output matrix C , let \mathcal{O}_Δ and $\mathcal{O}_{\tilde{A}}$ designate the observability matrices for the respective pairs $(\tilde{A} + \Delta M, C)$ and (\tilde{A}, C) . Also, let $N(Y)$ denote the nullspace of a general matrix Y . Then*

$$N(M) \cap N(\mathcal{O}_\Delta) = N(M) \cap N(\mathcal{O}_{\tilde{A}}).$$

The proof of Theorem 4.1 includes the established theory reviewed in Theorem 4.2.

Theorem 4.2. *$N(M)$ is \tilde{A} -invariant. That is, if $\tilde{x}(t) \in N(M)$, then $\tilde{A}\tilde{x}(t) \in N(M)$. [28]*

Proof, Thm 4.2. For system $\dot{\tilde{x}}(t) = \tilde{A}\tilde{x}(t)$, select any initial condition $\tilde{x}(t_0) = \tilde{x}_0 \in N(M)$. By definition, solutions that begin on the solution manifold remain on the solution manifold. Since $N(M)$ characterizes the solution manifold, the solution of $\dot{\tilde{x}}(t) = \tilde{A}\tilde{x}(t)$ with $\tilde{x}(t_0) = \tilde{x}_0$ is also

a solution of the linear time-invariant system of DAEs. Thus, $\tilde{x}(t) \in N(M)$ for all t . Then

$$M\tilde{x}(t) = 0 \implies M\dot{\tilde{x}}(t) = 0 \implies M\tilde{A}\tilde{x}(t) = 0,$$

resulting in $\tilde{A}\tilde{x}(t) \in N(M)$. \square

Proof, Thm 4.1. Let $n \times 1$ vector $\phi \in N(M)$.

The observability matrices for the pairs $(\tilde{A} + \Delta M, C)$ and (\tilde{A}, C) are

$$\mathcal{O}_\Delta = \begin{bmatrix} C \\ C(\tilde{A} + \Delta M) \\ C(\tilde{A} + \Delta M)^2 \\ \vdots \\ C(\tilde{A} + \Delta M)^{n-1} \end{bmatrix}, \quad \mathcal{O}_{\tilde{A}} = \begin{bmatrix} C \\ C\tilde{A} \\ C\tilde{A}^2 \\ \vdots \\ C\tilde{A}^{n-1} \end{bmatrix}.$$

Clearly, the first rows of $\mathcal{O}_\Delta \phi$ and $\mathcal{O}_{\tilde{A}} \phi$ equal $C\phi$. The second row of $\mathcal{O}_\Delta \phi$,

$$C(\tilde{A} + \Delta M)\phi = C\tilde{A}\phi + C\Delta M\phi = C\tilde{A}\phi,$$

equals the second row of $\mathcal{O}_{\tilde{A}} \phi$ since $M\phi = 0$ from $\phi \in N(M)$. Similarly, the third row of $\mathcal{O}_\Delta \phi$,

$$\begin{aligned} C(\tilde{A} + \Delta M)^2\phi &= C(\tilde{A} + \Delta M)(\tilde{A} + \Delta M)\phi \\ &= C(\tilde{A} + \Delta M)(\tilde{A}\phi + \Delta M\phi) \\ &= C(\tilde{A} + \Delta M)\tilde{A}\phi \\ &= C\tilde{A}^2\phi + C\Delta M\tilde{A}\phi \\ &= C\tilde{A}^2\phi, \end{aligned}$$

equals the third row of $\mathcal{O}_{\tilde{A}} \phi$ since $M\tilde{A}\phi = 0$ from Theorem 4.2. Continuing this row-by-row comparison shows

$$\mathcal{O}_\Delta \phi = \begin{bmatrix} C \\ C(\tilde{A} + \Delta M) \\ C(\tilde{A} + \Delta M)^2 \\ \vdots \\ C(\tilde{A} + \Delta M)^{n-1} \end{bmatrix} \phi = \begin{bmatrix} C \\ C\tilde{A} \\ C\tilde{A}^2 \\ \vdots \\ C\tilde{A}^{n-1} \end{bmatrix} \phi = \mathcal{O}_{\tilde{A}} \phi.$$

Let $\psi \in N(M) \cap N(\mathcal{O}_\Delta)$. Knowing $\psi \in N(M) \implies \mathcal{O}_\Delta \psi = \mathcal{O}_{\tilde{A}} \psi$,

$$\begin{aligned} \psi \in N(M) \cap N(\mathcal{O}_\Delta) &\iff \psi \in N(M) \text{ and } \psi \in N(\mathcal{O}_\Delta) \\ &\iff M\psi = 0 \text{ and } \mathcal{O}_\Delta \psi = 0 \\ &\iff M\psi = 0 \text{ and } \mathcal{O}_{\tilde{A}} \psi = 0 \\ &\iff \psi \in N(M) \text{ and } \psi \in N(\mathcal{O}_{\tilde{A}}) \\ &\iff \psi \in N(M) \cap N(\mathcal{O}_{\tilde{A}}). \end{aligned}$$

Therefore, $N(M) \cap N(\mathcal{O}_\Delta) = N(M) \cap N(\mathcal{O}_{\tilde{A}})$. \square

Theorem 4.1 indicates changing the completion does not alter which matrix pencil eigenvalues are observable. Thus, given a completion of a linear time-invariant system of DAEs and an output equation, if at least one matrix pencil eigenvalue is unstable and unobservable, our observer construction approach will never be able to observe the system of DAEs with that particular output equation.

The following remark's theoretical development shows completions of a linear time-invariant system of DAEs may have observability matrices with different ranks. Keeping the results from Theorem 4.1 in mind, we know any rank variation is dependent on the additional dynamics.

Remark 4.3. If $N(M) + N(\mathcal{O}_{\tilde{A}}) = \Re^n$, then $N(\mathcal{O}_{\tilde{A}})$ may not equal $N(\mathcal{O}_\Delta)$. In particular, observability properties can vary between completions.

Established theory Theorem 4.4 is required for the theoretical development of Remark 4.3.

Theorem 4.4. $N(\mathcal{O}_{\tilde{A}})$ is \tilde{A} -invariant (commonly $N(\mathcal{O})$ is A -invariant, $\dot{x}(t) = Ax(t)$). [4]

Proof, Thm 4.4. Let $n \times 1$ vector $\tilde{x}(t) \in N(\mathcal{O}_{\tilde{A}})$. Then $\mathcal{O}_{\tilde{A}} \tilde{x}(t) = 0$ implies

$$C\tilde{x}(t) = 0, \quad C\tilde{A}\tilde{x}(t) = 0, \quad C\tilde{A}^2\tilde{x}(t) = 0, \quad \dots, \quad C\tilde{A}^{n-1}\tilde{x}(t) = 0.$$

By the Cayley-Hamilton Theorem, $\tilde{A}^n = (-1)^{n+1} (\alpha_0 I + \alpha_1 \tilde{A} + \alpha_2 \tilde{A}^2 + \dots + \alpha_{n-1} \tilde{A}^{n-1})$, so $\mathcal{O}_{\tilde{A}} \tilde{A}\tilde{x}(t) = 0$ since

$$\begin{aligned} C\tilde{A}\tilde{x}(t) &= 0, \quad C\tilde{A}^2\tilde{x}(t) = 0, \quad \dots, \quad C\tilde{A}^{n-1}\tilde{x}(t) = 0, \quad \text{and} \\ C\tilde{A}^n\tilde{x}(t) &= C((-1)^{n+1} (\alpha_0 I + \alpha_1 \tilde{A} + \alpha_2 \tilde{A}^2 + \dots + \alpha_{n-1} \tilde{A}^{n-1})) \tilde{x}(t) \\ &= (-1)^{n+1} (\alpha_0 C\tilde{x}(t) + \alpha_1 C\tilde{A}\tilde{x}(t) + \alpha_2 C\tilde{A}^2\tilde{x}(t) + \dots + \alpha_{n-1} C\tilde{A}^{n-1}\tilde{x}(t)) \\ &= 0. \end{aligned}$$

Therefore, $\tilde{A}\tilde{x}(t) \in N(\mathcal{O}_{\tilde{A}})$. \square

Theoretical Development, Remark 4.3. Given $N(M) + N(\mathcal{O}_{\tilde{A}}) = \Re^n$. Then $\nu_1 \oplus \nu_2 \oplus \nu_3 = \Re^n$, where $\nu_3 = N(M) \cap N(\mathcal{O}_{\tilde{A}})$, $\nu_1 \oplus \nu_3 = N(\mathcal{O}_{\tilde{A}})$, and $\nu_2 \oplus \nu_3 = N(M)$. Since $N(M)$ and $N(\mathcal{O}_{\tilde{A}})$ are \tilde{A} -invariant, ν_3 , $\nu_1 \oplus \nu_3$, and $\nu_2 \oplus \nu_3$ are also \tilde{A} -invariant. This information helps simplify the structures of matrices

$$\tilde{A} = \begin{bmatrix} \tilde{A}_{11} & \tilde{A}_{12} & \tilde{A}_{13} \\ \tilde{A}_{21} & \tilde{A}_{22} & \tilde{A}_{23} \\ \tilde{A}_{31} & \tilde{A}_{32} & \tilde{A}_{33} \end{bmatrix}, \quad C = \begin{bmatrix} C_1 & C_2 & C_3 \end{bmatrix}, \quad M = \begin{bmatrix} M_1 & M_2 & M_3 \end{bmatrix}.$$

The \tilde{A} -invariance of $\nu_1 \oplus \nu_3$ implies the general structure of $\tilde{A}\mu_1$, where $\tilde{A}\mu_1 \in \nu_1 \oplus \nu_3$ for all $\mu_1 \in \nu_1 \oplus \nu_3$, is

$$\begin{bmatrix} \tilde{A}_{11} & \tilde{A}_{12} & \tilde{A}_{13} \\ \tilde{A}_{21} & \tilde{A}_{22} & \tilde{A}_{23} \\ \tilde{A}_{31} & \tilde{A}_{32} & \tilde{A}_{33} \end{bmatrix} \begin{bmatrix} * \\ 0 \\ * \end{bmatrix} = \begin{bmatrix} \xi \\ 0 \\ \xi \end{bmatrix}.$$

Placeholders $*$ and ξ simply represent where a nonzero entry could be. From this product, $\tilde{A}_{21}* + \tilde{A}_{23}* = 0$ for any μ_1 when $\tilde{A}_{21} = 0$ and $\tilde{A}_{23} = 0$. The \tilde{A} -invariance of $\nu_2 \oplus \nu_3$ implies the general structure of $\tilde{A}\mu_2$, where $\tilde{A}\mu_2 \in \nu_2 \oplus \nu_3$ for all $\mu_2 \in \nu_2 \oplus \nu_3$, is

$$\begin{bmatrix} \tilde{A}_{11} & \tilde{A}_{12} & \tilde{A}_{13} \\ \tilde{A}_{21} & \tilde{A}_{22} & \tilde{A}_{23} \\ \tilde{A}_{31} & \tilde{A}_{32} & \tilde{A}_{33} \end{bmatrix} \begin{bmatrix} 0 \\ * \\ * \end{bmatrix} = \begin{bmatrix} 0 \\ \xi \\ \xi \end{bmatrix}.$$

Thus, for any μ_2 , the linear combination $\tilde{A}_{12}* + \tilde{A}_{13}* = 0$ indicates $\tilde{A}_{12} = 0$ and $\tilde{A}_{13} = 0$. Without any additional information provided by the \tilde{A} -invariance of ν_3 , the simplified structure of matrix \tilde{A} is

$$\tilde{A}_s = \begin{bmatrix} \tilde{A}_{11} & 0 & 0 \\ 0 & \tilde{A}_{22} & 0 \\ \tilde{A}_{31} & \tilde{A}_{32} & \tilde{A}_{33} \end{bmatrix}.$$

Since $\mu_1 \in N(\mathcal{O}_{\tilde{A}})$ from $\mu_1 \in \nu_1 \oplus \nu_3$,

$$C\mu_1 = \begin{bmatrix} C_1 & C_2 & C_3 \end{bmatrix} \begin{bmatrix} * \\ 0 \\ * \end{bmatrix} = 0$$

reveals $C_1* + C_3* = 0$ always holds when $C_1 = 0$ and $C_3 = 0$, making $C_s = \begin{bmatrix} 0 & C_2 & 0 \end{bmatrix}$ the

simplified structure of matrix C . Similarly, since $\mu_2 \in N(M)$ from $\mu_2 \in \nu_2 \oplus \nu_3$,

$$M\mu_2 = \begin{bmatrix} M_1 & M_2 & M_3 \end{bmatrix} \begin{bmatrix} 0 \\ * \\ * \end{bmatrix} = 0$$

indicates $M_2* + M_3* = 0$ is true for any μ_2 when $M_2 = 0$ and $M_3 = 0$. Thus, $M_s = \begin{bmatrix} M_1 & 0 & 0 \end{bmatrix}$ is the simplified structure of matrix M .

Again consider $\mu_1 \in N(\mathcal{O}_{\tilde{A}})$. Although $C_s \tilde{A}_s \mu_1 = 0$,

$$C_s \tilde{A}_s \mu_1 = \begin{bmatrix} 0 & C_2 & 0 \end{bmatrix} \begin{bmatrix} \tilde{A}_{11} & 0 & 0 \\ 0 & \tilde{A}_{22} & 0 \\ \tilde{A}_{31} & \tilde{A}_{32} & \tilde{A}_{33} \end{bmatrix} \begin{bmatrix} * \\ 0 \\ * \end{bmatrix} = \begin{bmatrix} 0 & C_2 \tilde{A}_{22} & 0 \end{bmatrix} \begin{bmatrix} * \\ 0 \\ * \end{bmatrix} = 0,$$

$C_s (\tilde{A}_s + \Delta M_s) \mu_1$ may not equal zero. Using

$$\begin{aligned} \tilde{A}_s + \Delta M_s &= \begin{bmatrix} \tilde{A}_{11} & 0 & 0 \\ 0 & \tilde{A}_{22} & 0 \\ \tilde{A}_{31} & \tilde{A}_{32} & \tilde{A}_{33} \end{bmatrix} + \begin{bmatrix} \Delta_1 \\ \Delta_2 \\ \Delta_3 \end{bmatrix} \begin{bmatrix} M_1 & 0 & 0 \end{bmatrix} \\ &= \begin{bmatrix} \tilde{A}_{11} + \Delta_1 M_1 & 0 & 0 \\ \Delta_2 M_1 & \tilde{A}_{22} & 0 \\ \tilde{A}_{31} + \Delta_3 M_1 & \tilde{A}_{32} & \tilde{A}_{33} \end{bmatrix}, \end{aligned}$$

the product

$$\begin{aligned} C_s (\tilde{A}_s + \Delta M_s) \mu_1 &= \begin{bmatrix} 0 & C_2 & 0 \end{bmatrix} \begin{bmatrix} \tilde{A}_{11} + \Delta_1 M_1 & 0 & 0 \\ \Delta_2 M_1 & \tilde{A}_{22} & 0 \\ \tilde{A}_{31} + \Delta_3 M_1 & \tilde{A}_{32} & \tilde{A}_{33} \end{bmatrix} \begin{bmatrix} * \\ 0 \\ * \end{bmatrix} \\ &= \begin{bmatrix} C_2 \Delta_2 M_1 & C_2 \tilde{A}_{22} & 0 \end{bmatrix} \begin{bmatrix} * \\ 0 \\ * \end{bmatrix} \\ &= C_2 \Delta_2 M_1 * \end{aligned}$$

does not have to equal zero. Therefore, μ_1 may not be an element of $N(\mathcal{O}_\Delta)$, which means $N(\mathcal{O}_{\tilde{A}})$ may not equal $N(\mathcal{O}_\Delta)$. Additionally, in the lower triangular matrix $\tilde{A}_s + \Delta M_s$, the choice of Δ , particularly Δ_1 , has the potential to make the rank of \mathcal{O}_Δ different from the rank of $\mathcal{O}_{\tilde{A}}$. Thus, observability properties can vary between completions. \square

4.2 Full-order and Reduced-order Observers

Proofs of established theory are sometimes difficult to locate. As a result, detailed proofs of theory not original to our research but important to example analysis are included in this section. An original version of Proof Part 1, Theorem 4.5 can be found in [88], and a different but related proof of Theorem 4.6 can be found in [158].

Consider a linear time-invariant system of ODEs with output equation

$$\begin{aligned}\dot{x}(t) &= Ax(t) + Bu(t) \\ y(t) &= Cx(t),\end{aligned}$$

where matrix A is $n \times n$ and $m \times n$ output matrix C is assumed full row rank. This system does not need to be a completion, so general notation A rather than completion notation \tilde{A} is used throughout this section. For a specific example, if the full row rank condition is not met, that is $\text{rank}(C) = r < m$, then without losing any information provided by the output equation, r linearly independent rows of C can be taken to define a full row rank output matrix. An additional assumption for the following proofs is output matrix C has structure $\begin{bmatrix} I & 0 \end{bmatrix}$. If the output matrix does not have this block form, then the transformation discussed in Section 3.3.1 for constructing the reduced-order observer obtains the desired structure. Matrix A is considered in terms of the block form $\begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}$, where submatrix A_{11} is $m \times m$, submatrix A_{12} is $m \times (n - m)$, submatrix A_{21} is $(n - m) \times m$, and submatrix A_{22} is $(n - m) \times (n - m)$.

Theorems 4.5 and 4.7 reveal that for a linear time-invariant system of ODEs, the unobservable eigenvalues from the full-order (pair (A, C)) and the reduced-order (pair (A_{22}, A_{12})) systems are equivalent. That is, the full-order observer can be constructed to estimate the state if and only if the reduced-order observer can be constructed to estimate the state.

Theorem 4.5. *Pair (A, C) is observable iff pair (A_{22}, A_{12}) is observable.*

Proof Part 1, Thm 4.5. \Rightarrow Given pair (A, C) is observable. Then its observability matrix

$$\mathcal{O}_F = \begin{bmatrix} C \\ CA \\ CA^2 \\ \vdots \\ CA^{n-1} \end{bmatrix}$$

has full column rank ($\text{rank}(\mathcal{O}_F) = n$). In order for pair (A_{22}, A_{12}) to be observable, we must

show its observability matrix

$$\mathcal{O}_R = \begin{bmatrix} A_{12} \\ A_{12}A_{22} \\ A_{12}A_{22}^2 \\ \vdots \\ A_{12}A_{22}^{(n-m)-1} \end{bmatrix}$$

has rank $n - m$. Substituting into \mathcal{O}_F for A and C reveals the structure

$$\mathcal{O}_F = \begin{bmatrix} \begin{bmatrix} I & 0 \end{bmatrix} \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \\ \begin{bmatrix} I & 0 \end{bmatrix} \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}^2 \\ \vdots \\ \begin{bmatrix} I & 0 \end{bmatrix} \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}^{n-1} \end{bmatrix} = \begin{bmatrix} I & 0 \\ A_{11} & A_{12} \\ A_{11}^2 + A_{12}A_{21} & A_{11}A_{12} + A_{12}A_{22} \\ \vdots & \vdots \\ * & A_{11}^{n-2}A_{12} + \dots + A_{12}A_{22}^{n-2} \end{bmatrix}.$$

Due to the block form of output matrix C , the resulting structure of observability matrix \mathcal{O}_F has two (block) columns. Terms $\{A_{12}, A_{12}A_{22}, A_{12}A_{22}^2, \dots, A_{12}A_{22}^{(n-m)-1}\}$ associated with the observability matrix for pair (A_{22}, A_{12}) are contained within column 2 of \mathcal{O}_F . The $m \times m$ identity matrix in the first (block) row of \mathcal{O}_F 's column 1 indicates column 1's rank is m and column 2's rank is $n - m$. The \mathcal{O}_R terms can be isolated in \mathcal{O}_F 's column 2 by finding a matrix equivalent to observability matrix \mathcal{O}_F through row multiplication and row addition. Although row operations affect both columns, the focus on column 2's results permits the following simplified appearance of observability matrix \mathcal{O}_F :

$$\mathcal{O}_F = \begin{bmatrix} I & 0 \\ A_{11} & A_{12} \\ * & A_{11}A_{12} + A_{12}A_{22} \\ * & (A_{11}^2 + A_{12}A_{21})A_{12} + A_{11}A_{12}A_{22} + A_{12}A_{22}^2 \\ \vdots & \vdots \\ * & A_{11}^{n-2}A_{12} + \dots + A_{12}A_{22}^{n-2} \end{bmatrix}, \quad (4.1)$$

where * represents an entry in column 1 affected by row operations but irrelevant to the rank

determination of \mathcal{O}_R . The additional row resulting from $\begin{bmatrix} I & 0 \end{bmatrix} \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}^3$ is included in (4.1) for explanation purposes.

Row 2, column 2 of (4.1) contains term A_{12} from row 1 of observability matrix \mathcal{O}_R without implementing any row operations. Reducing row 3, column 2 of (4.1) to term $A_{12}A_{22}$ requires row multiplication and addition, $(-A_{11} \times \text{row 2}) + \text{row 3}$, to eliminate $A_{11}A_{12}$ from the sum. In matrix form, these two row operations appear as the left-multiplier nonsingular matrix

$$\left[\begin{array}{ccc|c} I_{m \times m} & 0_{m \times m} & 0_{m \times m} & 0_{(3m) \times (m(n-3))} \\ 0_{m \times m} & I_{m \times m} & 0_{m \times m} & 0_{(3m) \times (m(n-3))} \\ 0_{m \times m} & -A_{11} & I_{m \times m} & 0_{(m(n-3)) \times (3m)} \\ \hline & & & I_{(m(n-3)) \times (m(n-3))} \end{array} \right].$$

If $(-(A_{11}^2 + A_{12}A_{21}) \times \text{row 2})$ and $(-A_{11} \times \text{reduced row 3})$ are added to row 4, then row 4, column 2 of (4.1) reduces to term $A_{12}A_{22}^2$. Notice row i , $i = 2, \dots, n$, of column 2 in \mathcal{O}_F equals

$$\sum_{j=2}^i M_{ij} A_{12} A_{22}^{i-j},$$

where M_{ij} represents the coefficient matrix of $A_{12}A_{22}^{i-j}$ ($M_{i2} = I$). Once rows 3 through $k-1 < n$ in column 2 have been reduced to their respective desired terms $\{A_{12}A_{22}, \dots, A_{12}A_{22}^{k-3}\}$, appropriate row operations involving row 2 and these reduced rows help reduce row k , column 2 of observability matrix \mathcal{O}_F to desired term $A_{12}A_{22}^{k-2}$.

The reduction process continues until observability matrix \mathcal{O}_F has structure

$$\left[\begin{array}{cc} I & 0 \\ * & A_{12} \\ * & A_{12}A_{22} \\ \vdots & \vdots \\ * & A_{12}A_{22}^{(n-m)-1} \\ * & A_{12}A_{22}^{n-m} \\ \vdots & \vdots \\ * & A_{12}A_{22}^{n-2} \end{array} \right] = \left[\begin{array}{cc} I & 0 \\ * & \mathcal{O}_R \\ * & A_{12}A_{22}^{n-m} \\ \vdots & \vdots \\ * & A_{12}A_{22}^{n-2} \end{array} \right].$$

If $m = 1$, then $A_{12}A_{22}^{(n-m)-1} = A_{12}A_{22}^{n-2}$ reveals $\text{rank}(\mathcal{O}_R) = n - m$ since all of reduced \mathcal{O}_F 's nonzero rows in column 2 form observability matrix \mathcal{O}_R . Otherwise, powers greater than $(n-m)-1$ of $(n-m) \times (n-m)$ submatrix A_{22} can be written as linear combinations of terms

$\{I, A_{22}, A_{22}^2, \dots, A_{22}^{(n-m)-1}\}$ by the Cayley-Hamilton Theorem; for example,

$$A_{22}^{n-m} = (-1)^{(n-m)+1} \left(\alpha_0 I + \alpha_1 A_{22} + \alpha_2 A_{22}^2 + \dots + \alpha_{(n-m)-1} A_{22}^{(n-m)-1} \right).$$

This linear dependence means any reduced row containing a non- \mathcal{O}_R term does not affect the rank of \mathcal{O}_F 's column 2. Therefore, the observability matrix \mathcal{O}_R has full column rank ($\text{rank}(\mathcal{O}_R) = n - m$), so pair (A_{22}, A_{12}) is observable. \square

Proof Part 2, Thm 4.5. \Leftarrow Given pair (A_{22}, A_{12}) is observable. Then observability matrix \mathcal{O}_R has full column rank. The reduction process outlined in Part 1 of this proof is independent of \mathcal{O}_F 's rank. Thus, since an $m \times m$ identity matrix and \mathcal{O}_R are contained in columns 1 and 2 of reduced \mathcal{O}_F , respectively, the rank of \mathcal{O}_F equals $\text{rank}(I) + \text{rank}(\mathcal{O}_R) = m + (n - m) = n$. Therefore, observability matrix \mathcal{O}_F has full column rank, so pair (A, C) is observable. \square

Theorem 4.6. *Pair (A, C) is detectable iff pair (A_{22}, A_{12}) is detectable.*

In order to prove Theorem 4.6, we first need to show pairs (A, C) and (A_{22}, A_{12}) have the same unobservable eigenvalues. Then if either the full-order or the reduced-order system has stable, unobservable eigenvalues, the other system is detectable as well.

Theorem 4.7. *The unobservable eigenvalues of matrices A and A_{22} are equivalent.*

Proof Part 1, Thm 4.7. \Rightarrow Pair (A, C) is not observable. Let (λ_f, v_f) be an eigenpair of matrix A with λ_f an unobservable eigenvalue. Then $v_f \in N(\mathcal{O}_F)$, or $\mathcal{O}_F v_f = 0$, from which

$$Cv_f = 0 \implies \begin{bmatrix} I & 0 \end{bmatrix} v_f = 0.$$

The product $\begin{bmatrix} I & 0 \end{bmatrix} v_f$,

$$\begin{bmatrix} I & 0 \end{bmatrix} v_f = \begin{bmatrix} I & 0 \end{bmatrix} \begin{bmatrix} [v_1]_{m \times 1} \\ [v_2]_{(n-m) \times 1} \end{bmatrix} = v_1,$$

equals an $m \times 1$ zero vector if eigenvector v_f has structure $\begin{bmatrix} 0 \\ v_2 \end{bmatrix}$. Then $Av_f = \lambda_f v_f$ implies

$$\begin{aligned} (\lambda_f I_{n \times n} - A)v_f = 0 &\implies \left(\begin{bmatrix} [\lambda_f]_{m \times m} & 0 \\ 0 & [\lambda_f]_{(n-m) \times (n-m)} \end{bmatrix} - \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \right) \begin{bmatrix} 0 \\ v_2 \end{bmatrix} = 0 \\ &\implies \begin{bmatrix} \lambda_f - A_{11} & -A_{12} \\ -A_{21} & \lambda_f - A_{22} \end{bmatrix} \begin{bmatrix} 0 \\ v_2 \end{bmatrix} = 0 \\ &\implies \begin{bmatrix} -A_{12}v_2 \\ (\lambda_f - A_{22})v_2 \end{bmatrix} = 0. \end{aligned}$$

Thus, $A_{12}v_2 = 0$ and $A_{22}v_2 = \lambda_f v_2$, and since

$$\begin{aligned} A_{12}(A_{22}v_2) &= A_{12}(\lambda_f v_2) = \lambda_f A_{12}v_2 = 0 \\ A_{12}(A_{22}^2 v_2) &= A_{12}(\lambda_f^2 v_2) = \lambda_f^2 A_{12}v_2 = 0 \\ &\vdots \\ A_{12}\left(A_{22}^{(n-m)-1} v_2\right) &= A_{12}\left(\lambda_f^{(n-m)-1} v_2\right) = \lambda_f^{(n-m)-1} A_{12}v_2 = 0, \end{aligned}$$

then

$$0 = \begin{bmatrix} A_{12} \\ A_{12}A_{22} \\ A_{12}A_{22}^2 \\ \vdots \\ A_{12}A_{22}^{(n-m)-1} \end{bmatrix} v_2 = \mathcal{O}_R v_2.$$

Therefore, $v_2 \in N(\mathcal{O}_R)$, so λ_f of eigenpair (λ_f, v_2) is an unobservable eigenvalue of matrix A_{22} . The unobservable eigenvalues of A are also unobservable eigenvalues of A_{22} . \square

Proof Part 2, Thm 4.7. \Leftarrow Pair (A_{22}, A_{12}) is not observable. Let (λ_r, v_r) be an eigenpair of matrix A_{22} with λ_r an unobservable eigenvalue. Then $v_r \in N(\mathcal{O}_R)$, or $\mathcal{O}_R v_r = 0$. Thus,

$$\begin{aligned} A_{12}v_r &= 0 \implies -A_{12}v_r = 0, \\ A_{22}v_r &= \lambda_r v_r \implies (\lambda_r I_{(n-m) \times (n-m)} - A_{22}) v_r = 0, \end{aligned}$$

such that

$$\begin{bmatrix} -A_{12}v_r \\ (\lambda_r - A_{22})v_r \end{bmatrix} = 0.$$

Then

$$\begin{aligned} \begin{bmatrix} -A_{12}v_r \\ (\lambda_r - A_{22})v_r \end{bmatrix} = 0 &\implies \begin{bmatrix} \lambda_r - A_{11} & -A_{12} \\ -A_{21} & \lambda_r - A_{22} \end{bmatrix} \begin{bmatrix} 0 \\ v_r \end{bmatrix} = 0 \\ &\implies \left(\begin{bmatrix} [\lambda_r]_{m \times m} & 0 \\ 0 & [\lambda_r]_{(n-m) \times (n-m)} \end{bmatrix} - \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \right) \begin{bmatrix} 0 \\ v_r \end{bmatrix} = 0 \\ &\implies (\lambda_r I_{n \times n} - A)v_+ = 0, \end{aligned}$$

where $v_+ = \begin{bmatrix} 0 \\ v_r \end{bmatrix}$. Since $Av_+ = \lambda_r v_+$, (λ_r, v_+) is an eigenpair for A .

Also, $Cv_+ = \begin{bmatrix} I & 0 \end{bmatrix} \begin{bmatrix} 0 \\ v_r \end{bmatrix} = 0$ implies

$$\begin{aligned} C(Av_+) &= C(\lambda_r v_+) = \lambda_r Cv_+ = 0 \\ C(A^2 v_+) &= C(\lambda_r^2 v_+) = \lambda_r^2 Cv_+ = 0 \\ &\vdots \\ C(A^{n-1} v_+) &= C(\lambda_r^{n-1} v_+) = \lambda_r^{n-1} Cv_+ = 0. \end{aligned}$$

Hence,

$$0 = \begin{bmatrix} C \\ CA \\ CA^2 \\ \vdots \\ CA^{n-1} \end{bmatrix} v_+ = \mathcal{O}_F v_+.$$

Therefore, $v_+ \in N(\mathcal{O}_F)$, so λ_r of eigenpair (λ_r, v_+) is an unobservable eigenvalue of A . The unobservable eigenvalues of A_{22} are also unobservable eigenvalues of A . \square

Proof, Thm 4.6. Pair (A, C) is detectable.

\iff The unobservable eigenvalues of pair (A, C) are stable.

\iff The unobservable eigenvalues of matrix A and matrix A_{22} are equivalent by Theorem 4.7.

\iff The unobservable eigenvalues of pair (A_{22}, A_{12}) are stable.

\iff Pair (A_{22}, A_{12}) is detectable. \square

4.3 Comments on Linear Time-Varying Theory

Additional research on the completions of a linear time-varying system of DAEs and their observability is required before drawing theoretical conclusions similar to those in Section 4.1.

The theory if pair $(A(t), C(t))$ is observable, then $(A_{22}(t), A_{12}(t))$ is observable is proven in [135] by first considering the pairs at a specific value of time t and then continuing with a rank-dependent linear time-invariant proof. Our suggested time-varying explanation for this relationship and its reverse implication focuses on alternative descriptions of observable:

1. a linear system of ODEs is observable at t_0 if its state $x(t_0) = 0$ is the only unobservable state at t_0 ;
2. if the output $y(t) = C(t)\Phi(t, t_0)x(t_0) = 0$ for every $t \geq t_0$ when $u(t) = 0$, then $x(t_0)$ is unobservable ($\Phi(t, t_0)$ is the state transition matrix of the linear system of ODEs). [4]

Thus, a linear system of ODEs can be shown to be observable at t_0 if $x(t_0) = 0$ is implied when $u(t) = 0$ and $y(t) = C(t)\Phi(t, t_0)x(t_0) = 0$ for $t \geq t_0$.

Again assuming $C(t) = \begin{bmatrix} I & 0 \end{bmatrix}$ with an $m \times m$ identity matrix, the full-order system $\dot{x}(t) = A(t)x(t) + B(t)u(t)$ with output equation $y(t) = C(t)x(t)$ can be written as

$$\dot{x}_1(t) = A_{11}(t)x_1(t) + A_{12}(t)x_2(t) + B_1(t)u(t) \quad (4.2a)$$

$$\dot{x}_2(t) = A_{21}(t)x_1(t) + A_{22}(t)x_2(t) + B_2(t)u(t) \quad (4.2b)$$

$$y(t) = x_1(t). \quad (4.2c)$$

System (4.2)'s associated reduced-order system is

$$\dot{x}_2(t) = A_{22}(t)x_2(t) + \bar{u}(t) \quad (4.3a)$$

$$\bar{y}(t) = A_{12}(t)x_2(t), \quad (4.3b)$$

where available information on $x_1(t)$ from output (4.2c) is used to define new output $\bar{y}(t) = \dot{x}_1(t) - A_{11}(t)x_1(t) - B_1(t)u(t)$ and new input $\bar{u}(t) = A_{21}(t)x_1(t) + B_2(t)u(t)$.

Given system (4.2) is observable, let $\bar{u}(t) = 0$ and $\bar{y}(t) = A_{12}(t)\Phi_R(t, t_0)x_2(t_0) = 0$ for $t \geq t_0$, where $x_2(t_0)$ is arbitrary and $\Phi_R(t, t_0)$ is the state transition matrix of system (4.3). The reduced-order system simplifies to

$$\dot{x}_2(t) = A_{22}(t)x_2(t) \quad (4.4a)$$

$$0 = A_{12}(t)x_2(t). \quad (4.4b)$$

However, for some $\bar{x}(t) = 0$, system (4.4) also has structure

$$\dot{\bar{x}}(t) = A_{11}(t)\bar{x}(t) + A_{12}(t)x_2(t) \quad (4.5a)$$

$$\dot{x}_2(t) = A_{21}(t)\bar{x}(t) + A_{22}(t)x_2(t) \quad (4.5b)$$

$$y(t) = \bar{x}(t) = 0, \quad (4.5c)$$

which is equivalent to system (4.2) when $u(t) = 0$ and $y(t) = x_1(t) = 0$. Thus, $x_2(t_0) = 0$ from $x(t_0) = \begin{bmatrix} x_1(t_0) \\ x_2(t_0) \end{bmatrix} = 0$ since the full-order system is observable. Therefore, $\bar{u}(t) = 0$ and $\bar{y}(t) = A_{12}(t)\Phi_R(t, t_0)x_2(t_0) = 0$ for $t \geq t_0$ implies $x_2(t_0) = 0$, so the reduced-order system is observable at t_0 . This observability check holds for all $t \geq t_0$, resulting in pair $(A_{22}(t), A_{12}(t))$ being observable.

Instead, given system (4.3) is observable, let $u(t) = 0$ and $y(t) = C(t)\Phi(t, t_0)x(t_0) = 0$ for $t \geq t_0$, where $x_2(t_0)$ from $x(t_0) = \begin{bmatrix} x_1(t_0) \\ x_2(t_0) \end{bmatrix}$ is arbitrary and $\Phi(t, t_0)$ is the state transition matrix of system (4.2). The initial condition for $x_1(t_0)$ is not arbitrary since the assumption $y(t) = 0$ implies $x_1(t) = 0$, so $x_1(t_0) = 0$. But when $y(t) = x_1(t) = 0$, the full-order system simplifies to system (4.4), which is equivalent to system (4.3) when $\bar{u}(t) = 0$ and $\bar{y}(t) = 0$. Thus, $x_2(t_0) = 0$ since the reduced-order system is observable. Therefore, $u(t) = 0$ and $y(t) = C(t)\Phi(t, t_0)x(t_0) = 0$ for $t \geq t_0$ implies $x(t_0) = 0$, so the full-order system is observable at t_0 . With the observability check holding for all $t \geq t_0$, pair $(A(t), C(t))$ is observable.

Although there is no reason to suspect the detectability implication between the full-order and reduced-order observers does not also hold in the time-varying case, it is harder to justify using the eigenpair-dependent linear time-invariant proof at a specific value of time t when eigenvalues are not used to show linear time-varying observability or detectability. Instead, a topic from control theory can be used to reason why the detectability implication holds. The zero dynamics of a system of ODEs are its solutions that produce a zero output [17], [96], making these states indistinguishable from the zero solution. Thus, the zero dynamics are unobservable. From this theory, the basis vectors for the unobservable subspace have structure $b_f = \begin{bmatrix} 0_{(n-m) \times 1} \\ b_{f2} \end{bmatrix}$ for the full-order system and $b_r = \alpha b_{f2}$, α some multiple, for the reduced-order system. Due to these basis vectors' structural relationship, a detectable full-order or reduced-order system implies its counterpart is also detectable. For a computational example, the basis vectors for the reduced-order system's $N(W_o(t))$ can be extended to become $\begin{bmatrix} 0_{(n-m) \times 1} \\ b_r \end{bmatrix} = \begin{bmatrix} 0_{(n-m) \times 1} \\ \alpha b_{f2} \end{bmatrix}$ and then compared with the full-order system's basis vectors for $N(W_o(t))$ through rank checks to confirm the connection between the unobservable subspaces.

Chapter 5

Linear Time-Invariant Example

5.1 Mechanics Example Introduction

The first implementation of our observer construction approach was on a linear time-invariant system of DAEs. The example system

$$\dot{x}_1(t) = x_2(t) \quad (5.1a)$$

$$\dot{x}_2(t) = Kx_1(t) + Sx_2(t) + H^T x_3(t) + Gu_1(t) \quad (5.1b)$$

$$0 = Hx_1(t) + u_2(t) \quad (5.1c)$$

is representative of a constrained mechanical system [24], [25]. State variables $x_1(t)$ and $x_2(t)$ are position and velocity, respectively, while equation (5.1c) is a physical constraint that produces the force $H^T x_3(t)$. The input $u(t)$ applies a force through $Gu_1(t)$ in equation (5.1b) and affects constraint (5.1c) through $u_2(t)$. Additionally, matrix H is assumed full row rank.

The example system has a special structure, allowing its index to be determined without specifying K , S , or G . A linear or nonlinear system of DAEs in Hessenberg form of size k ,

$$\begin{aligned} \dot{f}_1 &= F_1(f_1, f_2, \dots, f_k, t) \\ \dot{f}_2 &= F_2(f_1, f_2, \dots, f_{k-1}, t) \\ &\vdots \\ \dot{f}_i &= F_i(f_{i-1}, f_i, \dots, f_{k-1}, t), \quad 3 \leq i \leq k-1, \\ &\vdots \\ \text{and } 0 &= F_k(f_{k-1}, t), \end{aligned}$$

has index k if the product $(\partial F_k / \partial f_{k-1}) (\partial F_{k-1} / \partial f_{k-2}) \cdots (\partial F_2 / \partial f_1) (\partial F_1 / \partial f_k)$ is nonsingular

[28]. For $f_1 = x_2(t)$, $f_2 = x_1(t)$, and $f_3 = x_3(t)$, system (5.1) is in Hessenberg form of size 3:

$$\begin{aligned}\dot{f}_1(t) &= F_1(f_1, f_2, f_3, t) \longleftrightarrow \dot{x}_2(t) = Kx_1(t) + Sx_2(t) + H^T x_3(t) + Gu_1(t) \\ \dot{f}_2(t) &= F_2(f_1, f_2, t) \longleftrightarrow \dot{x}_1(t) = x_2(t) \\ 0 &= F_3(f_2, t) \longleftrightarrow 0 = Hx_1(t) + u_2(t).\end{aligned}$$

Product $(\partial F_3 / \partial f_2)(\partial F_2 / \partial f_1)(\partial F_1 / \partial f_3) = HIH^T$ is nonsingular from the full row rank assumption on H , so the example system is an index 3 linear time-invariant system of DAEs.

Without the rank assumption, matrix H would need to be specified before determining whether or not HIH^T is nonsingular. Furthermore, if the Hessenberg form of system (5.1) had not been recognized, an algorithm based on Definition 1.2 could compute the index after selecting matrices for a specific example. This algorithm assumes a value of i is the index of the linear time-invariant system of DAEs; uses i to construct matrices \mathcal{E} , \mathcal{F} from $\mathcal{E}w(t) + \mathcal{F}x(t) = \mathcal{B}v(t)$, the derivative array equations found without using stabilized differentiation; and then checks that $[\mathcal{E} \quad \mathcal{F}]$ is full row rank and that the first n entries ($x(t)$ is $n \times 1$) of $N(\mathcal{E})$'s basis vectors are zero. When both checks are true, index k is defined as i . Otherwise, using a for loop, the algorithm repeats with the next value $i + 1$ assumed as the index of the linear time-invariant system of DAEs. Since the example is linear time-invariant, the constant rank requirement in Definition 1.2 is automatic. Our numerical algorithm for this index check is not example specific and is included in Appendix B as MATLAB program index.m.

Due to this system's Hessenberg form and the row rank assumption on H , the solution of system (5.1) can be determined without a completion. Being able to calculate $x(t)$ permits an assessment of the completion's solution $\tilde{x}(t)$ and of the observer's estimate using estimation error $e(t) = x(t) - \hat{x}(t)$ rather than $e(t) = \tilde{x}(t) - \hat{x}(t)$. Differentiating equation (5.1c) once and substituting equation (5.1a) in for $\dot{x}_1(t)$ produces

$$0 = Hx_2(t) + \dot{u}_2(t). \quad (5.2)$$

After equation (5.2) is differentiated once and equation (5.1b) is substituted in for $\dot{x}_2(t)$,

$$0 = HKx_1(t) + HSx_2(t) + HH^T x_3(t) + HGu_1(t) + \ddot{u}_2(t),$$

state variable $x_3(t)$ is solved for explicitly since HH^T is invertible (row rank assumption):

$$x_3(t) = - (HH^T)^{-1} (HKx_1(t) + HSx_2(t) + HGu_1(t) + \ddot{u}_2(t)).$$

Once this equation for $x_3(t)$ is substituted back into equation (5.1b),

$$\begin{aligned}\dot{x}_2(t) &= Kx_1(t) + Sx_2(t) \\ &\quad + H^T \left(- (HH^T)^{-1} (HKx_1(t) + HSx_2(t) + HGu_1(t) + \ddot{u}_2(t)) \right) + Gu_1(t),\end{aligned}$$

simplification using $H^\dagger = H^T (HH^T)^{-1}$, the Moore-Penrose pseudoinverse of H [128], and $P_H = (I - H^\dagger H)$ reveals

$$\begin{aligned}\dot{x}_2(t) &= Kx_1(t) + Sx_2(t) - H^\dagger(HKx_1(t) + HSx_2(t) + HGu_1(t) + \ddot{u}_2(t)) + Gu_1(t) \\ &= (I - H^\dagger H)Kx_1(t) + (I - H^\dagger H)Sx_2(t) + (I - H^\dagger H)Gu_1(t) - H^\dagger\ddot{u}_2(t) \\ &= P_HKx_1(t) + P_HSx_2(t) + P_HGu_1(t) - H^\dagger\ddot{u}_2(t).\end{aligned}$$

System (5.1) becomes

$$\dot{x}_1(t) = x_2(t) \quad (5.3a)$$

$$\dot{x}_2(t) = P_HKx_1(t) + P_HSx_2(t) + P_HGu_1(t) - H^\dagger\ddot{u}_2(t) \quad (5.3b)$$

$$x_3(t) = -(HH^T)^{-1}(HKx_1(t) + HSx_2(t) + HGu_1(t) + \ddot{u}_2(t)) \quad (5.3c)$$

$$0 = Hx_1(t) + u_2(t) \quad (5.3d)$$

$$0 = Hx_2(t) + \dot{u}_2(t), \quad (5.3e)$$

where the solution of state variable $x_3(t)$ can be found separately from $x_1(t)$ and $x_2(t)$ (for example, outside of a function called by one of MATLAB's solvers) and a consistent initial condition is dependent on constraints (5.3d) and (5.3e).

5.2 Observer Implementation

System (5.1) is a general representation of a constrained mechanical system. In order to illustrate our observer construction approach, we specify K , S , H , and G to generate

$$\dot{x}_1(t) = x_2(t) \quad (5.4a)$$

$$\dot{x}_2(t) = \begin{bmatrix} -2 & 1 \\ 1 & -2 \end{bmatrix}x_1(t) + \frac{1}{4}\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}x_2(t) + \begin{bmatrix} 1 \\ -1 \end{bmatrix}x_3(t) + \begin{bmatrix} 1 \\ 1 \end{bmatrix}u_1(t) \quad (5.4b)$$

$$0 = \begin{bmatrix} 1 & -1 \end{bmatrix}x_1(t) + u_2(t) \quad (5.4c)$$

with input $u(t) = \begin{bmatrix} u_1(t) \\ u_2(t) \end{bmatrix} = \begin{bmatrix} \sin(t) \\ \sin(t) \end{bmatrix}$. Notice, $x_1(t)$ is 2×1 , $x_2(t)$ is 2×1 , and $x_3(t)$ is 1×1 . Program index.m confirms system (5.4) is index 3. From the common description of a linear time-invariant system of DAEs $E\dot{x}(t) + Fx(t) = Bu(t)$,

$$E = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \quad F = \begin{bmatrix} 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 \\ 2 & -1 & -\frac{1}{4} & 0 & -1 \\ -1 & 2 & 0 & -\frac{1}{4} & 1 \\ -1 & 1 & 0 & 0 & 0 \end{bmatrix}, \quad B = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}.$$

Three output equations are tested with system (5.4). The output matrices considered are

$$C_a = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}, \quad C_b = \begin{bmatrix} 0 & 0 & 1 & -1 & 0 \end{bmatrix}, \quad C_c = \begin{bmatrix} 0 & 0 & 1 & 1 & 0 \end{bmatrix}.$$

Based on a method described in Section 6.1 from [54] for checking the observability of a linear system of DAEs, system (5.4) is observable with output matrices C_a and C_c but not with output matrix C_b . Although an observable system of DAEs is not a requirement for our observer construction approach, this check helps corroborate later results.

The initial conditions for solving system (5.4), finding the completions, and constructing the observers are defined at time $t_0 = 0$. From constraints (5.3d) and (5.3e) and equation (5.3c),

$$\begin{aligned} x_1(0) &= -H^\dagger u_2(0) \\ &= -\begin{bmatrix} 0.5 \\ -0.5 \end{bmatrix} \sin(0) = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \\ x_2(0) &= -H^\dagger \dot{u}_2(0) \\ &= -\begin{bmatrix} 0.5 \\ -0.5 \end{bmatrix} \cos(0) = \begin{bmatrix} -0.5 \\ 0.5 \end{bmatrix} \\ x_3(0) &= -(HH^T)^{-1} (HKx_1(0) + HSx_2(0) + HGu_1(0) + \ddot{u}_2(0)) \\ &= -0.5 \left(\begin{bmatrix} -3 & 3 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0.25 & -0.25 \end{bmatrix} \begin{bmatrix} -0.5 \\ 0.5 \end{bmatrix} + 0(\sin(0)) - \sin(0) \right) = 0.125 \end{aligned}$$

produces the consistent initial condition $x(0) = \tilde{x}(0) = \begin{bmatrix} 0 & 0 & -0.5 & 0.5 & 0.125 \end{bmatrix}^T$ for solving system (5.4) (structurally, system (5.3)) and the completions. Vector $\hat{x}(0) = \begin{bmatrix} 6 & 7 & 8 & 9 & 10 \end{bmatrix}^T$ is chosen arbitrarily (except that $\hat{x}(0) \neq x(0)$) as the initial condition for the full-order observer and is transformed accordingly for the other three observers discussed in Chapter 3.

System (5.4) is a regular linear time-invariant system of DAEs since the determinant of the matrix pencil, $\det(sE + F) = -2s^2 + \frac{1}{2}s - 2$, is zero for only two values of s . The matrix pencil eigenvalues are $0.125000 \pm 0.992157i$ and, hence, unstable due to their positive real parts. Whenever these eigenvalues are unobservable, the observer for which the observability check is being performed will fail to converge to the true state if constructed. Similarly, for the least squares completion, the additional dynamics eigenvalues are not stable since they equal 0, which keeps the completion from being detectable when considered with any output equation. Recall, the stability of the three additional dynamics eigenvalues for the stabilized completions depends on the choice of stabilization parameter λ . Unless otherwise noted, the stabilized least squares completion and the alternative stabilized completion are stabilized with $\lambda = 2$, so their additional dynamics eigenvalues equal -2 and are stable.

The following matrices are the completions' coefficient matrices (notationally, \tilde{A} , \tilde{B} from $\dot{\tilde{x}}(t) = \tilde{A}\tilde{x}(t) + \tilde{B}v(t)$, the linear time-invariant version of equation (2.2)).

STABILIZED LEAST SQUARES COMPLETION:

$$\begin{aligned}\tilde{A} &= \begin{bmatrix} -0.521739 & 0.521739 & 0.413043 & 0.586957 & -0.173913 \\ 0.521739 & -0.521739 & 0.586957 & 0.413043 & 0.173913 \\ -1.782609 & 0.782609 & -0.130435 & 0.380435 & 0.739130 \\ 0.782609 & -1.782609 & 0.380435 & -0.130435 & -0.739130 \\ 4.538043 & -4.538043 & -2.316576 & 2.316576 & -4.445652 \end{bmatrix}, \\ \tilde{B} &= \begin{bmatrix} 0.0 & -0.782609 & 0.0 & -0.565217 & 0.0 & -0.086957 & 0.0 & 0.0 \\ 0.0 & 0.782609 & 0.0 & 0.565217 & 0.0 & 0.086957 & 0.0 & 0.0 \\ 1.0 & -0.173913 & 0.0 & -0.347826 & 0.0 & -0.130435 & 0.0 & 0.0 \\ 1.0 & 0.173913 & 0.0 & 0.347826 & 0.0 & 0.130435 & 0.0 & 0.0 \\ 0.0 & -2.130435 & 0.0 & -3.260870 & 0.0 & -2.097826 & 0.0 & -0.5 \end{bmatrix}.\end{aligned}$$

ALTERNATIVE STABILIZED COMPLETION:

$$\begin{aligned}\tilde{A} &= \begin{bmatrix} -1.0 & 1.0 & 0.500 & 0.500 & 0.0 \\ 1.0 & -1.0 & 0.500 & 0.500 & 0.0 \\ -0.5 & -0.5 & -0.875 & 1.125 & 0.0 \\ -0.5 & -0.5 & 1.125 & -0.875 & 0.0 \\ 0.0 & 0.0 & 0.000 & 0.000 & -2.0 \end{bmatrix}, \\ \tilde{B} &= \begin{bmatrix} 0.0 & -1.0 & 0.0 & -0.50 & 0.0 & 0.000 & 0.0 & 0.0 \\ 0.0 & 1.0 & 0.0 & 0.50 & 0.0 & 0.000 & 0.0 & 0.0 \\ 1.0 & 0.0 & 0.0 & -1.00 & 0.0 & -0.500 & 0.0 & 0.0 \\ 1.0 & 0.0 & 0.0 & 1.00 & 0.0 & 0.500 & 0.0 & 0.0 \\ 0.0 & -3.0 & 0.0 & -1.25 & 0.0 & -0.875 & 0.0 & -0.5 \end{bmatrix}.\end{aligned}$$

LEAST SQUARES COMPLETION:

$$\begin{aligned}\tilde{A} &= \begin{bmatrix} 0.000 & 0.000 & 0.666667 & 0.333333 & 0.00 \\ 0.000 & 0.000 & 0.333333 & 0.666667 & 0.00 \\ -1.400 & 0.400 & 0.200000 & 0.050000 & 0.60 \\ 0.400 & -1.400 & 0.050000 & 0.200000 & -0.60 \\ 0.225 & -0.225 & 0.481250 & -0.481250 & -0.15 \end{bmatrix}, \\ \tilde{B} &= \begin{bmatrix} 0.0 & 0.0 & 0.0 & -0.333333 & 0.0 & 0.00 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.333333 & 0.0 & 0.00 & 0.0 & 0.0 \\ 1.0 & 0.0 & 0.0 & 0.000000 & 0.0 & -0.20 & 0.0 & 0.0 \\ 1.0 & 0.0 & 0.0 & 0.000000 & 0.0 & 0.20 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & -1.000000 & 0.0 & 0.05 & 0.0 & -0.5 \end{bmatrix}.\end{aligned}$$

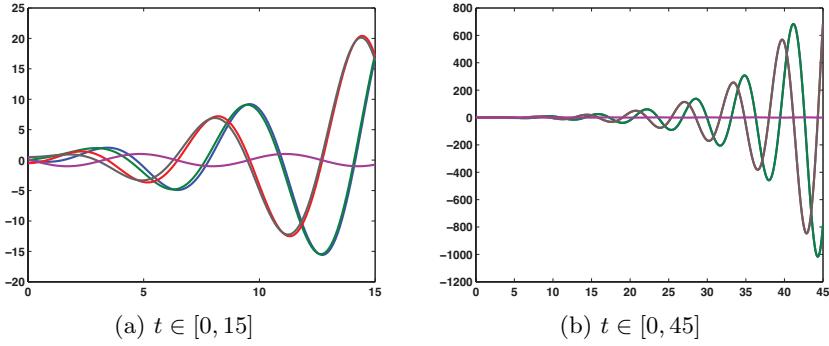


Figure 5.1: $x(t)$, solution of system (5.4).

The observer construction code for this linear time-invariant example system of DAEs was programmed using MATLAB version 7.0.1.15 (R14), 2004 Student Version, and was run on an IBM ThinkPad A20m with an Intel Pentium III processor, 547 MHz, 512 MB of RAM, and a 19.5 GB C: drive. For our numerical algorithms, the systems of ordinary differential equations are solved by MATLAB's `ode45` solver. This solver employs a variation of a Runge-Kutta iterative process for computing the solutions. The solver's returned results are output every 0.02 units of time for a visually smooth solution. The relative tolerance error (`RelTol`) is set at $1e-6$ instead of its default $1e-3$. Rounding is carried to six decimal places unless space is saved by not including repeated zeros. A comprehensive list of the MATLAB programs comprising the observer construction code is included in Section 5.3.

For all figures in this chapter, each color corresponds with a particular state variable: blue for the first component of $x_1(t)$, green for the second component of $x_1(t)$, red for the first component of $x_2(t)$, gray for the second component of $x_2(t)$, and magenta for $x_3(t)$. Notice the use of ‘component’, while ‘state variable’ is used to describe $x_1(t)$, $x_2(t)$, and $x_3(t)$. In captions, vs represents vertical scale; Sol stands for solution; Obs designates observer; and Diff identifies difference.

Figure 5.1a displays $x(t)$, the solution of system (5.4) and the state the observers are attempting to estimate on the $t \in [0, 15]$ interval. The final time $t_f = 15$ is arbitrary, but it provides a long enough interval on which to examine the effects of λ and ρ on observer convergence. Parameter ρ is the desired eigenvalue for the eigenvalue placement of observable eigenvalues by the gain matrix. The solution of system (5.4) is not periodic, a behavior that cannot be concluded from Figure 5.1a but is implied on the extended time interval $t \in [0, 45]$ in Figure 5.1b. The imaginary matrix pencil eigenvalues with positive real parts also indicate the oscillating solution will continue to grow rather than become periodic.

The completion solutions in Figure 5.2 appear to be the same as $x(t)$ in Figure 5.1a. The differences in Figure 5.3 between $x(t)$ and each completion solution ($x(t) - \tilde{x}(t)$) is zero except

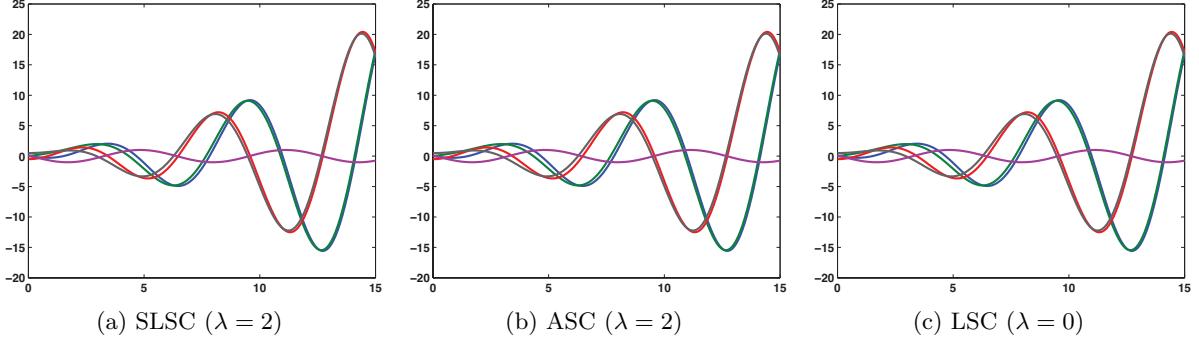


Figure 5.2: $\tilde{x}(t)$, completion solutions of system (5.4).

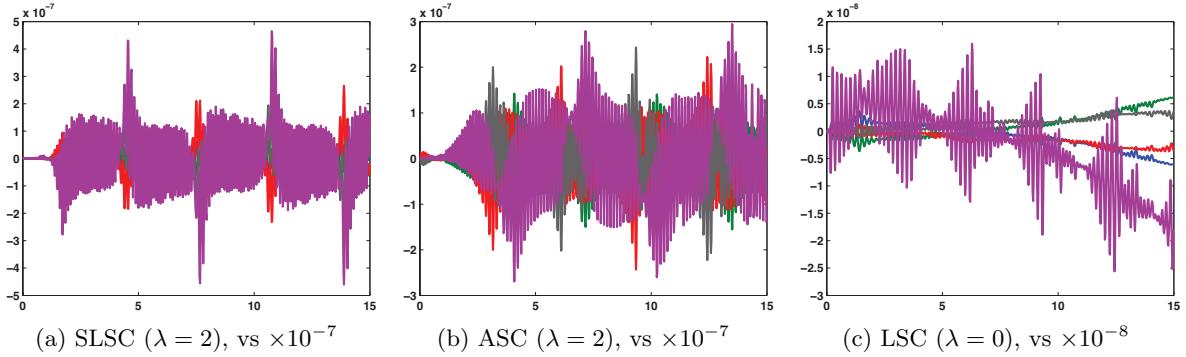


Figure 5.3: $x(t) - \tilde{x}(t)$, comparing exact solution with completion solutions.

for some numerical error, experimentally justifying the use of a completion to find the solution of a linear time-invariant system of DAEs. Although the vertical scale is $\times 10^{-8}$ in Figure 5.3c, the growth in the magnitude of the difference is not surprising since the least squares completion's additional dynamics are not stabilized.

In the linear time-invariant case, the dimension of the solution manifold equals the number of matrix pencil eigenvalues. Therefore, matrix \mathcal{G} from the characterization of the solution manifold $0 = -\mathcal{G}\mathcal{F}x(t) + \mathcal{G}\mathcal{B}v(t)$ is expected to have rank 3. Using MATLAB's svd command to determine \mathcal{G} , the matrices for constructing the maximally reduced observer's extended output equation are

$$\mathcal{G}\mathcal{F} = \begin{bmatrix} 0.113598 & -0.113598 & -0.181428 & 0.181428 & -0.572177 \\ 0.540337 & -0.540337 & 0.499090 & -0.499090 & -0.316543 \\ 1.579599 & -1.579599 & -0.252638 & 0.252638 & -0.610258 \end{bmatrix},$$

$$\mathcal{GB} = \begin{bmatrix} 0.0 & 0.744667 & 0.0 & 0.109906 & 0.0 & 0.286088 & 0.0 & 0.0 \\ 0.0 & -0.065523 & 0.0 & -0.538658 & 0.0 & 0.158271 & 0.0 & 0.0 \\ 0.0 & -0.664212 & 0.0 & 0.176356 & 0.0 & 0.305129 & 0.0 & 0.0 \end{bmatrix}.$$

Case 1 (output matrix C_a) investigates the effects from ρ and λ on the convergence of the full-order, reduced-order, and maximally reduced observers. Additionally, benefits of the maximally reduced observer are introduced.

Case 2 (output matrix C_b) considers the implications on our observer construction approach from knowing the matrix pencil eigenvalues are unobservable for the stabilized least squares completion paired with output matrix C_b .

Case 3 (output matrix C_c) also examines the effects from ρ and λ on the convergence of the full-order, reduced-order, and maximally reduced observers but only considers the stabilized completions.

Case 4 (DAE Manifold Observer) attempts to construct the DAE manifold observer for system (5.4) paired with each output matrix and analyzes the ensuing results.

Note, ‘full-order system’ and ‘completion with output equation’ are interchangeable. Also, reduced-order system describes the system associated with pair $(A_{R_{22}}, A_{R_{12}})$. In order to specify a pair for a particular completion, subscripts are included: $(\cdot, \cdot)_{SL}$ for the stabilized least squares completion, $(\cdot, \cdot)_A$ for the alternative stabilized completion, and $(\cdot, \cdot)_L$ for the least squares completion.

Case 1, C_a

Observer Preliminaries

The rank of observability matrix \mathcal{O} for pair $(\tilde{A}, C_a)_{SL}$ is 5, so all eigenvalues of \tilde{A} are observable. Every rank in Chapters 5 and 6 is determined in three steps. Using \mathcal{O} as an example matrix, first, MATLAB’s svd command computes the singular value decomposition of \mathcal{O} : $[U, D0, V] = \text{svd}(\mathcal{O})$. Second, our program Rounding.m is called, receiving singular value matrix $D0$ as the only input, in order to define entries of the input matrix between -10^{-10} and 10^{-10} as zero. The purpose of Rounding.m is to keep MATLAB’s internal rounding errors from causing a false rank if MATLAB fails to recognize these $\times 10^{-10}$ or smaller numbers as zero. Third, the rank of the returned and rounded singular value matrix D is computed and then identified as the rank of \mathcal{O} . The program Rounding.m is not example specific and is included in Appendix B.

The theoretical development of Remark 4.3 showed observability matrices for different completions of the same linear time-invariant system of DAEs may not have the same rank. Although $\text{rank}(\mathcal{O}) = 3$ for pair $(\tilde{A}, C_a)_A$ and $\text{rank}(\mathcal{O}) = 4$ for pair $(\tilde{A}, C_a)_L$, the matrix pencil eigenvalues are observable for both completions due to the observability of the stabilized least squares

completion with output matrix C_a and Theorem 4.1. Recall, Theorem 4.1 implies that for a particular output equation, if the matrix pencil eigenvalues are observable for one completion, they are observable for all completions. The standard form for the unobservable system

$$\begin{bmatrix} A_1 & 0 \\ A_2 & A_3 \end{bmatrix} = \left[\begin{array}{ccc|cc} -2.0 & 0.000000 & 0.000000 & 0.0 & 0.0 \\ 0.0 & 0.247006 & -0.972806 & 0.0 & 0.0 \\ 0.0 & 1.027194 & 0.002994 & 0.0 & 0.0 \\ \hline 0.0 & 0.000000 & 0.000000 & -2.0 & 0.0 \\ 0.0 & 0.000000 & 0.000000 & 0.0 & -2.0 \end{array} \right]$$

confirms pair $(\tilde{A}, C_a)_A$ is detectable since the two unobservable eigenvalues (block A_3) of the alternative stabilized completion with output matrix C_a equal -2 . The single eigenvalue of the unobservable block in

$$\begin{bmatrix} A_1 & 0 \\ A_2 & A_3 \end{bmatrix} = \left[\begin{array}{cccc|c} 0.050041 & 0.000000 & 0.000000 & 0.001252 & 0.0 \\ 0.000000 & 0.247006 & 0.972806 & 0.000000 & 0.0 \\ 0.000000 & -1.027194 & 0.002994 & 0.000000 & 0.0 \\ \hline -2.000030 & 0.000000 & 0.000000 & -0.050041 & 0.0 \\ 0.240763 & 0.000000 & 0.000000 & 0.785792 & 0.0 \end{array} \right]$$

equals 0 and verifies the least squares completion with output matrix C_a is unobservable.

Since 2×5 output matrix C_a is full row rank, measurements on two state variables are available through the output. Thus, the reduced-order observer must estimate only three unmeasurable components. Due to the structure of this output matrix, $x_2(t)$ is measurable, but the first m components of the state vector should be measurable when constructing the reduced-order

observer. Transformation matrix $R = \begin{bmatrix} C_a \\ C_R \end{bmatrix}$, where $C_R = \begin{bmatrix} -1 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$, provides

the desired $CR^{-1} = \begin{bmatrix} I & 0 \end{bmatrix}$ structure. Recall, the chosen method for defining C_R described in Section 3.3.1 utilizes the singular value decomposition of the output matrix.

Having already performed observability checks on the completions with output matrix C_a for constructing full-order observers, theory from Section 4.2 identifies whether the completions' reduced-order systems are observable, detectable, or unobservable without any computations. The pair $(A_{R22}, A_{R12})_{SL}$ from

$$\begin{bmatrix} A_{R11} & A_{R12} \\ A_{R21} & A_{R22} \end{bmatrix} = \left[\begin{array}{cc|cc} -0.130435 & 0.380435 & 1.782609 & -0.782609 & 0.739130 \\ 0.380435 & -0.130435 & -0.782609 & 1.782609 & -0.739130 \\ \hline -0.413043 & -0.586957 & -0.521739 & 0.521739 & 0.173913 \\ -0.586957 & -0.413043 & 0.521739 & -0.521739 & -0.173913 \\ -2.316576 & 2.316576 & -4.538043 & 4.538043 & -4.445652 \end{array} \right]$$

is observable due to Theorem 4.5: the full-order system is observable iff the reduced-order system is observable. Thus, there are no unobservable eigenvalues, so the observability matrix has rank 3. The pair $(A_{R22}, A_{R12})_A$ from

$$\begin{bmatrix} A_{R11} & A_{R12} \\ A_{R21} & A_{R22} \end{bmatrix} = \left[\begin{array}{cc|ccc} -0.875 & 1.125 & 0.5 & 0.5 & 0.0 \\ 1.125 & -0.875 & 0.5 & 0.5 & 0.0 \\ \hline -0.500 & -0.500 & -1.0 & 1.0 & 0.0 \\ -0.500 & -0.500 & 1.0 & -1.0 & 0.0 \\ 0.000 & 0.000 & 0.0 & 0.0 & -2.0 \end{array} \right].$$

is detectable due to Theorem 4.6: the full-order system is detectable iff the reduced-order system is detectable. Therefore, the reduced-order system also has two unobservable eigenvalues equaling -2 , so the observability matrix has rank 1. Finally, the pair $(A_{R22}, A_{R12})_L$ is unobservable due to Theorem 4.7: the full-order and reduced-order systems have the same unobservable eigenvalues. The one unobservable eigenvalue remains unobservable and since it equals zero, the least squares completion does not produce a reduced-order observer that asymptotically estimates the state.

For the construction of the maximally reduced observer,

$$C_\Xi = \begin{bmatrix} 0.000000 & 0.000000 & 1.000000 & 0.000000 & 0.000000 \\ 0.000000 & 0.000000 & 0.000000 & 1.000000 & 0.000000 \\ 0.113598 & -0.113598 & -0.181428 & 0.181428 & -0.572177 \\ 0.540337 & -0.540337 & 0.499090 & -0.499090 & -0.316543 \end{bmatrix}$$

is the extended output matrix when all of output matrix C_a is saved. The inclusion of the third row of \mathcal{GF} would keep the extended output matrix from being full row rank.

This extension of the output provides information on four state variables, so the order of the maximally reduced observer is one; that is, this observer must estimate only one unmeasurable component. For $C_{\bar{R}} = \begin{bmatrix} -0.707107 & -0.707107 & 0.0 & 0.0 & 0.0 \end{bmatrix}$, after applying transformation matrix $\bar{R} = \begin{bmatrix} C_\Xi \\ C_{\bar{R}} \end{bmatrix}$, the coefficient matrices $A_{\bar{R}}$ have structure

$$\begin{bmatrix} A_{\bar{R}11} & A_{\bar{R}12} \\ A_{\bar{R}21} & A_{\bar{R}22} \end{bmatrix} = \left[\begin{array}{cc|cc|c} 1.061206 & -0.811206 & 0.024234 & -2.378816 & 0.707107 \\ -0.811206 & 1.061206 & -0.024234 & 2.378816 & 0.707107 \\ \hline 2.956460 & -2.956460 & -1.815682 & -3.781789 & 0.000000 \\ 3.394567 & -3.394567 & -0.069677 & -6.056730 & 0.000000 \\ \hline -0.707107 & -0.707107 & 0.000000 & 0.000000 & 0.000000 \end{array} \right]$$

for the stabilized least squares completion;

$$\left[\begin{array}{cc} A_{\bar{R}_{11}} & A_{\bar{R}_{12}} \\ A_{\bar{R}_{21}} & A_{\bar{R}_{22}} \end{array} \right] = \left[\begin{array}{cccc|c} -0.875000 & 1.125000 & 0.0 & 0.0 & 0.707107 \\ 1.125000 & -0.875000 & 0.0 & 0.0 & 0.707107 \\ 0.000000 & 0.000000 & -2.0 & 0.0 & 0.000000 \\ 0.000000 & 0.000000 & 0.0 & -2.0 & 0.000000 \\ \hline -0.707107 & -0.707107 & 0.0 & 0.0 & 0.000000 \end{array} \right]$$

for the alternative stabilized completion; and

$$\left[\begin{array}{cc} A_{\bar{R}_{11}} & A_{\bar{R}_{12}} \\ A_{\bar{R}_{21}} & A_{\bar{R}_{22}} \end{array} \right] = \left[\begin{array}{cccc|c} 0.990094 & -0.740094 & -0.143896 & -1.635376 & 0.707107 \\ -0.740094 & 0.990094 & 0.143896 & 1.635376 & 0.707107 \\ -0.438380 & 0.438380 & 0.031630 & 0.359475 & 0.000000 \\ 0.953820 & -0.953820 & -0.155021 & -1.761817 & 0.000000 \\ \hline -0.707107 & -0.707107 & 0.000000 & 0.000000 & 0.000000 \end{array} \right]$$

for the least squares completion. The observability matrix is 4×1 with rank 1 for all three completions, indicating the only eigenvalue of their $A_{\bar{R}_{22}}$ submatrices, 0, is observable. Therefore, each of the three completions can be used to construct a maximally reduced observer that estimates the state. This case reveals it may be possible to construct the maximally reduced observer even if the full-order and reduced-order systems are unobservable.

Observer Results

The full-order observers constructed using the stabilized least squares completion (SLSC/FO observer) and the alternative stabilized completion (ASC/FO observer) are each plotted with the solution of system (5.4) in Figure 5.4. For these observers, the gain matrices place the observable eigenvalues at -1 ($\rho = -1$). An immediate indication the ASC/FO observer produces a better estimate of $x(t)$ is the figures' vertical scales. Recall, all five eigenvalues of the stabilized least squares completion are observable compared to just three of the alternative stabilized completion. Having fewer eigenvalues to place translates into less work by the observer to force the estimation error to zero since the stable unobservable eigenvalues assist with convergence. Additionally, the SLSC/FO observer is influenced by the polynomial terms connected with the completion's Jordan blocks.

An alternative measurement of an observer's estimate is the difference $x(t) - \hat{x}(t)$. Based on the estimation error's magnitude and the time it takes to converge to zero, Figures 5.5a and 5.5d reinforce the observation that with output matrix C_a , $\lambda = 2$, and $\rho = -1$, the ASC/FO observer provides a better estimate of $x(t)$ than the SLSC/FO observer. For both completions, reducing the observer's order to three when $\rho = -1$ produces a more accurate estimate of the state

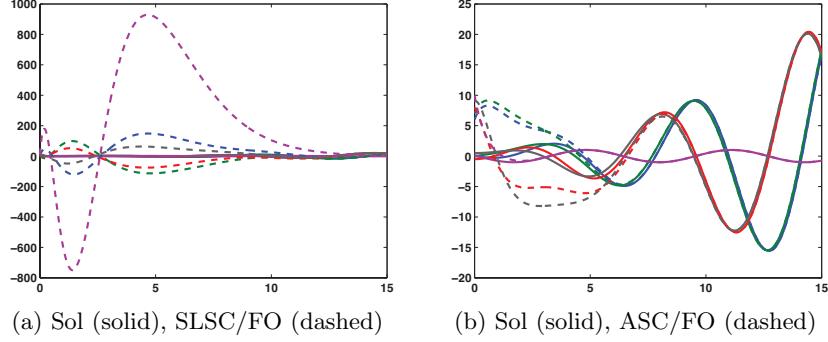


Figure 5.4: SLSC/FO, ASC/FO observers plotted with solution $x(t)$. (C_a , $\lambda = 2$, $\rho = -1$)

than when observing the full state vector. However, the distinct vertical scales in Figures 5.5b and 5.5e again reveal the reduced-order observer constructed using the alternative stabilized completion (ASC/RO observer) is closer to $x(t)$ than the reduced-order observer constructed using the stabilized least squares completion (SLSC/RO observer). Since $x_2(t)$ is measurable, the estimation error for this state variable is zero for both reduced-order observers.

Once the order of the observer is reduced to one, the estimation errors when $\rho = -1$ in Figures 5.5c, 5.5f, and 5.6a appear to be the same for the maximally reduced observers constructed using the stabilized least squares completion (SLSC/MR observer), the alternative stabilized completion (ASC/MR observer), and the least squares completion (LSC/MR observer). Figures 5.6b and 5.6c confirm these suspicions by plotting the difference between their maximally reduced observers (SLSC/MR – ASC/MR and SLSC/MR – LSC/MR, respectively). The differences are zero except for some numerical error, which suggests changing neither the completion nor λ alters the maximally reduced observer's estimate of $x(t)$. The first four components of transformed state vector $q(t)$ are known, so only the fifth component of $q(t)$ is estimated. However, transformation matrix \bar{R} listed under *Observer Preliminaries* is not a permutation matrix, so the maximally reduced observer $\hat{x}(t)$ estimates more than one component of $x(t)$. Note for $\hat{x}(t)$, the estimation error $x_1(t) - \hat{x}_1(t)$ converges to zero monotonically and the estimation errors $x_2(t) - \hat{x}_2(t)$ and $x_3(t) - \hat{x}_3(t)$ are zero, so the maximally reduced observer is providing a better estimate of the state than the full-order and reduced-order observers.

The results displayed in Figures 5.7, 5.8, and 5.9 consider how different values of ρ affect observer convergence. The effects are viewed using an estimation error progression, or a graph of $e(t)$ for an individual state variable as ρ is varied and output matrix C_a and $\lambda = 2$ do not change. The analysis investigates the response in the second component of $x_1(t)$, the first component of $x_2(t)$, and $x_3(t)$ to $\rho = -1, -2, -3, -4, -5$. Although not included, similar responses are observed for the first component of $x_1(t)$ and the second component of $x_2(t)$. The estimation errors are presented on the $t \in [0, 10]$ interval for a closer view of the initial difference.

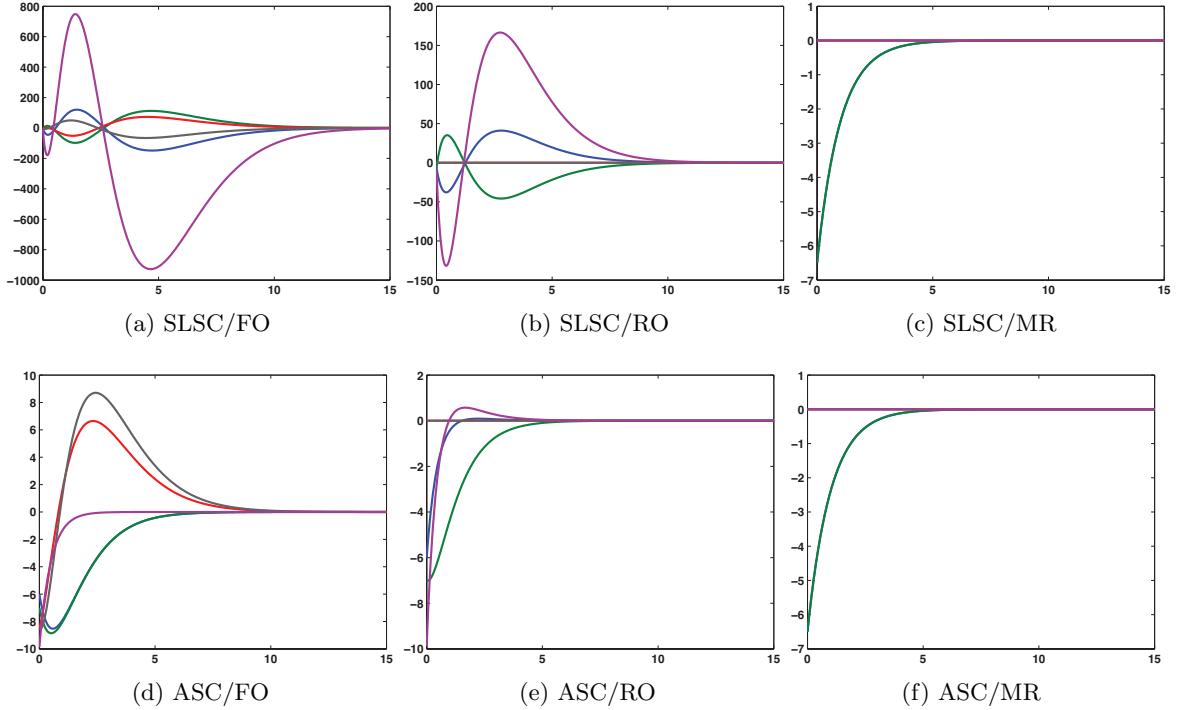


Figure 5.5: $x(t) - \hat{x}(t)$ comparison. ($C_a, \lambda = 2, \rho = -1$)

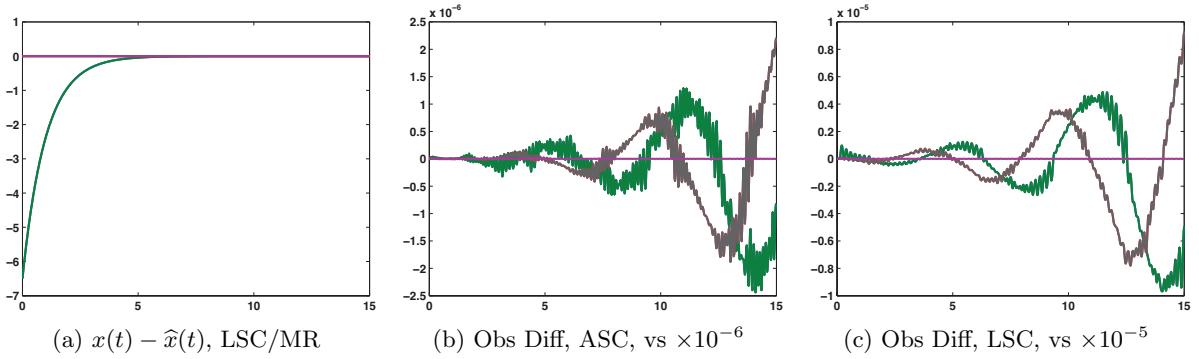


Figure 5.6: LSC/MR observer's $x(t) - \hat{x}(t)$; the MR observer differences are SLSC – ASC and SLSC – LSC. Neither the completion nor λ influences the MR observer. ($C_a, \lambda = 0, \lambda = 2, \rho = -1$)

Figure 5.7, the second component of $x_1(t)$: As ρ decreases, although $e(t) \rightarrow 0$ in less time, the initial difference between the solution of system (5.4) and $\hat{x}(t)$ increases in Figures 5.7a, 5.7b, and 5.7d when observing with the SLSC/FO, SLSC/RO, and ASC/FO observers. In Figures 5.7c and 5.7e, decreasing ρ also improves the convergence rate of the maximally reduced observer and the ASC/RO observer but without causing the observers' initial estimates to become less accurate. Notice, if a ρ less than -5 is chosen, the convergence properties and rates of the reduced-order observers may not differ from those when $\rho = -5$ since the progressions in Figures 5.7b and 5.7e appear bounded. Even for the full-order and reduced-order systems associated with a particular completion, altering ρ can affect the resulting observers' convergence properties and rates differently.

Figure 5.8, the first component of $x_2(t)$: Figures of the estimation error progressions for this component observed by the SLSC/RO, ASC/RO, and maximally reduced observers are not included because the differences are zero. In Figure 5.8a, the SLSC/FO observer's estimate exchanges initial accuracy for a faster convergence to $x(t)$ as ρ decreases. Figure 5.8b shows decreasing ρ improves the convergence properties and convergence rate of the ASC/FO observer in this component.

Figure 5.9, $x_3(t)$: Figure 5.9c is the first estimation error progression example where the choice of ρ does not affect the convergence of the observer in a specific component. An explanation comes from the structure of the gain matrices; for example, when $\rho = -1$, the SLSC/FO and ASC/FO observers' respective gain matrices are

$$L_{\text{SL}} = \begin{bmatrix} -20.046875 & -26.656250 \\ 12.046875 & 17.656250 \\ 4.000000 & 10.000000 \\ 1.250000 & -4.750000 \\ -94.484375 & -144.687500 \end{bmatrix}, \quad L_A = \begin{bmatrix} -3.00 & 2.875 \\ -3.00 & 2.875 \\ -2.75 & 3.000 \\ -4.75 & 4.000 \\ 0.00 & 0.000 \end{bmatrix}.$$

Recall, the ASC/FO observer is detectable, so the gain matrix is designed to place only two eigenvalues. The resulting structure of gain matrix L_A has a zero row that corresponds with $\hat{x}_3(t)$, so parameter ρ does not affect the convergence of the ASC/FO observer in this component. Otherwise, previously described responses from changing ρ are noticed again for the SLSC/FO, SLSC/RO, and ASC/RO observers. The estimation error progression is not included for the maximally reduced observer since $x_3(t) - \hat{x}_3(t)$ is zero.

When constructing an observer, the selection of ρ may force the user to choose between accuracy and desired rate of convergence. The following figures demonstrate how varying λ affects the convergence of the full-order and reduced-order observers but not the maximally reduced observer. The analysis investigates the response in the same three components from making $\lambda = 1, 2, 3, 4, 5$ without changing output matrix C_a and $\rho = -1$.

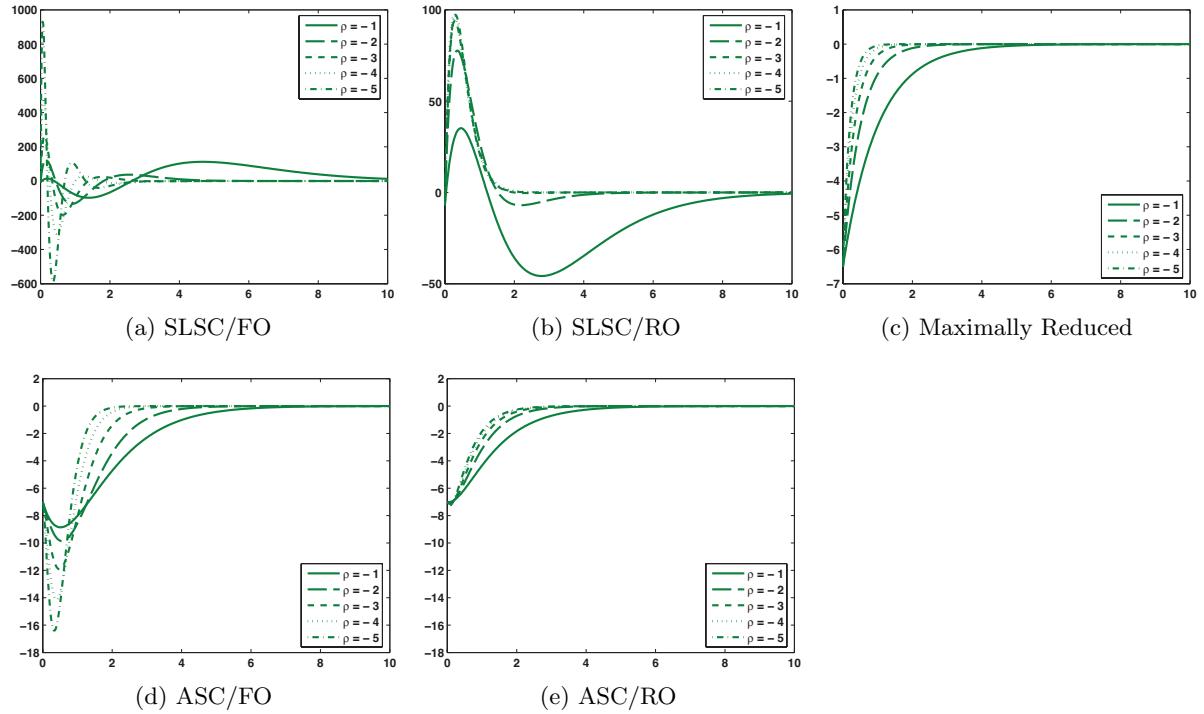


Figure 5.7: $x(t) - \hat{x}(t)$ progression as ρ is varied, 2nd component of $x_1(t)$. ($C_a, \lambda = 2$)

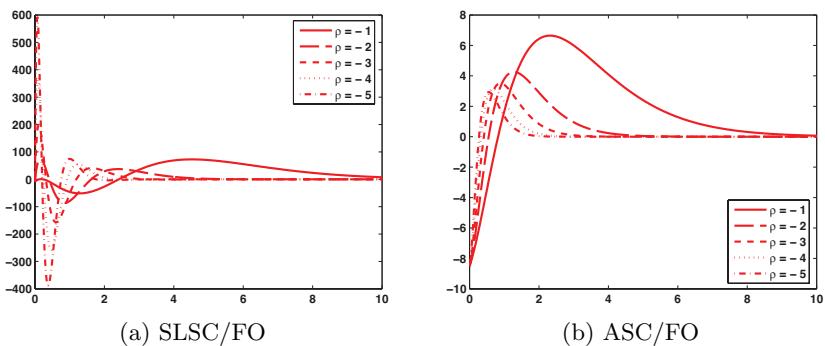


Figure 5.8: $x(t) - \hat{x}(t)$ progression as ρ is varied, 1st component of $x_2(t)$. ($C_a, \lambda = 2$)

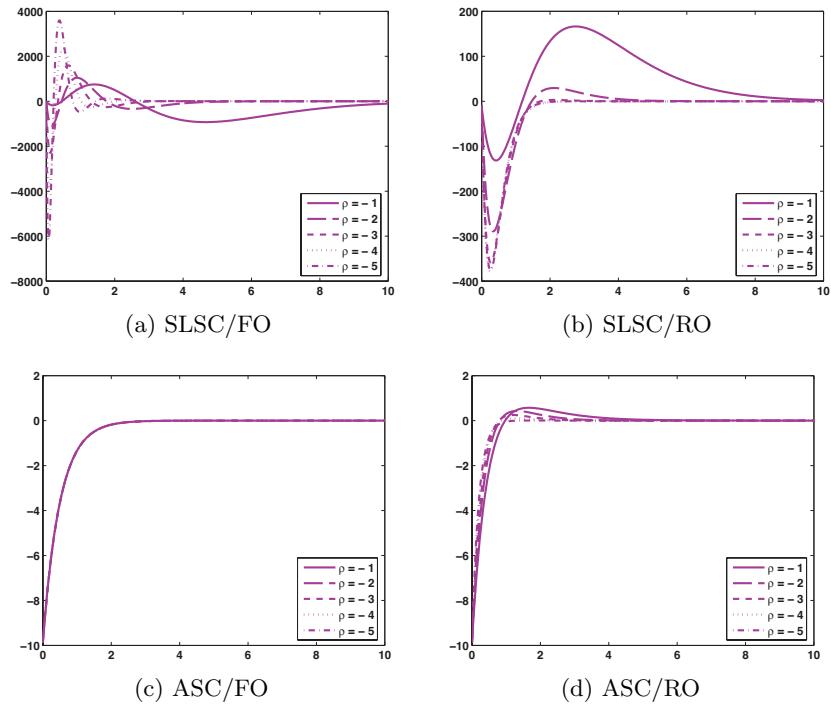


Figure 5.9: $x(t) - \hat{x}(t)$ progression as ρ is varied, $x_3(t)$. (C_a , $\lambda = 2$)

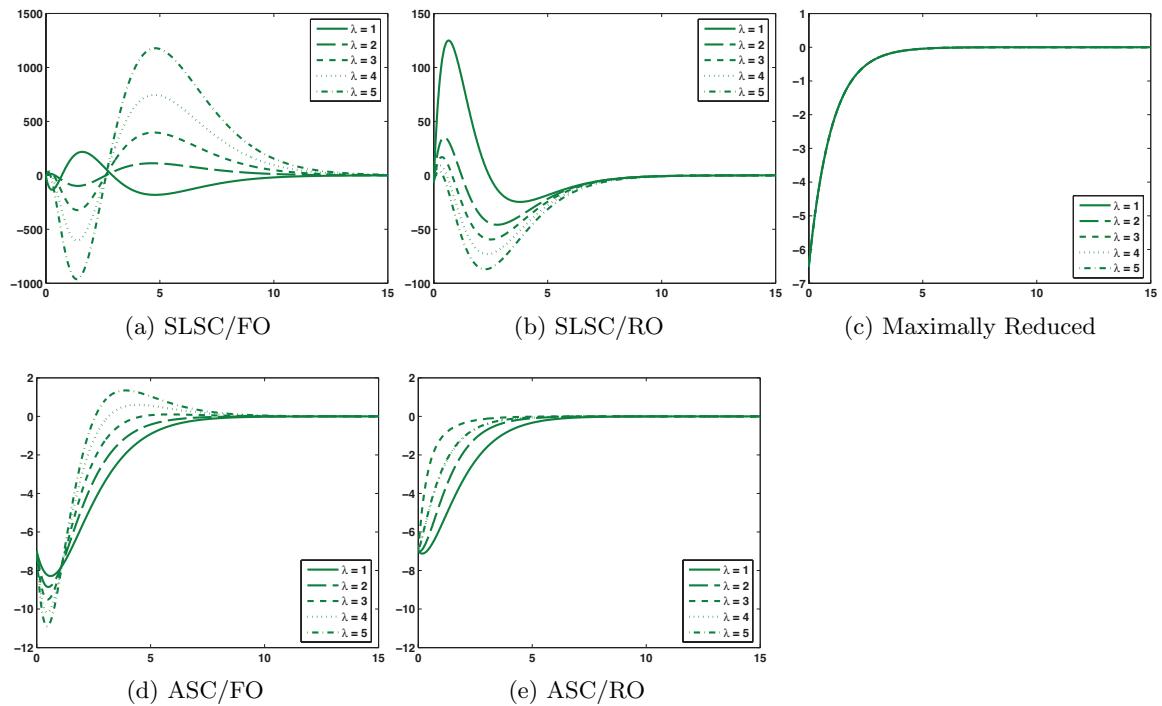


Figure 5.10: $x(t) - \hat{x}(t)$ progression as λ is varied, 2nd component of $x_1(t)$. ($C_a, \rho = -1$)

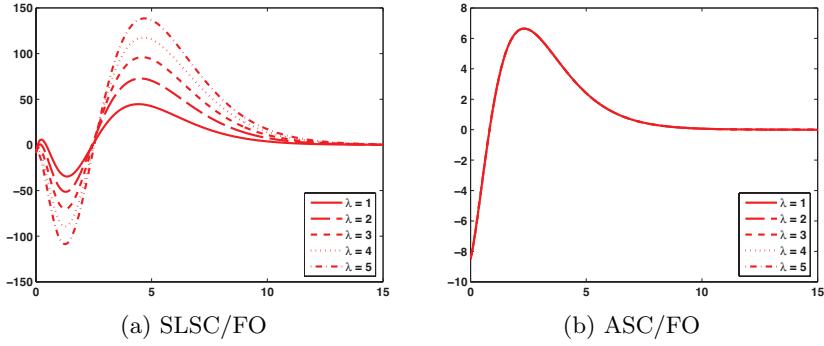


Figure 5.11: $x(t) - \hat{x}(t)$ progression as λ is varied, 1st component of $x_2(t)$. ($C_a, \rho = -1$)

Figure 5.10, the second component of $x_1(t)$: Figures 5.10a, 5.10d, and 5.10b show that increasing λ does not affect the convergence rate of the full-order observers or the SLSC/RO observer. However, stabilizing the additional dynamics so their eigenvalues are farther into the left half-plane does not improve the accuracy of the observers' estimates either. In Figure 5.10e, as λ increases, the estimation error for the ASC/RO observer begins to converge to zero monotonically. In Figure 5.10c, changing λ does not alter the maximally reduced observer's estimate of the state. This independence suggests the extension of the output matrix using information from the solution manifold keeps the completions' additional dynamics from influencing the behavior of the maximally reduced observer.

Figure 5.11, the first component of $x_2(t)$: The response of the SLSC/FO observer's estimate of this component when λ is decreased mimics the behavior observed in Figure 5.10a. But for the ASC/FO observer, λ and the completion's additional dynamics do not dictate how the observer estimates the first component of $x_2(t)$.

Figure 5.12, $x_3(t)$: Previously described responses from changing λ are noticed again for the SLSC/FO and SLSC/RO observers. Although Figure 5.9c showed the choice of ρ did not affect the ASC/FO observer's estimate of this component, increasing λ improves the convergence rate. In Figure 5.12d, it appears the estimation error progression will eventually converge to a difference that does not change when λ is increased.

In the *Observer Preliminaries*, all of output matrix C_a was saved when extending the output matrix. If all of \mathcal{GF} from the characterization of the solution manifold is saved instead, extended output matrix C_{Ξ} becomes

$$\begin{bmatrix} 0.113598 & -0.113598 & -0.181428 & 0.181428 & -0.572177 \\ 0.540337 & -0.540337 & 0.499090 & -0.499090 & -0.316543 \\ 1.579599 & -1.579599 & -0.252638 & 0.252638 & -0.610258 \\ 0.000000 & 0.000000 & 1.000000 & 0.000000 & 0.000000 \end{bmatrix}.$$

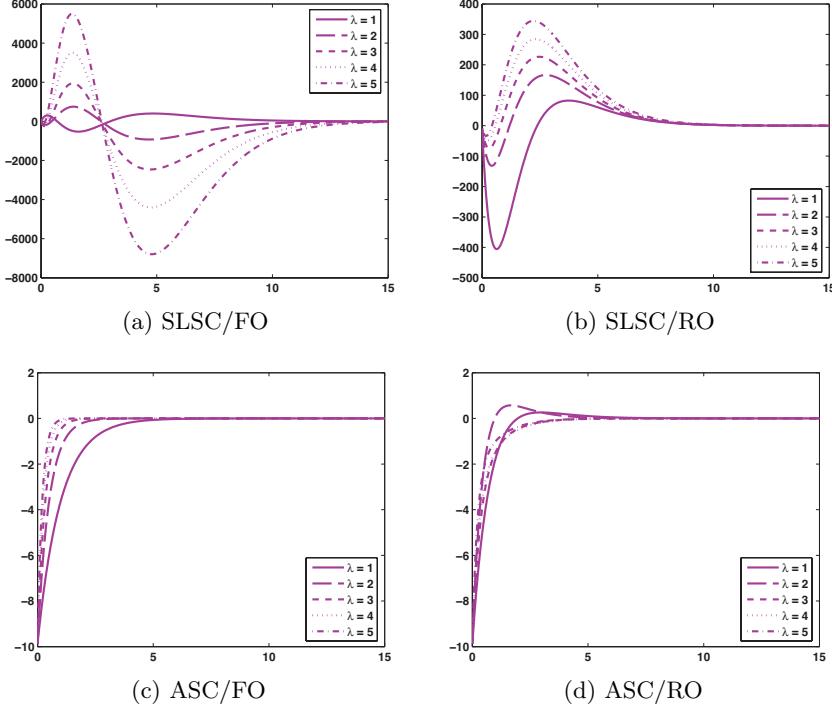


Figure 5.12: $x(t) - \hat{x}(t)$ progression as λ is varied, $x_3(t)$. (C_a , $\rho = -1$)

A different transformation matrix \bar{R} then alters the transformed system from which pair $(\bar{A}_{\bar{R}22}, \bar{A}_{\bar{R}12})$ comes. Figures 5.13a and 5.13b plot the transformed systems (solid) with their maximally reduced observers (dashed). The estimation errors $q(t) - \hat{q}(t)$ in Figure 5.14 show the modification to C_Ξ caused a reflection in the component being estimated by the maximally reduced observer. Once transformation $\hat{x}(t) = (\bar{R})^{-1}\hat{q}(t)$ is applied to both sets of results, Figures 5.15a and 5.15b imply the maximally reduced observers are the same, confirmed by the observer difference $((C_\Xi, C_a) - (C_\Xi, \mathcal{GF}))$ in Figure 5.15c. The two options for how to extend the output matrix produce the same final results (except for some numerical error).

Case 2, C_b

The observability matrix for pair $(\tilde{A}, C_b)_{SL}$ has rank 3. The standard form for the unobservable system reveals the eigenvalues of the unobservable block are the matrix pencil eigenvalues. Due to their positive real parts, this SLSC/FO observer cannot be designed to estimate the state. From Theorem 4.1, changing the completion does not alter the observability of the resulting full-order system paired with output equation $\tilde{y}(t) = C_b \tilde{x}(t)$. Both pairs $(\tilde{A}, C_b)_A$ and $(\tilde{A}, C_b)_L$ are unobservable with two of the four unobservable eigenvalues for the alternative stabilized completion being the matrix pencil eigenvalues and with the rank of \mathcal{O} for the least squares completion only equaling 2.

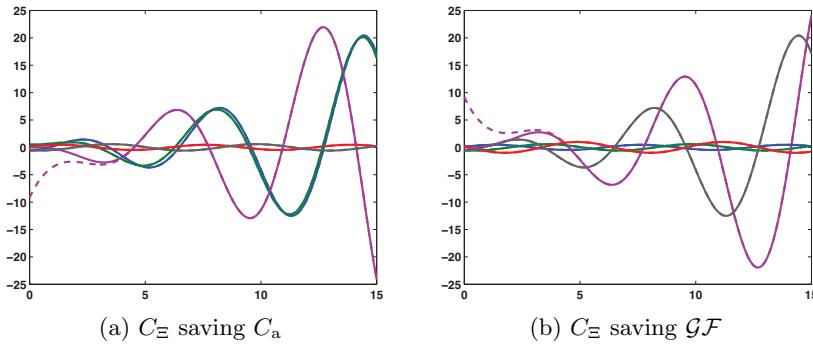


Figure 5.13: $q(t)$ (solid) and $\hat{q}(t)$ (dashed) for two extended output matrices. (C_a , $\rho = -1$)

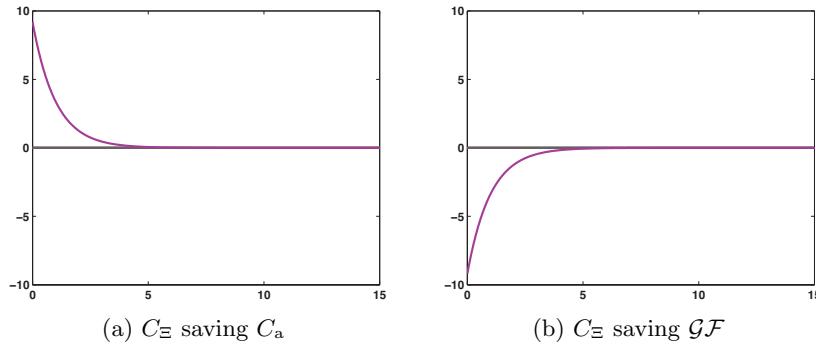


Figure 5.14: MR observers' $q(t) - \hat{q}(t)$ for two extended output matrices. (C_a , $\rho = -1$)

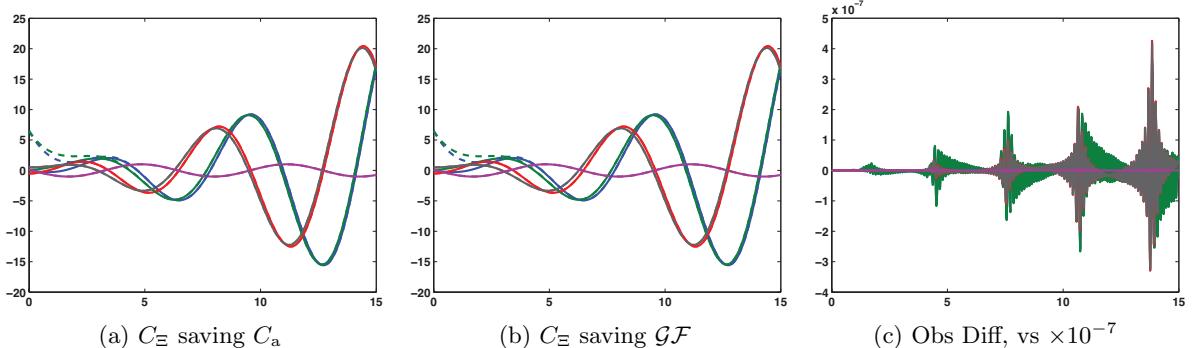


Figure 5.15: $x(t)$ (solid) and $\hat{x}(t)$ (dashed) for two extended output matrices; the MR observer difference is $(C_\Xi, C_a) - (C_\Xi, \mathcal{G}\mathcal{F})$. The results in terms of $x(t), \hat{x}(t)$ are the same. ($C_a, \rho = -1$)

Since the unobservable eigenvalues are the same for both the full-order and reduced-order systems (Theorem 4.7), neither the SLSC/RO, the ASC/RO, the LSC/RO, nor any completion's reduced-order observer can be constructed to converge to $x(t)$ when system (5.4)'s output equals the first minus the second component of $x_2(t)$. The extended output matrix for the construction of the maximally reduced observer is

$$C_{\Xi} = \begin{bmatrix} 0.000000 & 0.000000 & 1.000000 & -1.000000 & 0.000000 \\ 0.113598 & -0.113598 & -0.181428 & 0.181428 & -0.572177 \\ 0.540337 & -0.540337 & 0.499090 & -0.499090 & -0.316543 \end{bmatrix}.$$

Output matrix C_b is linearly independent from only two rows of \mathcal{GF} , indicating the output is providing information on the state already known from the constraints characterizing the solution manifold, or information on just the additional dynamics. Therefore, unobservable matrix pencil eigenvalues are expected when checking the observability of pair $(A_{\bar{R}_{22}}, A_{\bar{R}_{12}})$.

Recall, the observability check for the linear time-invariant system of DAEs and output equation $\tilde{y}(t) = C_b \tilde{x}(t)$ also failed. Although our observer construction approach is interested in the observability of the completions and their associated reduced-order systems, implementing a preliminary observability check for the system of DAEs would become a time-saving measure; that is, the process of constructing an observer would be terminated before unnecessarily spending resources to find a completion.

Case 3, C_c

Observer Preliminaries

Pairs $(\tilde{A}, C_c)_{SL}$ and $(\tilde{A}, C_c)_A$ are detectable. The rank of the observability matrix is 2 for both stabilized completions. For the alternative stabilized completion, the standard form for the unobservable system

$$\begin{bmatrix} A_1 & 0 \\ A_2 & A_3 \end{bmatrix} = \left[\begin{array}{cc|ccc} 0.247006 & 0.972806 & 0.0 & 0.0 & 0.0 \\ -1.027194 & 0.002994 & 0.0 & 0.0 & 0.0 \\ \hline 0.000000 & 0.000000 & -2.0 & 0.0 & 0.0 \\ 0.000000 & 0.000000 & 0.0 & -2.0 & 0.0 \\ 0.000000 & 0.000000 & 0.0 & 0.0 & -2.0 \end{array} \right]$$

reveals the eigenvalues in the unobservable block are the additional dynamics eigenvalues. Therefore, the matrix pencil eigenvalues are observable for all completions paired with output equation $\tilde{y}(t) = C_c \tilde{x}(t)$ (Theorem 4.1).

In order to construct the reduced-order observers, transformation matrix

$$R = \begin{bmatrix} 0.000000 & 0.0 & 1.0 & 1.0 & 0.0 \\ 0.000000 & 1.0 & 0.0 & 0.0 & 0.0 \\ -0.707107 & 0.0 & 0.5 & -0.5 & 0.0 \\ -0.707107 & 0.0 & -0.5 & 0.5 & 0.0 \\ 0.000000 & 0.0 & 0.0 & 0.0 & 1.0 \end{bmatrix}$$

transforms the completions so the output equals the first component of the transformed state vector. With output matrix C_c being 1×5 , the order of the reduced-order observer is four. The reduced-order systems' unobservable eigenvalues remain the same as their associated full-order systems' unobservable eigenvalues (Theorem 4.7), so the observability matrices for pairs $(A_{R_{22}}, A_{R_{12}})_{\text{SL}}$ and $(A_{R_{22}}, A_{R_{12}})_A$ have rank 1.

The single row of output matrix C_c is linearly independent from the rows of \mathcal{GF} , forming the extended output matrix

$$C_{\Xi} = \begin{bmatrix} 0.000000 & 0.000000 & 1.000000 & 1.000000 & 0.000000 \\ 0.113597 & -0.113598 & -0.181428 & 0.181428 & -0.572177 \\ 0.540337 & -0.540337 & 0.499090 & -0.499090 & -0.316543 \\ 1.579599 & -1.579599 & -0.252638 & 0.252638 & -0.610258 \end{bmatrix}$$

for the construction of the maximally reduced observer. Transformation matrix $\bar{R} = \begin{bmatrix} C_{\Xi} \\ C_{\bar{R}} \end{bmatrix}$ is nonsingular with $C_{\bar{R}} = \begin{bmatrix} 0.707107 & 0.707107 & 0.0 & 0.0 & 0.0 \end{bmatrix}$. Although the single eigenvalue of $A_{\bar{R}_{22}}$ seen in the block consideration of $A_{\bar{R}}$ for the stabilized least squares completion,

$$\left[\begin{array}{cc} A_{\bar{R}_{11}} & A_{\bar{R}_{12}} \\ A_{\bar{R}_{21}} & A_{\bar{R}_{22}} \end{array} \right] = \left[\begin{array}{cccc|c} 0.250000 & 0.000000 & 0.000000 & 0.000000 & -1.414214 \\ 0.000000 & -2.790479 & 0.995774 & -1.564168 & 0.000000 \\ 0.000000 & -1.188927 & -0.571196 & -1.795957 & 0.000000 \\ 0.000000 & -0.817106 & 0.908199 & -2.638324 & 0.000000 \\ \hline 0.707107 & 0.000000 & 0.000000 & 0.000000 & 0.000000 \end{array} \right],$$

and the alternative stabilized completion,

$$\left[\begin{array}{cc} A_{\bar{R}_{11}} & A_{\bar{R}_{12}} \\ A_{\bar{R}_{21}} & A_{\bar{R}_{22}} \end{array} \right] = \left[\begin{array}{cccc|c} 0.250000 & 0.0 & 0.0 & 0.0 & -1.414214 \\ 0.000000 & -2.0 & 0.0 & 0.0 & 0.000000 \\ 0.000000 & 0.0 & -2.0 & 0.0 & 0.000000 \\ 0.000000 & 0.0 & 0.0 & -2.0 & 0.000000 \\ \hline 0.707107 & 0.0 & 0.0 & 0.0 & 0.000000 \end{array} \right],$$

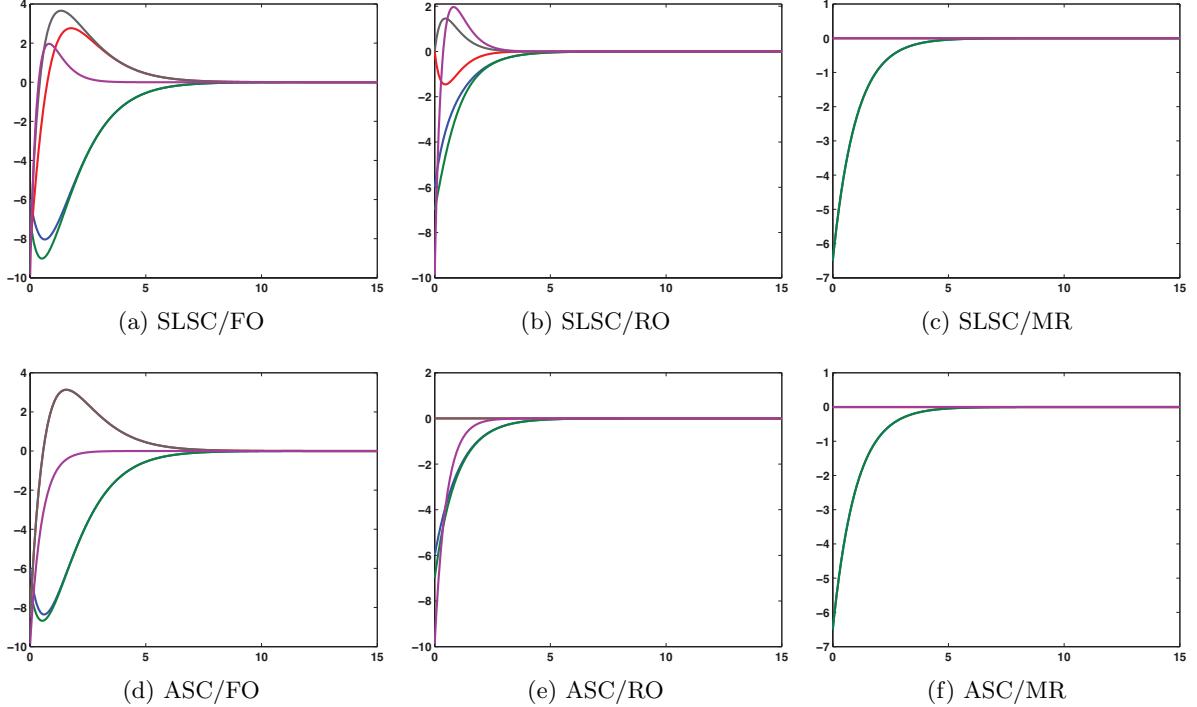


Figure 5.16: $x(t) - \hat{x}(t)$ comparison. (C_c , $\lambda = 2$, $\rho = -1$)

is zero, it is observable. Only one component of the transformed state vector must be estimated when constructing the maximally reduced observer.

Observer Results

For each of the estimation errors considered in Figure 5.16, the observable eigenvalues are placed at -1 ($\rho = -1$). Unlike Case 1, the stabilized completions paired with output matrix C_c have the same number of unobservable eigenvalues. Therefore, equal effort is exerted on eigenvalue placement, reflected in the comparable estimation errors appearing in Figures 5.16a and 5.16d for the SLSC/FO and ASC/FO observers. The influence from polynomial terms in the stabilized least squares completion's Jordan block is seen when contrasting the observers' estimates of $x_3(t)$. As the order of the observer is reduced, an unexpected dissimilarity in the SLSC/RO and ASC/RO observers appears in their estimates of $x_2(t)$ (see Figures 5.16b and 5.16e). Due to how the alternative stabilized completion is derived, having an output equal the sum of the first and second components of $x_2(t)$ turns out to provide enough information about these components so the ASC/RO observer's $\hat{x}_2(t)$ equals $x_2(t)$ beginning at $t_0 = 0$. However, the maximally reduced observers' estimates are even better since their $\hat{x}_3(t)$ estimates equal $x_3(t)$ beginning at $t_0 = 0$ as well.

Notice, pairs $(A_{\bar{R}_{22}}, A_{\bar{R}_{12}})_{\text{SL}}$ and $(A_{\bar{R}_{22}}, A_{\bar{R}_{12}})_{\text{A}}$ are the same:

$$\left(\begin{bmatrix} 0.000000 \\ 0.000000 \end{bmatrix}, \begin{bmatrix} -1.414214 \\ 0.000000 \\ 0.000000 \\ 0.000000 \end{bmatrix} \right).$$

Even the necessary modification to make $A_{\bar{R}_{12}}$ full row rank for observer construction does not distinguish the completions' systems to be observed. Therefore, the construction of the SLSC/MR and ASC/MR observers is based on the same homogeneous system. This system analysis reiterates the observation that the maximally reduced observer's estimate does not depend on the completion.

Figures 5.17, 5.18, and 5.19 plot estimation error progressions for examining how observer convergence is affected as ρ is varied and output matrix C_c and $\lambda = 2$ do not change. The tested values of parameter ρ are $\{-1, -2, -3, -4, -5\}$. Likewise, Figures 5.20, 5.21, and 5.22 are for analyzing the effects from changing λ and keeping output matrix C_c and $\rho = -1$ the same. In addition to $\lambda = 2$, other values considered are $\lambda = 1, 3, 4, 5$. The responses from varying these parameters are considered for the first component of $x_1(t)$, the second component of $x_2(t)$, and $x_3(t)$. Although not included, the second component of $x_1(t)$ and the first component of $x_2(t)$ behave similarly to the presented results. The estimation errors are plotted on the $t \in [0, 10]$ interval for a closer view of the initial difference. Estimation errors $x_2(t) - \hat{x}_2(t)$ and $x_3(t) - \hat{x}(t)$ are zero for the maximally reduced observer, so their progression figures are not included.

Figure 5.17, the first component of $x_1(t)$: Decreasing ρ makes the initial estimate of this component less accurate but improves the convergence rate for both the SLSC/FO and ASC/FO observers in Figures 5.17a and 5.17d. No matter the ρ , the ASC/RO observer's estimation error converges to zero monotonically in Figure 5.17e, but as ρ decreases for the SLSC/RO observer, the estimation error's monotonic convergence in Figure 5.17b disappears. As with the full-order observers, a smaller ρ translates into the reduced-order observers and the maximally reduced observer converging to $x(t)$ faster.

Figure 5.18, the second component of $x_2(t)$: For this component, the SLSC/FO and ASC/FO observers' estimates and convergence rates improve when ρ is decreased. In contrast to Figures 5.18a and 5.18c, Figures 5.18b and 5.18d show ρ does not affect the convergence of the SLSC/RO and ASC/RO observers in this component.

Figure 5.19, $x_3(t)$: All of the additional dynamics eigenvalues are unobservable for both stabilized completions' full-order and reduced-order systems. Thus, if SLSC/FO and ASC/FO observer convergence in $x_3(t)$ does not respond as ρ is varied, reducing the order of the observer with information from the output does not change the reduced-order observers' estimates from their respective full-order observers' estimates.

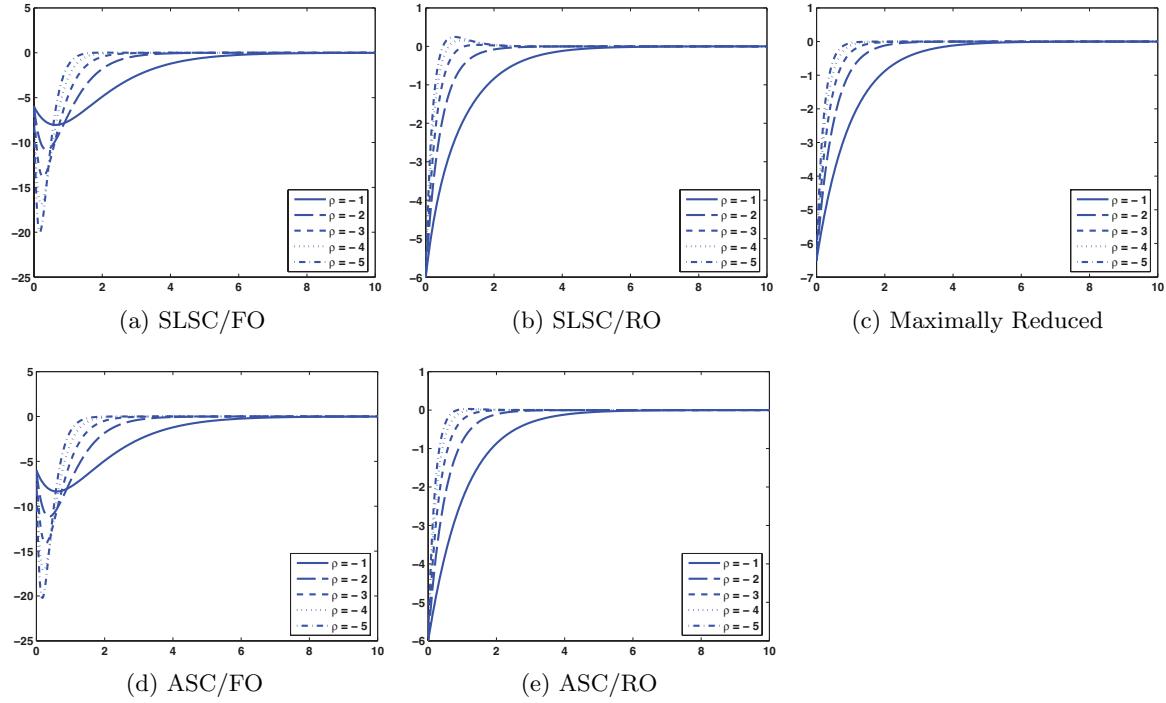


Figure 5.17: $x(t) - \hat{x}(t)$ progression as ρ is varied, 1st component of $x_1(t)$. (C_c , $\lambda = 2$)

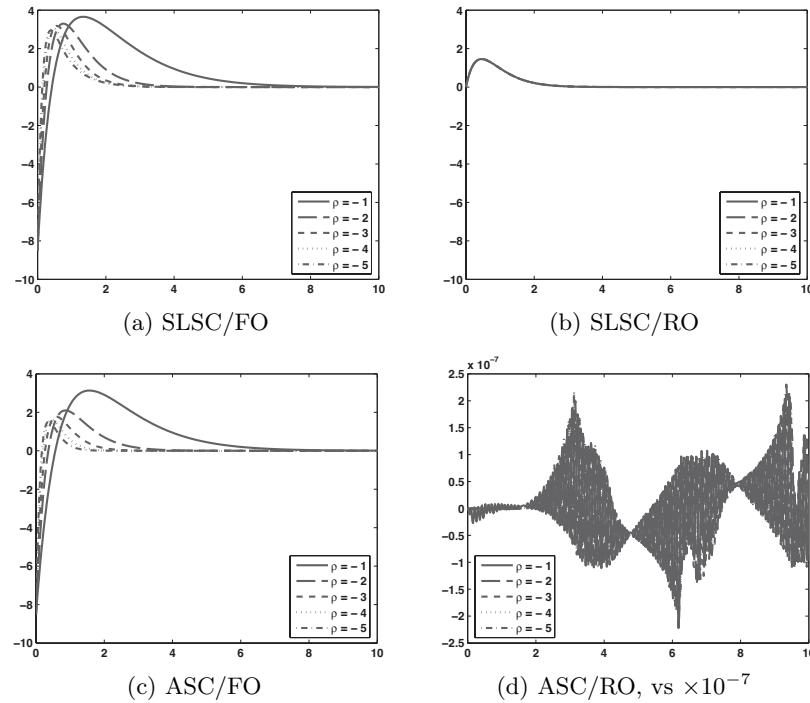


Figure 5.18: $x(t) - \hat{x}(t)$ progression as ρ is varied, 2nd component of $x_2(t)$. (C_c , $\lambda = 2$)

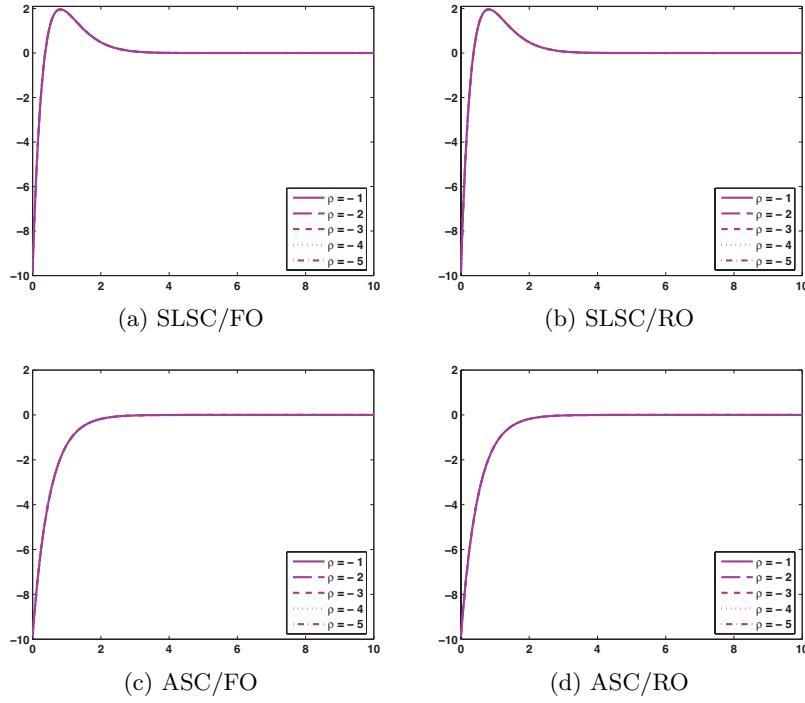


Figure 5.19: $x(t) - \hat{x}(t)$ progression as ρ is varied, $x_3(t)$. (C_c , $\lambda = 2$)

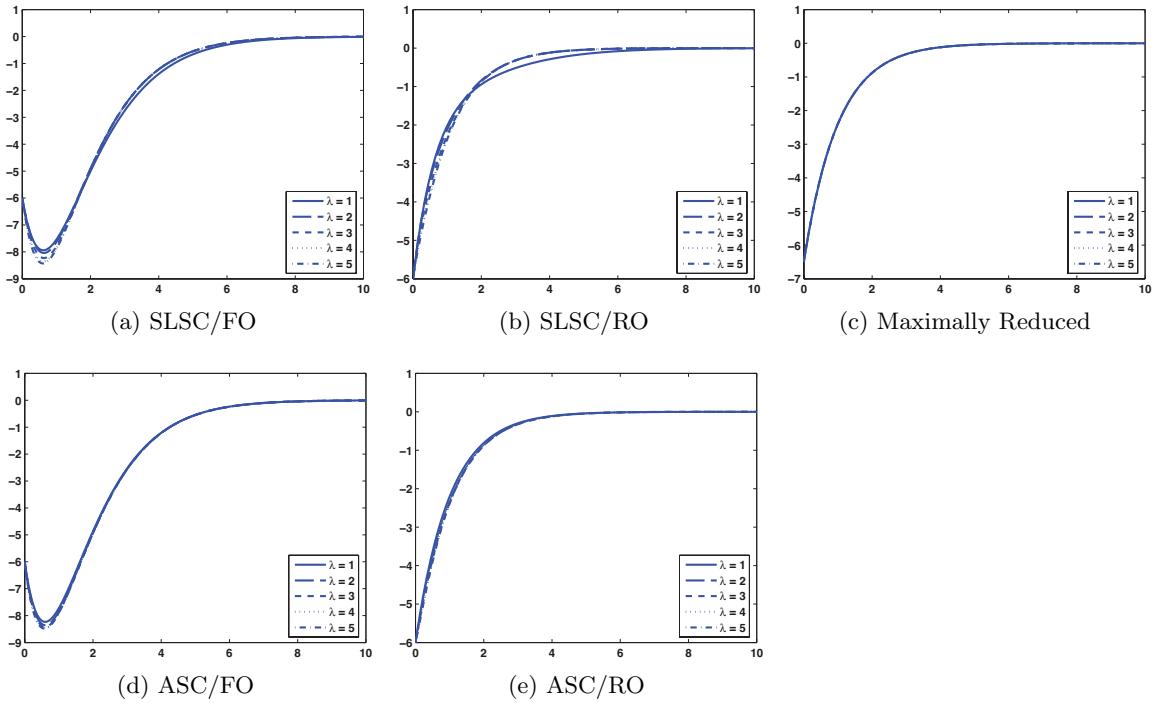


Figure 5.20: $x(t) - \hat{x}(t)$ progression as λ is varied, 1st component of $x_1(t)$. (C_c , $\rho = -1$)

Figure 5.20, the first component of $x_1(t)$: The influence of the additional dynamics is very little, if at all, on the observers' estimates of this component, demonstrated by the minimal response from increasing λ .

Figure 5.21, the second component of $x_2(t)$: The additional dynamics affect the convergence of the SLSC/FO and SLSC/RO observers. The estimate and convergence rate improve in Figure 5.21b as the additional dynamics move farther into the left half-plane, but it appears the SLSC/FO observer's estimate is converging to the ASC/FO observer's estimate of the component (compare Figures 5.21a and 5.21c).

Figure 5.22, $x_3(t)$: As expected from the lack of response in Figure 5.19 to the different values of ρ , modifying λ affects the stabilized completions' full-order and reduced-order estimates of $x_3(t)$. Increasing λ makes the observers' initial estimate less accurate for the stabilized least squares completion and increases the convergence rate for both completions.

Case 4, DAE Manifold Observer

The DAE manifold observer introduced in Section 3.5 to estimate the state of a linear time-invariant system of DAEs is constructed from the canonical form of the system's matrix pencil. This canonical form allows an observer to be constructed without ever finding a completion. The matrix pencil of system (5.4) is regular since $\det(sE + F) = 0$ for only two values of s . Thus, nonsingular matrices \mathcal{P} and \mathcal{Q} exist such that

$$\mathcal{P}E\mathcal{Q} = \begin{bmatrix} I & 0 \\ 0 & N \end{bmatrix}, \quad \mathcal{P}F\mathcal{Q} = \begin{bmatrix} \Omega & 0 \\ 0 & I \end{bmatrix}.$$

Command `pencan(E, A)` from Scilab (version 5.3.3) returns the desired nonsingular matrices for the matrix pencil $sE - A$ of system $E\dot{x}(t) = Ax(t) + Bu(t)$. Notice, $-F$ should be input for A . Currently, this Scilab operation is separate from the MATLAB programs, so the results for

$$\mathcal{P} = \begin{bmatrix} -0.707107 & -0.707107 & 0.000000 & 0.000000 & 0.000000 \\ 0.000000 & 0.000000 & 0.707107 & 0.707107 & 0.000000 \\ 0.125954 & -0.125954 & -0.346887 & 0.346887 & -1.232632 \\ 0.699331 & -0.699331 & 0.021623 & -0.021623 & 0.130749 \\ 0.007409 & -0.007409 & 0.348024 & -0.348024 & -0.194367 \end{bmatrix},$$

$$\mathcal{Q} = \begin{bmatrix} -0.707107 & 0.000000 & -0.343970 & 0.065627 & -0.346924 \\ -0.707107 & 0.000000 & 0.343970 & -0.065627 & 0.346924 \\ 0.000000 & 0.707107 & 0.070296 & 0.702028 & 0.026448 \\ 0.000000 & 0.707107 & -0.070296 & -0.702028 & -0.026448 \\ 0.000000 & 0.000000 & -1.243084 & 0.043080 & 0.194984 \end{bmatrix}$$

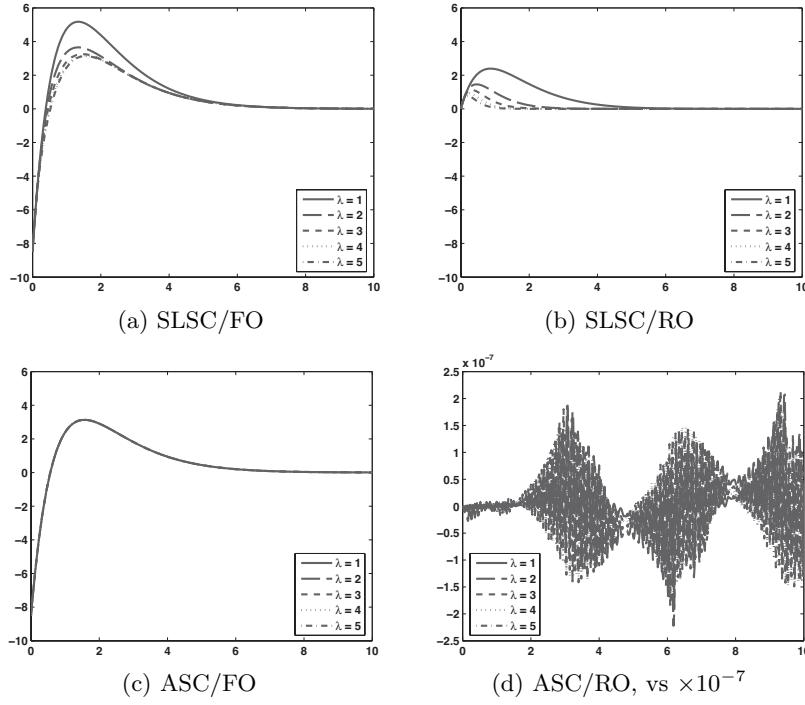


Figure 5.21: $x(t) - \hat{x}(t)$ progression as λ is varied, 2nd component of $x_2(t)$. ($C_c, \rho = -1$)

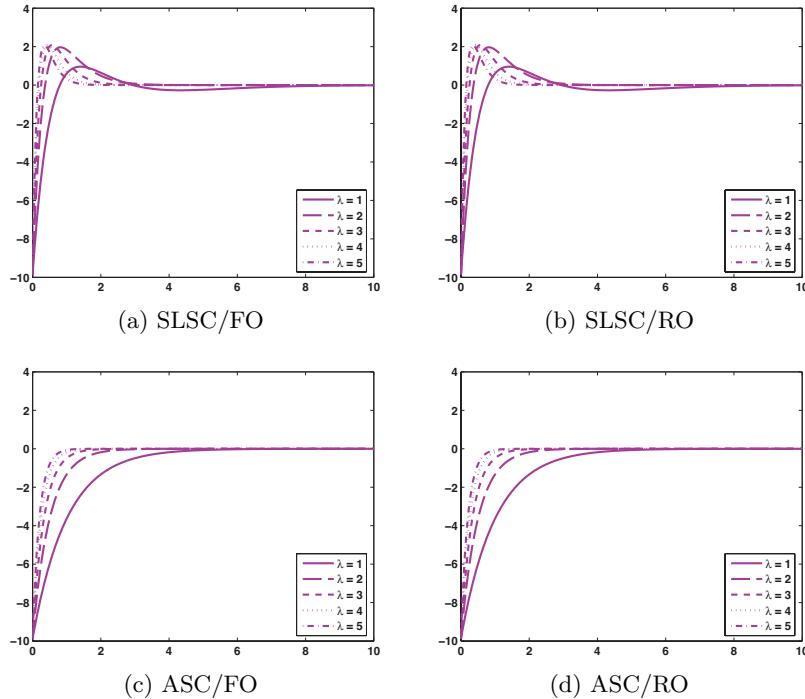


Figure 5.22: $x(t) - \hat{x}(t)$ progression as λ is varied, $x_3(t)$. ($C_c, \rho = -1$)

are hard coded into the main script DAEManObs.m (fourteen, not six, decimal places are included). Once the coefficient matrices of $\mathcal{P}E\dot{Q}p(t) = \mathcal{P}AQp(t) + \mathcal{P}Bu(t)$ are calculated, the transformed system can be written as

$$\dot{p}_1(t) = \begin{bmatrix} 0.0 & -1.00 \\ 1.0 & 0.25 \end{bmatrix} p_1(t) + \begin{bmatrix} 0.000000 & 0.0 \\ 1.414214 & 0.0 \end{bmatrix} u(t) \quad (5.5a)$$

$$\begin{bmatrix} -0.135419 & -0.470517 & -0.105742 \\ -0.478058 & 0.122150 & -0.484085 \\ 0.043832 & 0.489618 & 0.013269 \end{bmatrix} \dot{p}_2(t) = p_2(t) + \begin{bmatrix} 0.0 & -1.232632 \\ 0.0 & 0.130749 \\ 0.0 & -0.194367 \end{bmatrix} u(t). \quad (5.5b)$$

Recall, the derivation of the DAE manifold observer includes the characterization of the solution manifold $\mathcal{G}\mathcal{F}x(t) = \mathcal{G}\mathcal{B}v(t)$. The product $\mathcal{G}\mathcal{F}\mathcal{Q}$, which results when $\mathcal{Q}p(t)$ is substituted in for $x(t)$, has block structure

$$\begin{bmatrix} 0 & G \end{bmatrix} = \begin{bmatrix} 0.0 & 0.0 & 0.607608 & -0.264475 & -0.199982 \\ 0.0 & 0.0 & 0.091938 & 0.758036 & -0.410232 \\ 0.0 & 0.0 & -0.363587 & -0.173680 & -1.228354 \end{bmatrix},$$

where G is an invertible matrix. Thus, from equation (3.24),

$$\begin{aligned} p_2(t) &= G^{-1}\mathcal{G}\mathcal{B}v(t) \\ &= \begin{bmatrix} 0.0 & 1.232632 & 0.0 & -0.125954 & 0.0 & 0.346887 & 0.0 & 0.0 \\ 0.0 & -0.130749 & 0.0 & -0.699331 & 0.0 & -0.021623 & 0.0 & 0.0 \\ 0.0 & 0.194367 & 0.0 & -0.007409 & 0.0 & -0.348024 & 0.0 & 0.0 \end{bmatrix} v(t). \end{aligned}$$

Since the dimension of the solution manifold is 2, the DAE manifold observer is designed to estimate two state variables of the transformed state. The observer is then constructed for equation (5.5a) and output equations

$$\begin{aligned} \bar{y}(t) &= \begin{bmatrix} 0.000000 & 0.707107 \end{bmatrix} p_1(t) \quad \text{for Case 1,} \\ \bar{y}(t) &= \begin{bmatrix} 0.000000 & 1.414214 \end{bmatrix} p_1(t) \quad \text{for Case 3.} \end{aligned}$$

Note, $(C_{11}\mathcal{Q}_{11} + C_{12}\mathcal{Q}_{21}) = \begin{bmatrix} 0.000000 & 0.707107 \\ 0.000000 & 0.707107 \end{bmatrix}$ for original output matrix C_a and is redefined to be full row rank.

An observability check for this observer has not formally been decided upon. However, due to how this observer is constructed, the eigenvalues of the coefficient matrix of $p_1(t)$ will always be the matrix pencil eigenvalues. Therefore, an observability check for the linear time-invariant

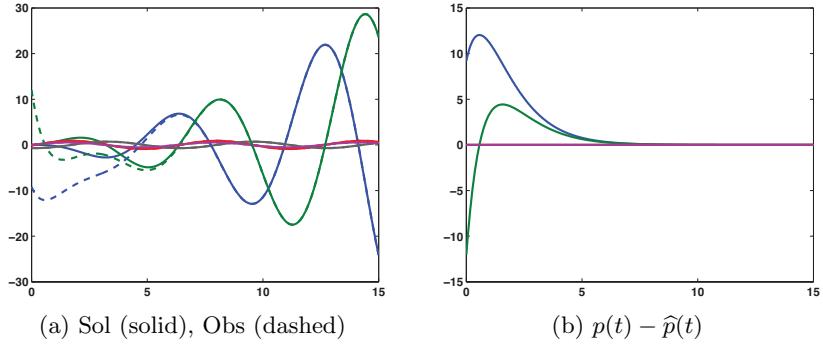


Figure 5.23: Solution $p(t)$ and DAE/M observer $\hat{p}(t)$ plotted together; DAE/M observer's $p(t) - \hat{p}(t)$. ($C_a, \rho = -1$)

system of DAEs would reveal the observability of the matrix pencil eigenvalues. Since the matrix pencil eigenvalues for system (5.4) have a positive real part, if they are unobservable, as with output matrix C_b , a DAE manifold observer cannot be designed to estimate the state. We suspect for matrix pencil eigenvalues that are not observable but have a negative real part, the construction of our DAE manifold observer will allow for detectable systems since the observer estimating $p_1(t)$ is a Luenberger observer.

For Case 1's DAE manifold observer results when $\rho = -1$, Figure 5.23a plots $\hat{p}(t) = \begin{bmatrix} \hat{p}_1(t) \\ p_2(t) \end{bmatrix}$ (dashed) with $p(t) = \begin{bmatrix} p_1(t) \\ p_2(t) \end{bmatrix}$ (solid), while Figure 5.23b graphs the estimation error $p(t) - \hat{p}(t)$. Figures 5.24a and 5.24b display the results from Figure 5.23 in terms of the original state vector, having applied transformation $\hat{x}(t) = Q\hat{p}(t)$. Abbreviation DAE/M is used for DAE manifold in the captions of this case's figures.

The convergence properties and rate are comparable to those for the ASC/FO observer with output matrix C_c , $\lambda = 2$, and $\rho = -1$, included again in Figure 5.24c (originally seen in Figure 5.16d). However, since the DAE manifold observer is constructed without a completion, there is no stabilization parameter λ to influence the observer's convergence to $x(t)$. The effects from decreasing ρ shown via estimation error progressions in the first component of $x_1(t)$ (representative of the second component of $x_1(t)$) and the first component of $x_2(t)$ (representative of the second component of $x_2(t)$) appear in Figure 5.25. The estimation error $x(t) - \hat{x}(t)$ in $x_3(t)$ is zero for all time.

The DAE manifold observer does not change when output matrix C_a is exchanged for output matrix C_c . The differences between the observers ($C_a - C_c$) in Figure 5.26b is zero (no noise appears). This zero difference can be explained by reconsidering the construction of these cases' maximally reduced observers. Vector $C_{\bar{R}}$ from transformation matrix $\bar{R} = \begin{bmatrix} C_{\Xi} \\ C_{\bar{R}} \end{bmatrix}$ is the same

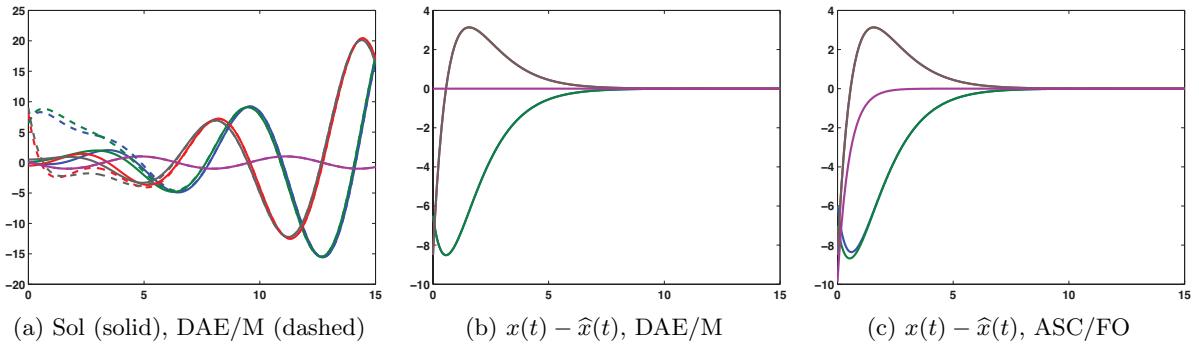


Figure 5.24: Solution $x(t)$ and DAE/M observer's $\hat{x}(t)$ plotted together; DAE/M observer's $x(t) - \hat{x}(t)$. This observer's $e(t)$ is comparable to the $e(t)$ for the ASC/FO observer with C_c , $\lambda = 2$, $\rho = -1$. (C_a , $\rho = -1$)

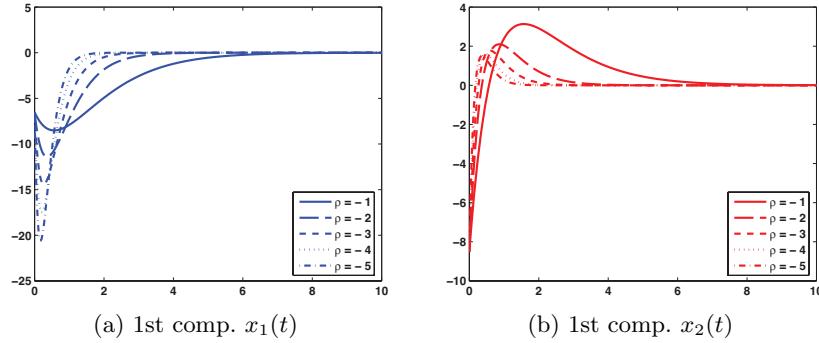


Figure 5.25: $x(t) - \hat{x}(t)$ progression as ρ is varied, DAE/M observer. (C_a)

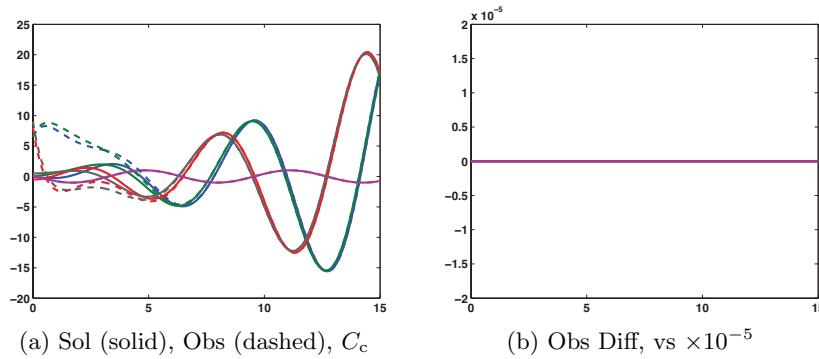


Figure 5.26: Solution $x(t)$ and DAE/M observer's $\hat{x}(t)$ plotted together; the DAE/M observer difference is $C_a - C_c$. The DAE/M observers are the same for Cases 1 and 3.

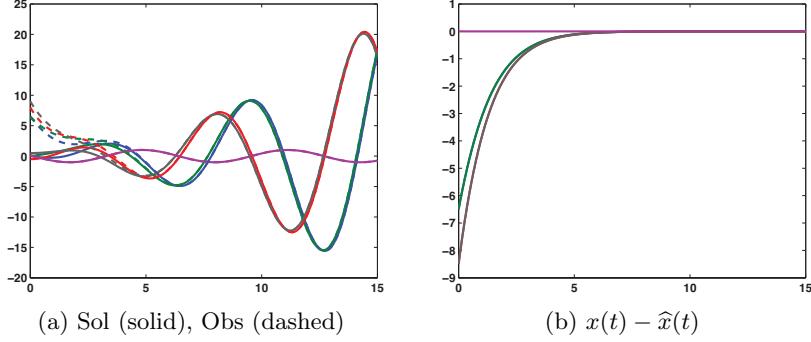


Figure 5.27: Solution $x(t)$ and DAE/M observer's $\hat{x}(t)$ plotted together; DAE/M observer's $x(t) - \hat{x}(t)$. ($C_d, \rho = -1$)

for output matrices C_a and C_c . Recall, only two rows of \mathcal{GF} were linearly independent from the rows of C_a (or only one row of C_a was linearly independent from the rows of \mathcal{GF}), while C_c and \mathcal{GF} were linearly independent. Since output matrices C_a and C_c provide information on the same variables ($y(t) = C_a x(t)$ outputs the first component of $x_2(t)$ and the second component of $x_2(t)$; $y(t) = C_c x(t)$ outputs the first component of $x_2(t)$ plus the second component of $x_2(t)$), these cases' extended output equations end up outputting measurements for the same state variables. Their output equations $\bar{y}(t)$ do the same for the DAE manifold observer. Therefore, the gain matrices are computed to identically affect convergence. That is, the product of the gain matrix and the transformed system's output matrix is the same for each case:

$$\text{Case 1 : } \begin{bmatrix} 0.000000 \\ 3.181981 \end{bmatrix} \begin{bmatrix} 0.0 & 0.707107 \end{bmatrix} = \begin{bmatrix} 0.0 & 0.00 \\ 0.0 & 2.25 \end{bmatrix},$$

$$\text{Case 3 : } \begin{bmatrix} 0.000000 \\ 1.590990 \end{bmatrix} \begin{bmatrix} 0.0 & 1.414214 \end{bmatrix} = \begin{bmatrix} 0.0 & 0.00 \\ 0.0 & 2.25 \end{bmatrix}.$$

In order to visually demonstrate the DAE manifold observer is not the same for every output matrix when the system is observable, consider output matrix $C_d = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{bmatrix}$. Both rows are linearly independent from \mathcal{GF} , and $\begin{bmatrix} 0.000000 & -2.828427 \\ 0.353553 & 0.000000 \end{bmatrix}$ is the gain matrix associated with output matrix $\bar{y}(t) = \begin{bmatrix} 0.000000 & 0.707107 \\ -0.707107 & 0.000000 \end{bmatrix} p_1(t)$ when $\rho = -1$. Figure 5.27a plots the observer $\hat{x}(t)$ (dashed) with the solution $x(t)$ (solid), and the estimation error in Figure 5.27b converges to zero monotonically.

5.3 Observer Construction Code

index.m: This function computes the index of a linear time-invariant system of DAEs. (pp. 212–213)

LTISystems.m: Example specific inputs are defined in this script. The goal is to keep MainL-TIObs.m and the programs it calls as general as possible by defining example-specific components in one location outside of the main algorithms. (pp. 214–215)

MainLTIObs.m: Main calling program. This function receives inputs defined by the user when called by LTISystems.m. The inputs are the coefficient matrices E , F , and B from the linear time-invariant system of DAEs; output matrix C ; the input (control) vector $u(t)$; λ for stabilized differentiation (0 if least squares completion); ρ for eigenvalue placement; and vector fn for constructing file names. The matrix for eigenvalue placement is currently ρI , so modifications are required if the user desires to place the observable eigenvalues at different values in the left half-plane. (pp. 216–219)

LTIDAE_SCDerArray.m, LTIDAE_AltSC.m: These functions are called by MainLTIObs.m to calculate coefficient matrices \tilde{A} and \tilde{B} for the least squares completion, the stabilized least squares completion, and the alternative stabilized completion. LTIDAE_SCDerArray.m also returns the coefficient matrices for the derivative array equations when computing \mathcal{G} for the maximally reduced observer. (pp. 219–221, pp. 221–222)

ExtendC.m: This function is called by MainLTIObs.m to compute the extended output matrix, other necessary elements for the extended output equation, and the maximally reduced observer's transformation matrix \bar{R} . A switch identifies whether output matrix C or \mathcal{GF} from the characterization of the solution manifold should be saved when extending the output matrix. (pp. 225–228)

FullOrdLTI_Calc.m, RedOrdLTI_Calc.m: These functions are called by MainLTIObs.m to determine gain matrices for constructing the full-order, reduced-order, and maximally reduced observers. (pp. 223–225, pp. 229–232)

EigPlace1.m, EigPlace2.m: These functions are called by FullOrdLTI_Calc.m and by RedOrdLTI_Calc.m and assist with eigenvalue placement for constructing the gain matrices (1 is for single output; 2 is for multi output). (pp. 233–234, pp. 234–237)

LTIDAE_Results.m: This function is called by MainLTIObs.m to solve the completion; to construct the full-order, reduced-order, and maximally reduced observers; and to plot the results. Function LTIDAE_Resultsmod.m, a variation of LTIDAE_Results.m that is not

included in Appendix B, is called by MainLTIObs.m when only the maximally reduced observer can be constructed to estimate the state. (pp. 237–244)

LTIDAE_Solve1.m: This function is called by LTIDAE_Results.m to find solutions for the example system, the completion, the transformed completions, and the full-order, reduced-order, and maximally reduced observers. Function LTIDAE_Solve2.m, a variation of LTIDAE_Solve1.m that is not included in Appendix B, is called by LTIDAE_Results.m when the output matrix cannot be extended (no maximally reduced observer). Additionally, a variation of LTIDAE_Solve1.m is called by LTIDAE_Resultsmod.m when only the maximally reduced observer can be constructed to estimate the state. (pp. 244–246)

DAEManObs.m: Example specifics are defined and \mathcal{P} , \mathcal{Q} for the construction of the DAE manifold observer are hard coded in this script. State variable $p_2(t)$ from the canonical form is solved for in this script using information from the characterization of the solution manifold. (pp. 246–251)

SolveDAEManObs.m: This function is called by DAEManObs.m to find the solutions for the example system and for $p_1(t)$ and $\hat{p}_1(t)$ from the process of constructing the DAE manifold observer. (pp. 251–252)

Rounding.m: This function can be called any time to force elements of a matrix that should be zero to be zero using a tolerance level with for loops and an if else statement. The only input is a matrix; these matrices have been calculated by MATLAB and are subject to internal rounding errors. The motivation behind this program was to eliminate false rank calculations by MATLAB’s rank command. (This function is also used for the linear time-varying example in Chapter 6.) (p. 294)

DAEobservable.m: This function determines whether or not a linear system of DAEs is observable based on discussion in [54]. Currently this program is set up for both the linear time-invariant and linear time-varying examples specific to Chapters 5 and 6. (pp. 294–298)

General programs with no example specific sections are index.m, MainLTIObs.m, ExtendC.m, FullOrdLTI_Calc.m, RedOrdLTI_Calc.m, EigPlace1.m, EigPlace2.m, and Rounding.m.

Chapter 6

Linear Time-Varying Example

6.1 Circuit Example Introduction

Rather than fabricating an arbitrary linear time-varying system of DAEs on which to test our observer construction approach, we chose a system with an application. The circuit in Figure 6.1 is based on an example in [125], [148] and can be described by the system of equations

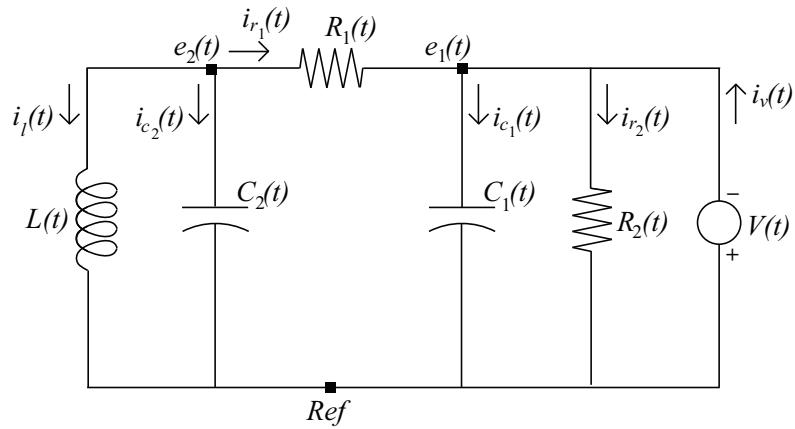


Figure 6.1: Circuit Example

$$(C_1(t)e_1(t))' - i_{r_1}(t) + i_{r_2}(t) - i_v(t) = 0 \quad (6.1a)$$

$$(C_2(t)e_2(t))' + i_l(t) + i_{r_1}(t) = 0 \quad (6.1b)$$

$$(L(t)i_l(t))' - e_2(t) = 0 \quad (6.1c)$$

$$-e_1(t) + e_2(t) - R_1(t)i_{r_1}(t) = 0 \quad (6.1d)$$

$$e_1(t) - R_2(t)i_{r_2}(t) = 0 \quad (6.1e)$$

$$e_1(t) = -V(t). \quad (6.1f)$$

In order to improve readability and maintain notation consistency in this chapter, the time derivative traditionally designated as $\dot{g}(t)$ will be designated as $g(t)'$. Ref represents ground (or the reference point); $L(t)$ is an inductor; $C_1(t)$, $C_2(t)$ are capacitors; $R_1(t)$, $R_2(t)$ are resistors; $V(t)$ represents a voltage source; $e_1(t)$, $e_2(t)$ are nodes at which to measure voltage; and $i_l(t)$, $i_{r_1}(t)$, $i_{r_2}(t)$, $i_v(t)$ are currents through $L(t)$, $R_1(t)$, $R_2(t)$, and $V(t)$, respectively. Additionally, $i_{c_1}(t)$, $i_{c_2}(t)$ are currents through $C_1(t)$ and $C_2(t)$, and $e_l(t)$, $e_{r_1}(t)$, $e_{r_2}(t)$ represent the voltage drops across the inductor and the two resistors. $C_1(t)$, $C_2(t)$, $L(t)$, $R_1(t)$, and $R_2(t)$ are assumed nonsingular. In terms of the matrices and vectors from the general linear time-varying system of DAEs $E(t)x(t)' + F(t)x(t) = B(t)u(t)$, system (6.1) can be written as $u(t) = V(t)$,

$$x(t) = \begin{bmatrix} e_1(t) \\ e_2(t) \\ i_l(t) \\ i_{r_1}(t) \\ i_{r_2}(t) \\ i_v(t) \end{bmatrix}, \quad E(t) = \begin{bmatrix} C_1(t) & 0 & 0 & 0 & 0 & 0 \\ 0 & C_2(t) & 0 & 0 & 0 & 0 \\ 0 & 0 & L(t) & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix},$$

$$F(t) = \begin{bmatrix} C_1(t)' & 0 & 0 & -I & I & -I \\ 0 & C_2(t)' & I & I & 0 & 0 \\ 0 & -I & L(t)' & 0 & 0 & 0 \\ -I & I & 0 & -R_1(t) & 0 & 0 \\ I & 0 & 0 & 0 & -R_2(t) & 0 \\ I & 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \quad B(t) = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ -I \end{bmatrix}.$$

State vector $x(t)$ is $n \times 1$ and its elements $\{e_1(t), e_2(t), i_l(t), i_{r_1}(t), i_{r_2}(t), i_v(t)\}$ are referred to as state variables or as components.

Kirchhoff's laws for voltage and current are used to construct system (6.1) [29]. Current equation (6.1a) comes from $i_{r_1}(t) + i_v(t) = i_{c_1}(t) + i_{r_2}(t)$. The current through node $e_1(t)$ from resistor $R_1(t)$ and the voltage source equals the sum of the currents through resistor $R_2(t)$ and capacitor $C_1(t)$. Since the change in charge with respect to the change in time is current,

$Q_{c_1}(t)' = i_{c_1}(t)$, and since the capacitance is the ratio of charge to voltage, $Q_{c_1}(t) = C_1(t)e_1(t)$,

$$\begin{aligned} i_{r_1}(t) + i_v(t) &= i_{c_1}(t) + i_{r_2}(t) \implies i_{c_1}(t) - i_{r_1}(t) + i_{r_2}(t) - i_v(t) &= 0 \\ Q_{c_1}(t)' &- i_{r_1}(t) + i_{r_2}(t) - i_v(t) &= 0 \\ C_1(t)'e_1(t) &+ C_1(t)e_1(t)' - i_{r_1}(t) + i_{r_2}(t) - i_v(t) &= 0 \\ (C_1(t)e_1(t))' &- i_{r_1}(t) + i_{r_2}(t) - i_v(t) &= 0. \end{aligned}$$

Equation (6.1b) also describes current and is derived from $-i_l(t) = i_{c_2}(t) + i_{r_1}(t)$. Due to the way the arrows are drawn, the sum of the currents through resistor $R_1(t)$ and capacitor $C_2(t)$ measured from node $e_2(t)$ equals the negative of the current running through the inductor:

$$\begin{aligned} -i_l(t) &= i_{c_2}(t) + i_{r_1}(t) \implies i_{c_2}(t) + i_l(t) + i_{r_1}(t) &= 0 \\ Q_{c_2}(t)' &+ i_l(t) + i_{r_1}(t) &= 0 \\ C_2(t)'e_2(t) &+ C_2(t)e_2(t)' + i_l(t) + i_{r_1}(t) &= 0 \\ (C_2(t)e_2(t))' &+ i_l(t) + i_{r_1}(t) &= 0. \end{aligned}$$

Voltage equation (6.1c) is equivalent to saying the difference in voltage between node $e_2(t)$ and ground equals the voltage drop across the inductor:

$$\begin{aligned} e_2(t) &= e_l(t) \implies e_l(t) - e_2(t) &= 0 \\ L(t)'i_l(t) &+ L(t)i_l(t)' - e_2(t) &= 0 \\ (L(t)i_l(t))' &- e_2(t) &= 0 \end{aligned}$$

since $e_l(t) = L(t)'i_l(t) + L(t)i_l(t)'$. Equation (6.1d) states the difference in voltage between nodes $e_2(t)$ and $e_1(t)$ equals the voltage drop across resistor $R_1(t)$:

$$\begin{aligned} e_2(t) - e_1(t) &= e_{r_1}(t) \implies e_2(t) - e_1(t) - e_{r_1}(t) &= 0 \\ e_2(t) - e_1(t) &- R_1(t)i_{r_1}(t) &= 0 \end{aligned}$$

using Ohm's law $e_{r_1}(t) = R_1(t)i_{r_1}(t)$. The difference in voltage from node $e_1(t)$ to ground equals the voltage drop across resistor $R_2(t)$, resulting in voltage equation (6.1e):

$$\begin{aligned} e_1(t) &= e_{r_2}(t) \implies e_1(t) - e_{r_2}(t) &= 0 \\ e_1(t) &- R_2(t)i_{r_2}(t) &= 0. \end{aligned}$$

The original circuit in [125], [148] does not have a voltage source, but we add one so the example system can be controlled through an input. This modification affects equation (6.1a) ($i_v(t)$ does

not originally appear) and appends equation (6.1f), a description of the voltage drop across the voltage source between node $e_1(t)$ and ground, to the linear time-varying system of DAEs.

The method chosen for determining the index k of system (6.1) is based on Definition 1.1 where the index is the minimum number of times the system is differentiated with respect to t in order to determine $x(t)'$ as a continuous function of $x(t)$ and t . The coefficient matrices of $x(t)' = [e_1(t)' \ e_2(t)' \ i_l(t)' \ i_{r_1}(t)' \ i_{r_2}(t)' \ i_v(t)']^T$ and of $x(t)$ are divided into block form

$$E(t) = \left[\begin{array}{ccc|cc|c} C_1(t) & 0 & 0 & 0 & 0 & 0 \\ 0 & C_2(t) & 0 & 0 & 0 & 0 \\ 0 & 0 & L(t) & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 \end{array} \right] = \left[\begin{array}{ccc} E_1(t) & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{array} \right], \quad (6.2a)$$

$$F(t) = \left[\begin{array}{ccc|cc|c} C_1(t)' & 0 & 0 & -I & I & -I \\ 0 & C_2(t)' & I & I & 0 & 0 \\ 0 & -I & L(t)' & 0 & 0 & 0 \\ \hline -I & I & 0 & -R_1(t) & 0 & 0 \\ I & 0 & 0 & 0 & -R_2(t) & 0 \\ \hline I & 0 & 0 & 0 & 0 & 0 \end{array} \right] = \left[\begin{array}{ccc} F_1(t) & F_2 & F_3 \\ F_4 & F_5(t) & 0 \\ F_6 & 0 & 0 \end{array} \right] \quad (6.2b)$$

so that both $E_1(t)$ and $F_5(t)$ are invertible. The blocks from (6.2) are used to rewrite the example system as

$$E_1(t) \begin{bmatrix} e_1(t)' \\ e_2(t)' \\ i_l(t)' \end{bmatrix} + F_1(t) \begin{bmatrix} e_1(t) \\ e_2(t) \\ i_l(t) \end{bmatrix} + F_2 \begin{bmatrix} i_{r_1}(t) \\ i_{r_2}(t) \end{bmatrix} + F_3 \begin{bmatrix} i_v(t) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \quad (6.3a)$$

$$F_4 \begin{bmatrix} e_1(t) \\ e_2(t) \\ i_l(t) \end{bmatrix} + F_5(t) \begin{bmatrix} i_{r_1}(t) \\ i_{r_2}(t) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad (6.3b)$$

$$F_6 \begin{bmatrix} e_1(t) \\ e_2(t) \\ i_l(t) \end{bmatrix} = -V(t). \quad (6.3c)$$

The subvector $[e_1(t)' \ e_2(t)' \ i_l(t)']^T$ of the first derivative of the state vector can be solved for in equation (6.3a) since $E_1(t)$ is nonsingular. Differentiating the algebraic constraint equa-

tions (6.3b) and (6.3c) once produces

$$F_4 \begin{bmatrix} e_1(t)' \\ e_2(t)' \\ i_l(t)' \end{bmatrix} + F_5(t)' \begin{bmatrix} i_{r_1}(t) \\ i_{r_2}(t) \end{bmatrix} + F_5(t) \begin{bmatrix} i_{r_1}(t)' \\ i_{r_2}(t)' \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \quad (6.4a)$$

$$F_6 \begin{bmatrix} e_1(t)' \\ e_2(t)' \\ i_l(t)' \end{bmatrix} = -V(t)'. \quad (6.4b)$$

Subvector $\begin{bmatrix} i_{r_1}(t)' & i_{r_2}(t)' \end{bmatrix}^T$ can be solved for in equation (6.4a) since $F_5(t)$ is nonsingular. After substituting into equation (6.4b) for $\begin{bmatrix} e_1(t)' & e_2(t)' & i_l(t)' \end{bmatrix}^T$,

$$F_6 \left[-E_1(t)^{-1} \left(F_1(t) \begin{bmatrix} e_1(t) \\ e_2(t) \\ i_l(t) \end{bmatrix} + F_2 \begin{bmatrix} i_{r_1}(t) \\ i_{r_2}(t) \end{bmatrix} + F_3 \begin{bmatrix} i_v(t) \end{bmatrix} \right) \right] = -V(t)'. \quad (6.5)$$

This resulting algebraic constraint equation is differentiated once to get

$$\begin{aligned} & -F_6 \left[(E_1(t)^{-1})' \left(F_1(t) \begin{bmatrix} e_1(t) \\ e_2(t) \\ i_l(t) \end{bmatrix} + F_2 \begin{bmatrix} i_{r_1}(t) \\ i_{r_2}(t) \end{bmatrix} + F_3 \begin{bmatrix} i_v(t) \end{bmatrix} \right) + \right. \\ & \left. E_1(t)^{-1} \left(F_1(t)' \begin{bmatrix} e_1(t) \\ e_2(t) \\ i_l(t) \end{bmatrix} + F_1(t) \begin{bmatrix} e_1(t)' \\ e_2(t)' \\ i_l(t)' \end{bmatrix} + F_2 \begin{bmatrix} i_{r_1}(t)' \\ i_{r_2}(t)' \end{bmatrix} + F_3 \begin{bmatrix} i_v(t)' \end{bmatrix} \right) \right] = -V(t)''. \end{aligned}$$

$\begin{bmatrix} i_v(t)' \end{bmatrix}$ can be solved for since its coefficient matrix

$$F_6 E_1(t)^{-1} F_3 = \begin{bmatrix} I & 0 & 0 \end{bmatrix} \begin{bmatrix} C_1(t)^{-1} & 0 & 0 \\ 0 & C_2(t)^{-1} & 0 \\ 0 & 0 & L(t)^{-1} \end{bmatrix} \begin{bmatrix} -I \\ 0 \\ 0 \end{bmatrix} = -C_1(t)^{-1}$$

is nonsingular. Thus, the minimum number of times system (6.1) needs to be differentiated in order to determine $x(t)'$ as a continuous function of $x(t)$ is two ($k = 2$). Additionally, this differentiation process reveals four constraint equations (two from (6.3b), one from (6.3c), one from (6.5)), indicating the solution manifold is two-dimensional. Notice system (6.1) would have been index 1 if the entry in the location of the zero block on the diagonal of $F(t)$ had also been nonsingular. The resulting inclusion of $i_v(t)$ in equation (6.3c) would have made the construction and differentiation of equation (6.5) unnecessary.

Further examination of system (6.1) reveals output matrix structures that result in an observable system of DAEs; that is, together with the inputs, these outputs provide enough information to determine the state of system (6.1) [54]. Equation (6.1f) indicates state variable $e_1(t)$ is known from the input. After substituting equation (6.1f) into equation (6.1e),

$$\begin{aligned} e_1(t) - R_2(t)i_{r_2}(t) &= 0 \\ -V(t) - R_2(t)i_{r_2}(t) &= 0 \\ -R_2(t)i_{r_2}(t) &= V(t) \\ i_{r_2}(t) &= -R_2(t)^{-1}V(t), \end{aligned} \quad (6.6)$$

equation (6.6) shows state variable $i_{r_2}(t)$ is also determined from the input. Substituting equation (6.1f) into equation (6.1d) results in

$$\begin{aligned} -e_1(t) + e_2(t) - R_1(t)i_{r_1}(t) &= 0 \\ V(t) + e_2(t) - R_1(t)i_{r_1}(t) &= 0 \\ -R_1(t)i_{r_1}(t) &= -V(t) - e_2(t) \\ i_{r_1}(t) &= R_1(t)^{-1}(V(t) + e_2(t)). \end{aligned} \quad (6.7)$$

Equations (6.1a) and (6.7) with the definitions for $e_1(t)$ and $i_{r_2}(t)$ produce

$$\begin{aligned} 0 &= (C_1(t)e_1(t))' - i_{r_1}(t) + i_{r_2}(t) - i_v(t) \\ 0 &= C_1(t)'e_1(t) + C_1(t)e_1(t)' - i_{r_1}(t) + i_{r_2}(t) - i_v(t) \\ 0 &= -C_1(t)'V(t) - C_1(t)V(t)' - R_1(t)^{-1}(V(t) + e_2(t)) - R_2(t)^{-1}V(t) - i_v(t) \\ i_v(t) &= -(C_1(t)' + R_1(t)^{-1} + R_2(t)^{-1})V(t) - C_1(t)V(t)' - R_1(t)^{-1}e_2(t). \end{aligned} \quad (6.8)$$

Equations (6.1b), (6.1c), (6.7), and (6.8) imply that if $e_2(t)$ is measurable through the output, then $i_l(t)$, $i_{r_1}(t)$, and $i_v(t)$ are known as well. Additionally, if either $i_l(t)$, $i_{r_1}(t)$, or $i_v(t)$ is measurable, then $e_2(t)$ is known and can be used to determine the remaining current state variables. This examination suggests the output matrix $C(t)$ should be constructed so at least one of the state variables $e_2(t)$, $i_l(t)$, $i_{r_1}(t)$, or $i_v(t)$ is an output or is being output in a linear combination with any of the other state variables. The observability of a linear time-varying system of DAEs and its associated output equation can be confirmed using a method presented in [54] that checks the 1-fullness of an observability-like matrix. A matrix is 1-full with respect to $x(t)$ if it has row-echelon form $\begin{bmatrix} I_{n \times n} & 0 \\ 0 & * \end{bmatrix}$, where * has unspecified structure [54]. Although our observer construction approach does not require the LTV system of DAEs to be observable, we utilize this structure of $C(t)$ when selecting the output matrices presented in the next section.

6.2 Observer Implementation

The linear time-varying system of DAEs describing the circuit in Figure 6.1 is defined in general terms. In order to illustrate our observer construction approach on this circuit, we define the capacitors, inductor, and voltage source as $C_1(t) = 3 + \cos(t/3)$, $C_2(t) = 2 - \cos(2t)$, $L(t) = 2 - \exp(-t)$, and $V(t) = 4 \cos(2t) \sin(t/5)$, respectively. Two sets of resistors are considered. The system investigated in Subsection 6.2.1 has positive resistors $R_1(t) = 4 + 2 \sin(t)$ and $R_2(t) = 2 + \sin(t)$, while the system examined in Subsection 6.2.2 has their negative counterparts, $R_1(t) = -(4 + 2 \sin(t))$ and $R_2(t) = -(2 + \sin(t))$.

Although a system of equations is said to observe a physical system if the difference between the true state and the estimated state goes to zero as time goes to infinity, an effective observer has sufficiently small error by a specific time. Special consideration is given to constructing observers that converge in finite time in [127]. For both circuit examples, our goal is for the estimation error to be within the interval $(-10^{-3}, 10^{-3})$ by $t_f = 45$. This interval check was implemented as a response to some state variables' estimation errors having an oscillatory behavior near the desired final time. However, along with the estimation error plots, this numerical measure provides another check when discussing how well an observer is performing. At times, results are considered on the extended time interval $t \in [0, 135]$ to provide a better understanding of behavior over a longer period of time.

The observer construction code for this linear time-varying example system of DAEs was primarily programmed and executed using MATLAB version 7.0.1.15 (R14), 2004 Student Version, on the personal computer, an IBM ThinkPad A20m with Intel Pentium III processor, 547 MHz, 512 MB of RAM, and 19.5 GB C: drive. However, the numerical algorithms for checking detectability and for finding the alternative stabilized completion using the smooth decomposition method include MATLAB's `lyap` command available in the Control System Toolbox. The personal computer's student version of MATLAB does not have this special toolbox, so this code was run on North Carolina State University's Virtual Computing Lab (VCL) using MATLAB R2010a (WinXP). When this research began, sessions on the VCL computer lasted a maximum of four hours in order to allow other users access to a dedicated machine. Only since the end of year 2012 were the sessions extended to ten hours, so if run time on the personal computer became a factor, the code could be executed more quickly on the VCL computer. Unless otherwise noted, the results are from the personal computer's version of MATLAB.

For our numerical algorithms, MATLAB's `ode45` solver is called to solve the systems of ODEs. The time step for outputting the solver's returned results and for comparing other time-dependent computations is 0.01, a small enough step for producing smooth visual results without significantly increasing the programs' run times. The relative error tolerance (`RelTol`) is set at `1e-9` instead of its default `1e-3`, and the example that compelled this modification

appears in Case 2 of Subsection 6.2.2. Rounding is carried to six significant digits or to six decimal places unless space is saved by not including repeated zeros. A comprehensive list of the MATLAB programs comprising the observer construction code is included in Section 6.3.

6.2.1 Example with Positive Resistors

System Introduction

The first linear time-varying system of DAEs to which we apply our observer construction approach is

$$((3 + \cos(t/3)) e_1(t))' - i_{r_1}(t) + i_{r_2}(t) - i_v(t) = 0 \quad (6.9a)$$

$$((2 - \cos(2t)) e_2(t))' + i_l(t) + i_{r_1}(t) = 0 \quad (6.9b)$$

$$((2 - \exp(-t)) i_l(t))' - e_2(t) = 0 \quad (6.9c)$$

$$-e_1(t) + e_2(t) - (4 + 2 \sin(t)) i_{r_1}(t) = 0 \quad (6.9d)$$

$$e_1(t) - (2 + \sin(t)) i_{r_2}(t) = 0 \quad (6.9e)$$

$$e_1(t) = -(4 \cos(2t) \sin(t/5)). \quad (6.9f)$$

Six output equations with constant output matrices are tested with system (6.9). The output matrices considered are

$$C_a = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \quad C_b = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \quad C_c = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 1 \end{bmatrix},$$

$$C_d = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}, \quad C_e = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}, \quad C_f = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}.$$

System (6.9) is observable for each of these output matrices.

The time interval on which the completions are solved and the observers are constructed begins at $t_0 = 0$. Since the voltage source evaluated at t_0 is 0, $V(0) = 4 \cos(0) \sin(0) = 0$, state variables $e_1(0) = 0$ using equation (6.1f) and $i_{r_2}(0) = 0$ using equation (6.6). By letting $e_2(0) = 0$, state variable $i_{r_1}(0) = R_1(0)^{-1} (V(0) + e_2(0)) = 0$ from equation (6.7). Additionally,

$$\begin{aligned} 0 &= L(0)' i_l(0) + L(0) i_l(0)' - e_2(0) \\ \implies 0 &= \exp(0) i_l(0) + (2 - \exp(0)) i_l(0)' \\ \implies 0 &= i_l(0) + i_l(0)' \end{aligned}$$

indicates $i_l(0)$ can also be taken as 0. Evaluating $i_v(t)$ at t_0 reveals

$$\begin{aligned} i_v(0) &= -(C_1(0)' + R_1(0)^{-1} + R_2(0)^{-1}) V(0) - C_1(0)V(0)' - R_1(0)^{-1}e_2(0) \\ &= -(3 + \cos(0))(-8\sin(0)\sin(0) + (4/5)\cos(0)\cos(0)) \\ &= -16/5. \end{aligned}$$

Thus, $\tilde{x}(0) = [0 \ 0 \ 0 \ 0 \ 0 \ -16/5]^T$ is selected as the consistent initial condition when solving the completions. A program check that tests if the initial condition is consistent plots $-\mathcal{G}(t)\mathcal{F}(t)\tilde{x}(t) + \mathcal{G}(t)\mathcal{B}(t)v(t)$ from the characterization of the solution manifold. Since $\mathcal{G}(t)$, $\mathcal{F}(t)$, $\mathcal{B}(t)$, and $v(t)$ are defined from known matrices or vectors, a nonzero difference indicates completion $\tilde{x}(t)$ is incorrect. Except for the desire that $\hat{x}(0) \neq \tilde{x}(0)$, vector $\hat{x}(0) = [1 \ 2 \ 3 \ 4 \ 5 \ 6]^T$ is an arbitrary initial condition for the full-order observer and is transformed accordingly for the reduced-order and maximally reduced observers.

Analysis in Section 6.1 indicated the solution manifold is two-dimensional for circuit example (6.1). Thus, matrix $\mathcal{G}(t)$ from $0 = -\mathcal{G}(t)\mathcal{F}(t)\tilde{x}(t) + \mathcal{G}(t)\mathcal{B}(t)v(t)$ is expected to have rank 4. Each extended output matrix for constructing the maximally reduced observers in this chapter has structure $C_{\Xi} = \begin{bmatrix} \mathcal{G}(t)\mathcal{F}(t) \\ C_x \end{bmatrix}$. All of time-dependent $\mathcal{G}(t)\mathcal{F}(t)$ is saved and the extension C_x comes from the rows of the constant output matrices. Recall from Section 3.4, our algorithm defines $\mathcal{G}(t)$ as the transpose of the columns forming an orthonormal basis for $N(\mathcal{E}(t)^T)$. If Maple's NullSpace command from its LinearAlgebra package is used to determine $\mathcal{G}(t)$, then

$$\mathcal{G}(t)\mathcal{F}(t) = \begin{bmatrix} -1 & 1 & 0 & -(4 + 2\sin(t)) & 0 & 0 \\ 1 & 0 & 0 & 0 & -(2 + \sin(t)) & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ \frac{\sin(t/3)}{3(3+\cos(t/3))} & 0 & 0 & \frac{1}{3+\cos(t/3)} & \frac{-1}{3+\cos(t/3)} & \frac{1}{3+\cos(t/3)} \end{bmatrix}. \quad (6.10)$$

When $\bar{R}(t) = \begin{bmatrix} \mathcal{G}(t)\mathcal{F}(t) \\ C_x \\ C_{\bar{R}}(t) \end{bmatrix}$ is defined with the rows of (6.10), the inverse of $\bar{R}(t)$ required for the transformation $\hat{x}_{\bar{R}}(t) = \bar{R}(t)^{-1}\hat{q}(t)$ is undefined at certain times. Some elements of $\bar{R}(t)^{-1}$ have products in their denominators involving the term $\sin(t/3)$. Instead, MATLAB's null command returns a $\mathcal{G}(t)$ for a symbolically defined t such that

$$\mathcal{G}(t)\mathcal{F}(t) = \begin{bmatrix} -1 & 1 & 0 & -(4 + 2\sin(t)) & 0 & 0 \\ 1 & 0 & 0 & 0 & -(2 + \sin(t)) & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ -(1/3)\sin(t/3) & 0 & 0 & -1 & 1 & -1 \end{bmatrix}. \quad (6.11)$$

This full row rank matrix does not result in any computations with a zero in the denominator, so its rows are used in the construction of the extended output matrix for the maximally reduced observer. If an output matrix C with rank ≥ 2 has at least two rows linearly independent from the rows of (6.11) for all time, then enough information is known to determine the state variables without constructing an observer. By letting C_x equal these linearly independent rows of C , the extended output matrix $\begin{bmatrix} \mathcal{G}(t)\mathcal{F}(t) \\ C_x \end{bmatrix}$ is nonsingular. With $y_x(t) = C_x\tilde{x}(t)$ and

$$\mathcal{G}(t)\mathcal{B}(t) = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ -1 & 0 & 0 \\ 0 & 3 + \cos(t/3) & 0 \end{bmatrix}, \quad (6.12)$$

$x(t)$ can be solved for explicitly in the extended output equation:

$$x(t) = \begin{bmatrix} \mathcal{G}(t)\mathcal{F}(t) \\ C_x \end{bmatrix}^{-1} \begin{bmatrix} \mathcal{G}(t)\mathcal{B}(t)v(t) \\ y_x(t) \end{bmatrix}. \quad (6.13)$$

If output data is available, $y_x(t)$ will come from the data rather than $C_x\tilde{x}(t)$, so the solution $x(t)$ of the linear time-varying system of DAEs can be calculated using equation (6.13) without first computing a completion. If $C_{\Xi}(t)$ is not full row rank, then $C_{\bar{R}}(t)$ from constructing transformation matrix $\bar{R}(t)$ is determined in a symbolic environment so the time-varying definition of $\bar{R}(t)$ can be differentiated the desired number of times. Since $\mathcal{G}(t)\mathcal{F}(t)$ and a time-dependent $C_{\bar{R}}(t)$ are hard coded, a program check tests the rank of $\bar{R}(t)$ at each time t to confirm the transformation matrix remains nonsingular.

As with the linear time-invariant example, ‘full-order system’ and ‘completion with output equation’ are interchangeable and reduced-order system describes the unmeasurable subsystem or the system associated with pair $(A_{R_{22}}(t), A_{R_{12}}(t))$. Subscript notation is again included, identifying a particular completion’s pair or stabilization parameter matrix: $(\cdot, \cdot)_{SL}$ and Λ_{SL} for the stabilized least squares completion; $(\cdot, \cdot)_A$ and Λ_A for the alternative stabilized completion; and $(\cdot, \cdot)_L$ and Λ_L for the least squares completion.

Λ Selection

The discussion in Section 3.6 on linear time-varying completion stability and observability proposed differentiating with a stabilization parameter matrix Λ instead of a scalar λ . When selecting a stabilization parameter matrix for either the stabilized least squares completion (Λ_{SL}) or the alternative stabilized completion (Λ_A), we check four criteria.

1. $\|\Phi(t, 0)\|$: Recall, $\Phi(t, 0)$ is the state transition matrix, and for the general completion $\tilde{x}(t)' = \tilde{A}(t)\tilde{x}(t) + \tilde{B}(t)v(t)$, $\Phi(t, 0)' = \tilde{A}(t)\Phi(t, 0)$ with $\Phi(0, 0) = I$. If $\|\Phi(t, 0)\| \rightarrow 0$ as $t \rightarrow \infty$, then the completion and its additional dynamics are asymptotically stable [4]. Article [83] offers an alternative characterization of asymptotically stable not considered in this dissertation.
2. $\|\tilde{A}(t)\|$ and $\|\tilde{B}(t)\|$: If $\|\tilde{A}(t)\| < \infty$ and $\|\tilde{B}(t)\| < \infty$, then $\tilde{A}(t)$ and $\tilde{B}(t)$ are bounded, a requirement for observer construction.
3. The real part of matrix $\tilde{A}(t)$'s eigenvalues: Having eigenvalues with negative real parts for all time is not a requirement for observer construction in the linear time-varying case but may increase the chances of a stabilized completion being detectable rather than unobservable. Since the solution manifold of system (6.9) is two-dimensional, if this example were linear time-invariant, four out of six eigenvalues of \tilde{A} would come from the completions' additional dynamics. Thus, this criterion checks how many of $\tilde{A}(t)$'s eigenvalues out of four have a negative real part for the time intervals under consideration. Notice, this criterion is not a stability check as the signs of the eigenvalues are unrelated to a system's stability in the time-varying case [170].
4. $\text{rank}(W_o(t))$ for $(\tilde{A}(t), C)$: This rank check for the full-order systems is used to evaluate how the stabilization parameter matrix being considered affects the systems' observability.

Criteria 3 and 4 are a result of the discussion in Sections 2.4, 3.6, and 3.1. The assessment of five stabilization parameter matrices for each completion is included. The list assigned to each constant diagonal matrix Λ is its distinct entries in a specific order: $\Lambda = \{\Lambda(1, 1), \Lambda(2, 2), \dots\}$.

STABILIZED LEAST SQUARES COMPLETION:

$$\begin{aligned}\Lambda_{1\text{SL}} &= \{1.0, 1.0, 1.0, 1.0, 1.0, 1.0\}, & \Lambda_{2\text{SL}} &= \{1.0, 1.5, 2.0, 2.5, 3.0, 3.5\}, \\ \Lambda_{3\text{SL}} &= \{1.5, 2.0, 2.5, 3.0, 3.5, 4.0\}, & \Lambda_{4\text{SL}} &= \{1.2, 1.4, 1.6, 1.8, 2.0, 2.2\}, \\ \Lambda_{5\text{SL}} &= \{2.2, 2.0, 1.8, 1.6, 1.4, 1.2\}\end{aligned}$$

1. On the extended time interval $t \in [0, 135]$, Figure 6.2a shows that $\|\Phi(t, 0)\| \rightarrow 0$ for the five stabilized least squares completions being considered. The norms converge to one another around $t = 15$, and by $t = 92$, $\|\Phi(t, 0)\|$ has magnitude $\times 10^{-4}$. Figure 6.2b is included for a closer view of the norms on the $t \in [0, 10]$ interval. The largest magnitudes occur for $\Lambda_{3\text{SL}}$ and $\Lambda_{2\text{SL}}$, but $\|\Phi(t, 0)\|$ takes longer to converge to the other norms for $\Lambda_{1\text{SL}}$ and $\Lambda_{5\text{SL}}$.
2. Matrices $\tilde{A}(t)$ and $\tilde{B}(t)$ are bounded for all five choices of Λ_{SL} . Notice the vertical scales are not uniform in Figures 6.3 and 6.4. The smallest upper and lower bounds of $\|\tilde{A}(t)\|$ and the smallest difference between these bounds occur for $\Lambda_{1\text{SL}}$ in Figure 6.3a. The resulting $\|\tilde{A}(t)\|$

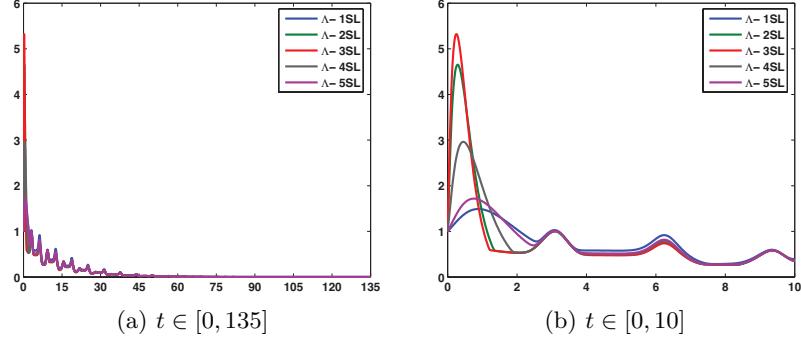


Figure 6.2: $\|\Phi(t, 0)\|$ on two intervals of time. $\|\Phi(t, 0)\| \rightarrow 0$ for each choice of Λ_{SL} .

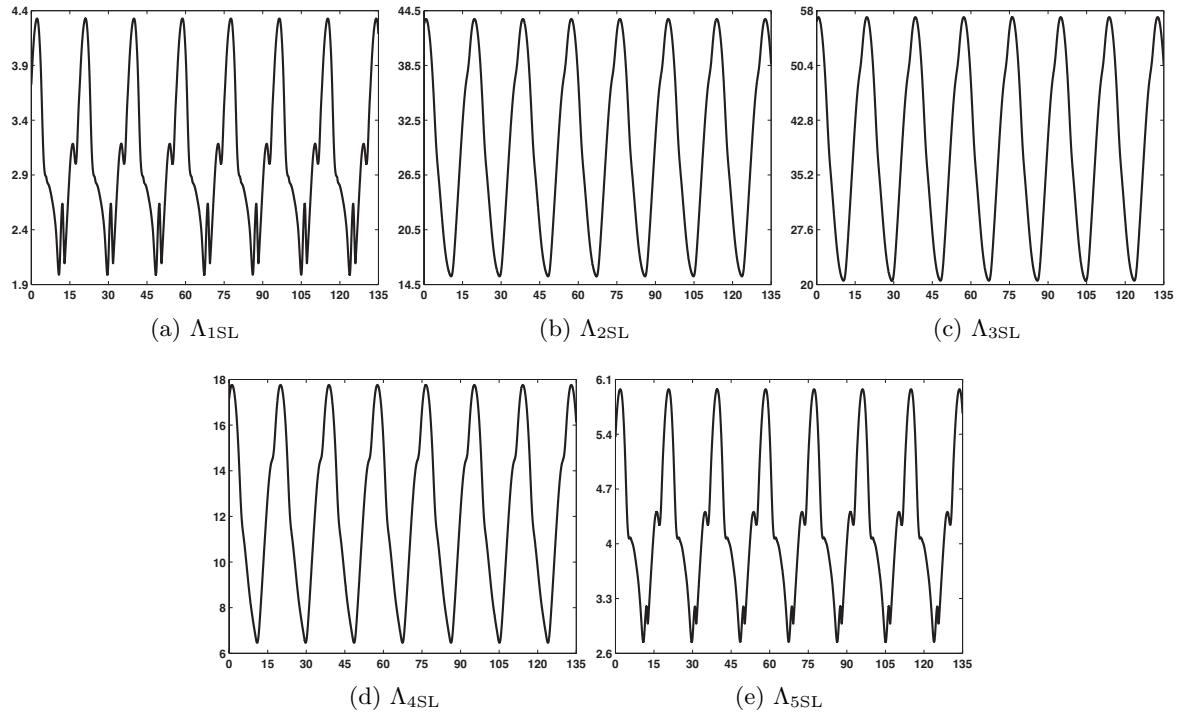


Figure 6.3: $\|\tilde{A}(t)\|$ for five choices of Λ_{SL} . Each $\|\tilde{A}(t)\|$ is bounded.

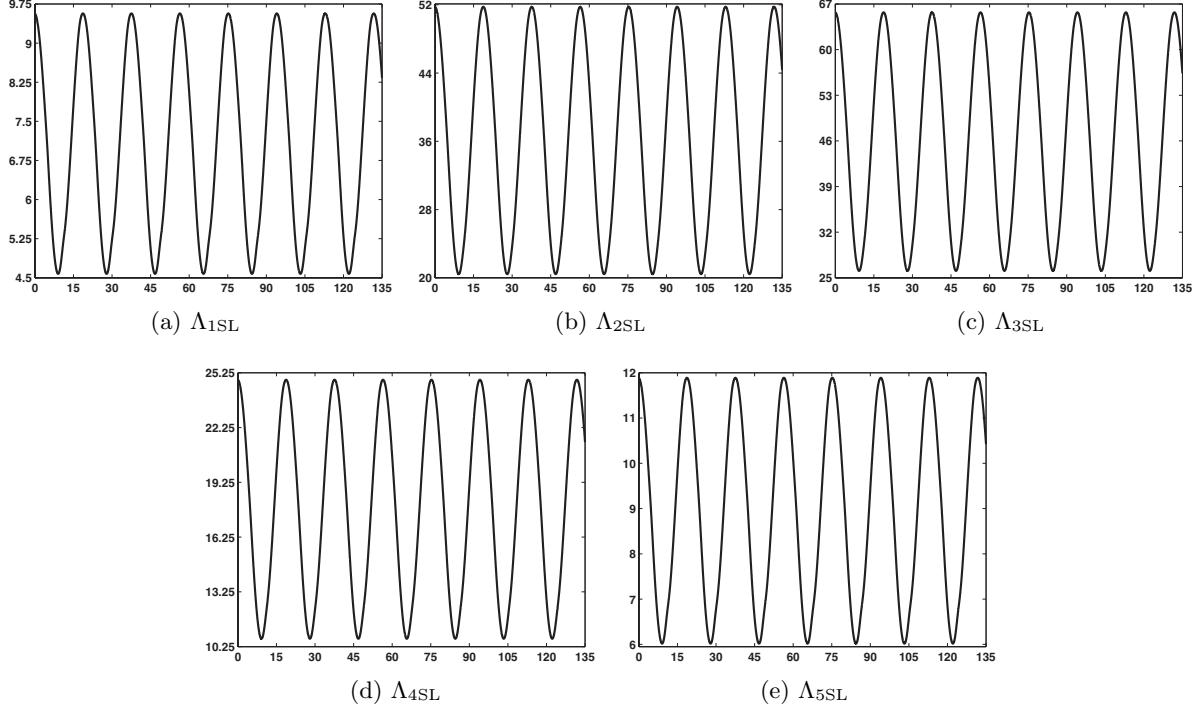


Figure 6.4: $\|\tilde{B}(t)\|$ for five choices of Λ_{SL} . Each $\|\tilde{B}(t)\|$ is bounded.

for $\Lambda_{5\text{SL}}$ in Figure 6.3e has a similar amplitude, and for both norms, the oscillating curve is not smooth. In Figure 6.3c, the norm of $\tilde{A}(t)$ resulting from $\Lambda_{3\text{SL}}$ is a smooth oscillating curve but has the greatest amplitude and largest upper bound. The order of $\|\tilde{A}(t)\|$ from smallest to largest in terms of the stabilization parameter matrices is $\{\Lambda_{1\text{SL}}, \Lambda_{5\text{SL}}, \Lambda_{4\text{SL}}, \Lambda_{2\text{SL}}, \Lambda_{3\text{SL}}\}$. This smallest to largest order represented by the Λ_{SL} matrices holds when comparing $\|\tilde{B}(t)\|$ in Figure 6.4 as well as $\|\Phi(t, 0)\|$ in Figure 6.2.

3. The number of $\tilde{A}(t)$'s eigenvalues with a negative real part is presented in Table 6.1 for each Λ_{SL} . Matrix $\Lambda_{1\text{SL}}$ produces the desired result on the extended time interval $t \in [0, 135]$, while four eigenvalues with a negative real part holds on the shorter $t \in [0, 45]$ interval for $\Lambda_{4\text{SL}}$. For each Λ_{SL} being considered, the real part of every eigenvalue oscillates and is periodic. When using $\Lambda_{4\text{SL}}$, the maximum values for three periods of the eigenvalue are 0.005275 at $t = 78.46$, 0.004334 at $t = 97.31$, and 0.003391 at $t = 116.20$. Although the maximum values of the earlier periods are just less than zero, it is possible that a finer time step would reveal the earlier periods also have a positive maximum value that did not correspond with a multiple of time step 0.01. The other three choices of Λ_{SL} cause just two eigenvalues resulting from additional dynamics to have a negative real part.

Table 6.1: For each Λ_{SL} , the number of $\tilde{A}(t)$'s eigenvalues with a negative real part.

	$t \in [0, 45]$	$t \in [0, 135]$
$\Lambda_{1\text{SL}}$	4	4
$\Lambda_{2\text{SL}}$	2	2
$\Lambda_{3\text{SL}}$	2	2
$\Lambda_{4\text{SL}}$	4	3
$\Lambda_{5\text{SL}}$	2	2

Table 6.2: For each Λ_{SL} , the ranks of $W_o(t)$ and their time intervals for two output matrices.

rank ($W_o(t)$)	C_a		C_d	
	5	6	4	5
$\Lambda_{1\text{SL}}$		0.01 to 135.00	0.01 to 135.00	
$\Lambda_{2\text{SL}}$	0.01	0.02 to 135.00	0.01 to 0.16	0.17 to 135.00
$\Lambda_{3\text{SL}}$	0.01	0.02 to 135.00	0.01 to 0.18	0.19 to 135.00
$\Lambda_{4\text{SL}}$	0.01	0.02 to 135.00	0.01 to 0.31	0.32 to 135.00
$\Lambda_{5\text{SL}}$	0.01	0.02 to 135.00	0.01 to 0.30	0.31 to 135.00

4. The rank of $W_o(t)$ for the pairs $(\tilde{A}(t), C_a)_{\text{SL}}$, $(\tilde{A}(t), C_d)_{\text{SL}}$, and $(\tilde{A}(t), C_c)_{\text{SL}}$ demonstrates how different Λ_{SL} choices cause observability to vary. The observability Gramians are determined by solving system (3.4). Due to the chosen 0.01 output time step for MATLAB's ode45 solver, the rank is calculated at these times as well. Tables 6.2 and 6.3 summarize the rank changes as time progresses on the interval $t \in (0, 135]$.

In Table 6.2, the pair $(\tilde{A}(t), C_a)_{\text{SL}}$ resulting from $\Lambda_{1\text{SL}}$, the stabilization parameter matrix without distinct diagonal entries, is observable. The other four Λ_{SL} matrices produce completions that are also observable with output matrix C_a since the rank of their observability Gramians reaches 6 in finite time (by $t = 0.02$). If, for example, the output time step had been 0.1 rather than 0.01, all five of these pairs would have $\text{rank} (W_o(t)) = 6$ for the entire interval. Each of the five stabilized least squares completions with output equation $\tilde{y}(t) = C_d \tilde{x}(t)$ is not observable. However, using a stabilization parameter matrix with distinct diagonal entries decreases the dimension of the unobservable subspace, thereby increasing the chances of having a detectable system. Also, the time at which the rank of the observability Gramian reaches 5 is later for $\Lambda_{4\text{SL}}$ and $\Lambda_{5\text{SL}}$ than for either $\Lambda_{2\text{SL}}$ or $\Lambda_{3\text{SL}}$.

The observability Gramians resulting from $\Lambda_{1\text{SL}}$ and $\Lambda_{5\text{SL}}$ are not well-conditioned when using output matrix C_c . For matrix $\Lambda_{1\text{SL}}$, the rank of $W_o(t)$ is 5 from 0.15 to 0.49, 6 from 0.50 to 1.17, 5 from 1.18 to 1.21, and then 6 for the remaining interval. Similarly, after having rank 5 from 0.16 to 0.44, rank ($W_o(t)$) for $\Lambda_{5\text{SL}}$ increases to 6 for $t = 0.45$ and $t = 0.46$ before

Table 6.3: For each Λ_{SL} , the ranks of $W_o(t)$ and their time intervals for output matrix C_c .

rank ($W_o(t)$)	C_c				
	2	3	4	5	6
$\Lambda_{1\text{SL}}$	0.01	0.02 to 0.05	0.06 to 0.14	*	*
$\Lambda_{2\text{SL}}$	0.01	0.02 to 0.10	0.11 to 0.42	0.43 to 0.65	0.66 to 135.00
$\Lambda_{3\text{SL}}$	0.01	0.02 to 0.09	0.10 to 0.43	0.44 to 0.73	0.74 to 135.00
$\Lambda_{4\text{SL}}$	0.01	0.02 to 0.14	0.15 to 0.31	0.32 to 0.53	0.54 to 135.00
$\Lambda_{5\text{SL}}$	0.01	0.02 to 0.07	0.08 to 0.15	*	*

decreasing to 5 from 0.47 to 0.93. At $t = 0.94$ the rank of the observability Gramian for the stabilized least squares completion with $\Lambda_{5\text{SL}}$ and output matrix C_c reaches 6 and remains there. Table 6.3 shows this fluctuation in the rank of $W_o(t)$ does not occur for stabilization parameter matrices $\Lambda_{2\text{SL}}$, $\Lambda_{3\text{SL}}$, and $\Lambda_{4\text{SL}}$. Of these three choices, it takes the least amount of time for the rank of the observability Gramian to reach 6 when stabilizing with $\Lambda_{4\text{SL}}$.

Λ_{SL} SELECTION: The fluctuation of the observability Gramian's rank in Criterion 4 leads us away from choosing either $\Lambda_{1\text{SL}}$ or $\Lambda_{5\text{SL}}$. Therefore, even though matrix $\Lambda_{1\text{SL}}$ produces the desired result for Criterion 3, $\Lambda_{4\text{SL}}$ becomes the preferred stabilization parameter matrix when considering Criteria 3 and 4 together. Since $\|\Phi(t, 0)\| \rightarrow 0$, $\|\tilde{A}(t)\| < \infty$, and $\|\tilde{B}(t)\| < \infty$ for each Λ_{SL} , Criteria 1 and 2 do not exclude any of the five options. However, the norm of $\tilde{A}(t)$ associated with $\Lambda_{4\text{SL}}$ has the smallest upper bound of the three smooth norms in Figure 6.3. The behavior of this $\tilde{A}(t)$ may prove beneficial when trying to manage the growth of $S_L(t)$ from the Riccati equation used to compute the gain matrices. Taking all of these criteria into consideration, our chosen stabilization parameter matrix to compute the stabilized least squares completion is $\Lambda_{4\text{SL}}$.

ALTERNATIVE STABILIZED COMPLETION:

$$\begin{aligned}\Lambda_{1A} &= \{1.4, 1.6, 1.8, 2.0\}, & \Lambda_{2A} &= \{2.5, 3.0, 3.5, 4.0\}, \\ \Lambda_{3A} &= \{3.2, 3.4, 3.6, 3.8\}, & \Lambda_{4A} &= \{4.0, 3.5, 3.0, 2.5\}, \\ \Lambda_{5A} &= \{4.0, 4.0, 4.0, 4.0\}\end{aligned}$$

- On the extended time interval $t \in [0, 135]$, Figure 6.5a shows that $\|\Phi(t, 0)\| \rightarrow 0$ for the five alternative stabilized completions being considered. Figure 6.5b is included for a closer view of the norms on the $t \in [0, 35]$ interval. While four of the norms are approximately the same, at times $\|\Phi(t, 0)\|$ resulting from Λ_{1A} has a slightly larger magnitude until it converges to the other norms around $t = 30$. By $t = 89$, $\|\Phi(t, 0)\|$ has magnitude $\times 10^{-4}$.
- Matrices $\tilde{A}(t)$ and $\tilde{B}(t)$ are bounded for all five choices of Λ_A . Figure 6.6 shows that the upper and lower bounds vary for each $\|\tilde{A}(t)\|$. The smallest bounds of $\|\tilde{A}(t)\|$ occur for Λ_{1A}

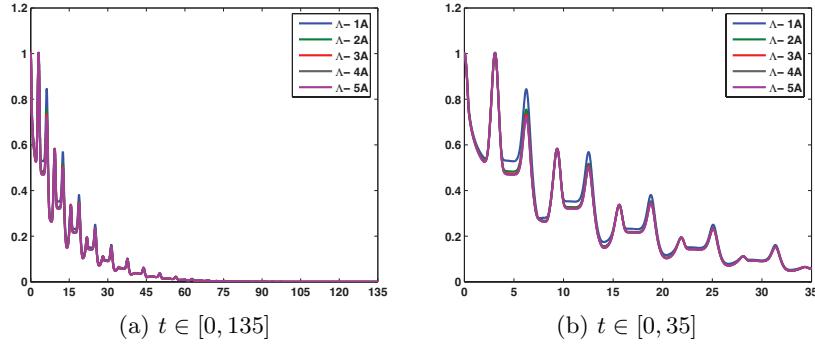


Figure 6.5: $\|\Phi(t, 0)\|$ on two intervals of time. $\|\Phi(t, 0)\| \rightarrow 0$ for each choice of Λ_A .

in Figure 6.6a but the smallest difference between the upper and lower bounds occurs for Λ_{2A} in Figure 6.6b. Furthermore, the only smooth $\|\tilde{A}(t)\|$ corresponds with stabilization parameter matrix Λ_{2A} . In Figure 6.6d, the norm of $\tilde{A}(t)$ resulting from Λ_{4A} has the largest upper bound and the largest difference between its bounds. In terms of the stabilization parameter matrices, the smallest to largest order of $\|\tilde{A}(t)\|$ based on the upper bounds is $\{\Lambda_{1A}, \Lambda_{3A}, \Lambda_{2A}, \Lambda_{5A}, \Lambda_{4A}\}$.

Recall, the derivation of the time-varying alternative stabilized completion in Subsection 2.3.2 did not define $\tilde{B}(t)$ independent of the input $u(t)$ and its $k + 1$ derivatives (see equation (2.11)). Therefore, the norm of $\tilde{B}(t)$ cannot be computed, but due to this circuit example system's bounded coefficient matrices

$$E(t) = \begin{bmatrix} 3 + \cos(t/3) & 0 & 0 & 0 & 0 & 0 \\ 0 & 2 - \cos(2t) & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 - \exp(-t) & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \quad B(t) = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ -1 \end{bmatrix},$$

$$F(t) = \begin{bmatrix} -(1/3) \sin(t/3) & 0 & 0 & -1 & 1 & -1 \\ 0 & 2 \sin(2t) & 1 & 1 & 0 & 0 \\ 0 & -1 & \exp(-t) & 0 & 0 & 0 \\ -1 & 1 & 0 & -(4 + 2 \sin(t)) & 0 & 0 \\ 1 & 0 & 0 & 0 & -(2 + \sin(t)) & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

and their bounded derivatives as well as matrix $E(t)$'s constant rank, matrix $\tilde{B}(t)$ is bounded. Figure 6.7 is included as a reference for comparing the stabilized completions' $\|\tilde{B}(t)v(t)\|$. The

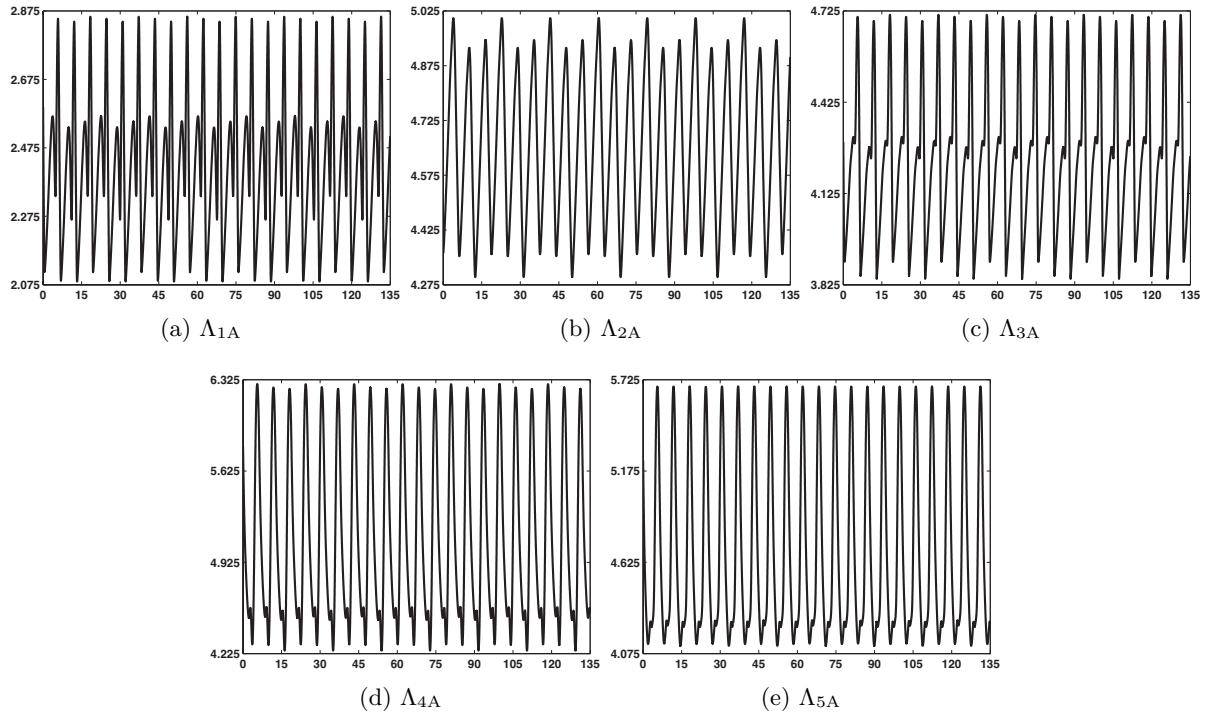


Figure 6.6: $\|\tilde{A}(t)\|$ for five choices of Λ_A . Each $\|\tilde{A}(t)\|$ is bounded.

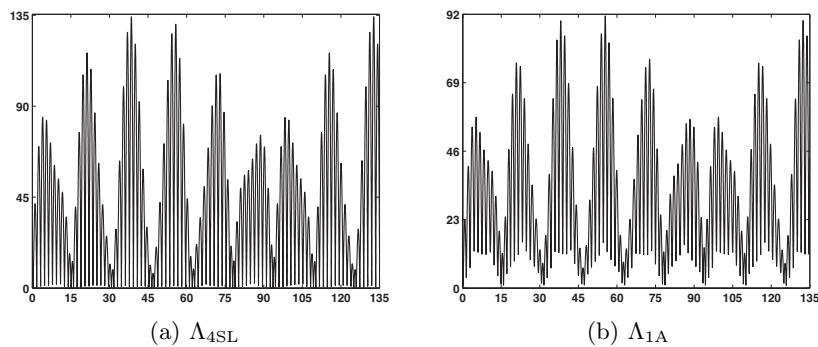


Figure 6.7: $\|\tilde{B}(t)v(t)\|$ for an SLSC and an ASC. $\|\tilde{B}(t)\|$ cannot be computed for the ASC, but comparing $\|\tilde{B}(t)v(t)\|$ for the stabilized completions suggests the ASC $\|\tilde{B}(t)\|$ is bounded.

Table 6.4: For each Λ_A , the number of $\tilde{A}(t)$'s eigenvalues with a negative real part.

	$t \in [0, 45]$	$t \in [0, 135]$
Λ_{1A}	3	3
Λ_{2A}	3	3
Λ_{3A}	3	3
Λ_{4A}	4	4
Λ_{5A}	4	4

norm of $\tilde{B}(t)v(t)$ for the stabilized least squares completion with Λ_{4SL} is displayed in Figure 6.7a. Knowing the behavior and magnitude of $\|\tilde{B}(t)\|$ for this stabilized least squares completion from Figure 6.4d, it is possible to visually deduce but not theoretically conclude from the figures that $\|\tilde{B}(t)\|$ is bounded for the alternative stabilized completions (using Λ_{1A} as an example) due to the similar results in Figures 6.7a and 6.7b.

3. The number of $\tilde{A}(t)$'s eigenvalues with a negative real part is presented in Table 6.4 for each Λ_A . Using the general description $\Lambda_A = \{\lambda_1, \lambda_2, \lambda_3, \lambda_4\}$, matrix $\tilde{A}(t)$ from the alternative stabilized completion of system (6.9) has two eigenvalues equaling $-\lambda_3$ and $-\lambda_4$. For example, two of the eigenvalues of $\tilde{A}(t)$ when stabilizing with Λ_{1A} are -1.8 and -2.0 . Therefore, each $\tilde{A}(t)$ will have at least two eigenvalues with negative real parts. The other four eigenvalues have real parts that oscillate and are periodic.

The desired eigenvalue behavior occurs when stabilizing the additional dynamics with Λ_{4A} and Λ_{5A} . Matrix $\tilde{A}(t)$ for each of the remaining choices of Λ_A has one eigenvalue with a negative real part in addition to the two eigenvalues that are already known.

4. Output matrices C_a , C_d , and C_c are again chosen to demonstrate how the choice of stabilization parameter matrix can affect the rank of the observability Gramian. Tables 6.5 and 6.6 summarize the rank changes over the interval $t \in (0, 135]$.

In Table 6.5, the pair $(\tilde{A}(t), C_a)_A$ is observable and has rank 6 beginning at $t = 0.01$ when stabilizing with either Λ_{2A} or Λ_{4A} . Stabilization parameter matrices Λ_{1A} , Λ_{3A} , and Λ_{5A} also produce completions that are observable with output matrix C_a , but their observability Gramians do not reach full row rank until $t = 0.02$, $t = 0.02$, and $t = 0.10$, respectively. All five alternative stabilized completions with output equation $\tilde{y}(t) = C_d\tilde{x}(t)$ are not observable. Again, the rank of the observability Gramian is greater when using a stabilization parameter matrix with distinct rather than repeated diagonal entries.

The only observability Gramian that is well-conditioned when using output matrix C_c results from Λ_{4A} . Otherwise, for matrix Λ_{1A} , the rank of $W_o(t)$ is 5 from 0.17 to 0.47, 6 from 0.48 to 0.95, 5 at $t = 0.96$, and then 6 for the remaining interval. The rank also fluctuates between 5 and 6 when stabilizing with Λ_{2A} . After having rank 5 from 0.14 to 0.54 and rank 6 from 0.55 to

Table 6.5: For each Λ_A , the ranks of $W_o(t)$ and their time intervals for two output matrices.

rank ($W_o(t)$)	C_a		C_d	
	5	6	3	4
Λ_{1A}	0.01	0.02 to 135.00		0.01 to 135.00
Λ_{2A}		0.01 to 135.00		0.01 to 135.00
Λ_{3A}	0.01	0.02 to 135.00		0.01 to 135.00
Λ_{4A}		0.01 to 135.00		0.01 to 135.00
Λ_{5A}	0.01 to 0.09	0.10 to 135.00	0.01 to 135.00	

Table 6.6: For each Λ_A , the ranks of $W_o(t)$ and their time intervals for output matrix C_c .

rank ($W_o(t)$)	C_c				
	2	3	4	5	6
Λ_{1A}	0.01	0.02 to 0.05	0.06 to 0.16	*	*
Λ_{2A}	0.01	0.02 to 0.04	0.05 to 0.13	*	*
Λ_{3A}	0.01	0.02 to 0.09	*	*	0.90 to 135.00
Λ_{4A}	0.01	0.02 to 0.11	0.12 to 0.37	0.38 to 0.68	0.69 to 135.00
Λ_{5A}	0.01	*	*	0.52 to 0.97	0.98 to 135.00

0.88, the rank of the observability Gramian decreases to 5 for $t \in [0.89, 0.94]$ before becoming full row rank for the rest of the interval. For stabilization parameter matrix Λ_{3A} , $\text{rank} (W_o(t))$ equals 4 from 0.10 to 0.14, increases to 5 from 0.15 to 0.50, decreases to 4 for $t = 0.51$ and $t = 0.52$, and then increases to 5 again for another interval before becoming 6 at $t = 0.90$. The final ill-conditioned observability Gramian occurs for Λ_{5A} . The rank is 3 on $t \in [0.02, 0.07]$ and $t \in [0.11, 0.13]$ but alternates to 4 on $t \in [0.08, 0.10]$ and $t \in [0.14, 0.51]$.

Λ_A SELECTION: The fluctuation of the observability Gramian's rank in Criterion 4 leads us toward choosing Λ_{4A} . Therefore, even though matrix Λ_{5A} produces the desired result for Criterion 3, Λ_{4A} becomes the preferred stabilization parameter matrix when considering Criteria 3 and 4 together. Again, Criteria 1 and 2 do not exclude any of the five options since $\|\Phi(t, 0)\| \rightarrow 0$, $\|\tilde{A}(t)\|$ is finite, and $\|\tilde{B}(t)\|$ is finite for each Λ_A . Although 6.290758 is the largest upper bound and 2.043183 is the largest difference between bounds, the norm of $\tilde{A}(t)$ associated with Λ_{4A} is still very close in bounds and in amplitude to the other possible stabilization parameter matrices. Taking all of these criteria into consideration, the alternative stabilized completion is computed with stabilization parameter matrix Λ_{4A} .

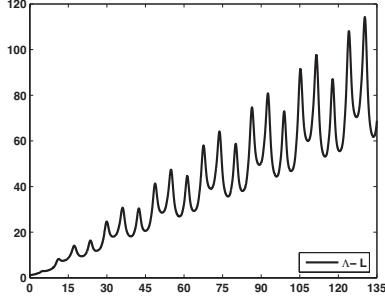


Figure 6.8: $\|\Phi(t, 0)\|$ fails to converge to zero for the LSC (Λ_L).

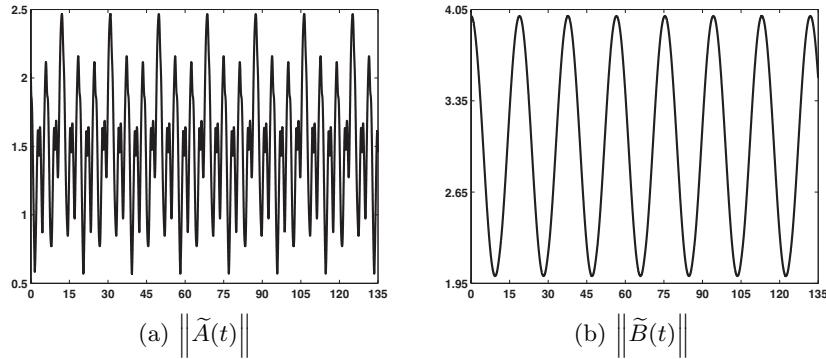


Figure 6.9: $\|\tilde{A}(t)\|$ and $\|\tilde{B}(t)\|$ are bounded for the LSC (Λ_L).

LEAST SQUARES COMPLETION:

As described in Sections 2.1 and 2.2, the algorithms for determining the least squares completion and the stabilized least squares completion are interchangeable except for the differential polynomial used to construct the derivative array equations. Thus, the same MATLAB code developed for the stabilized least squares completion can be implemented for the least squares completion if Λ is defined as a zero matrix ($\Lambda_L = \{0, 0, 0, 0, 0, 0\}$) since stabilized differentiation does not occur.

1. Section 3.2.2 highlighted some articles that differ on whether or not an exponentially asymptotically stable $\tilde{A}(t)$ is required for estimating the state. Coefficient matrix $\tilde{A}(t)$ from the least squares completion of system (6.9) is not exponentially asymptotically stable. On the extended time interval $t \in [0, 135]$, Figure 6.8 shows that $\|\Phi(t, 0)\|$ does not converge to zero.
2. Figure 6.9 reveals the coefficient matrices of the least squares completion are bounded.
3. None of $\tilde{A}(t)$'s eigenvalues have real parts that remain negative for all of $t \in [0, 45]$.

4. When considering output matrices C_a , C_d , and C_c , the observability Gramian for pair $(\tilde{A}(t), C_a)_L$ is the only one that is well-conditioned. This pair is observable since the rank of $W_o(t)$ equals 6 after having rank 5 at $t = 0.01$ and $t = 0.02$. The rank of $W_o(t)$ for pair $(\tilde{A}(t), C_d)_L$ is 4 from 0.01 to 24.18 and 5 from 24.19 to 24.23. Then the rank decreases to 4 before becoming 5 again at $t = 24.29$ and remaining there until $t = 135.00$. For output matrix C_c , the rank of the observability Gramian equals 2 at $t = 0.01$; 3 from 0.02 to 0.06; 4 for $t = 0.07$ and $t = 0.08$; and then 5 from 0.09 to 0.12, from 0.14 to 0.39, and from 0.44 to 0.54. At $t = 0.13$, the rank decreases to 4 but from 0.40 to 0.43 the rank increases to 6. For the rest of the interval beginning at $t = 0.55$, $W_o(t)$ of pair $(\tilde{A}(t), C_c)_L$ has rank 6.

Results

For the remaining figures in Subsection 6.2.1, each color corresponds with a particular state variable: blue for $e_1(t)$, green for $e_2(t)$, red for $i_l(t)$, gray for $i_{r_1}(t)$, magenta for $i_{r_2}(t)$, and brown for $i_v(t)$. Figures with a single black solution are of norms. In captions, vs represents vertical scale; Sol stands for solution; Obs designates observer; and Diff identifies difference.

Figure 6.10 displays the solution of the stabilized least squares completion (SLSC solution) when stabilizing with Λ_{4SL} . The solution of system (6.9) is periodic, a behavior that cannot be concluded from Figure 6.10a but is visible on the extended time interval $t \in [0, 135]$ in Figure 6.10b. The solution of the alternative stabilized completion (ASC solution) when stabilizing with Λ_{4A} is shown on the extended time interval in Figure 6.11a. The difference in Figure 6.11b between these SLSC and ASC solutions (SLSC – ASC) remains within the interval $(-1.5 \times 10^{-6}, 1.5 \times 10^{-6})$ and does not grow. The solution of the least squares completion (LSC solution) in Figure 6.12a also appears to be the same as the SLSC solution in Figure 6.10b. The difference between these solutions (SLSC – LSC) in Figure 6.12b stays within the interval $(-1.5 \times 10^{-6}, 1.5 \times 10^{-6})$ for $t \in [0, 45]$ but increases as time progresses.

In all six-color figures, the layering of the lines from blue plotted first (on bottom) to brown plotted last (on top) can conceal results. The removal of the prominent brown difference in component $i_v(t)$ from Figures 6.11c and 6.12c uncovers that the behaviors of the differences in the remaining state variables are consistent with what is observed in their respective $i_v(t)$ components. The growth in the difference SLSC – LSC and the absence of growth in the near-zero difference SLSC – ASC implies the choices of Λ_{4SL} and Λ_{4A} are effectively stabilizing the additional dynamics.

The computation of the gain matrix requires the user to define three matrices: $S_L(t_0)$, $M(t)$, and $\bar{Q}(t)$. Initial condition $S_L(t_0)$ is defined as δI for some scalar δ and an $n \times n$ identity matrix. We decided to make $M(t)$ constant and defined it as ηI for some scalar η and an $m \times m$ identity matrix. Recall, $Q_L(t) = \bar{Q}(t) + C(t)^T M(t) C(t)$ for the full-order observer, $Q_L(t) = \bar{Q}(t) + A_{R_{12}}(t)^T M(t) A_{R_{12}}(t)$ for the reduced-order observer, and $Q_L(t) = \bar{Q}(t) + A_{\bar{R}_{12}}(t)^T M(t) A_{\bar{R}_{12}}(t)$

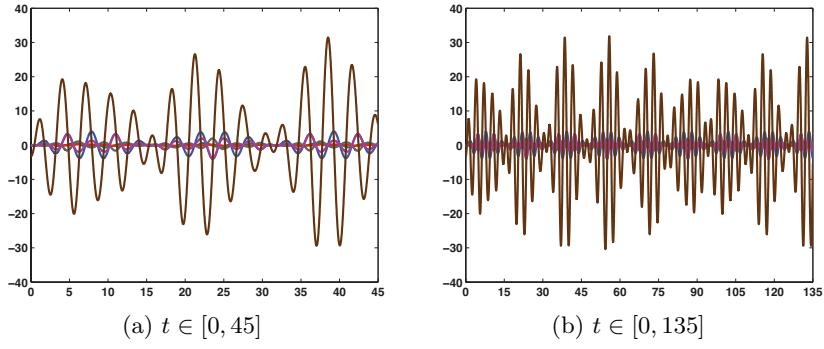


Figure 6.10: SLSC solution (Λ_{4SL}) on two intervals of time. Figure 6.10b suggests system (6.9)'s solution is periodic.

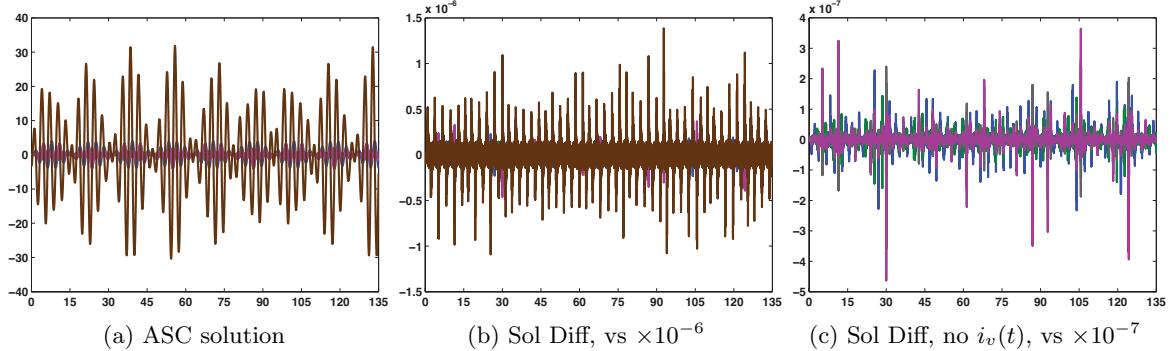


Figure 6.11: The ASC solution (Λ_{4A}) is comparable to the SLSC solution in Figure 6.10. The solution difference is SLSC – ASC.

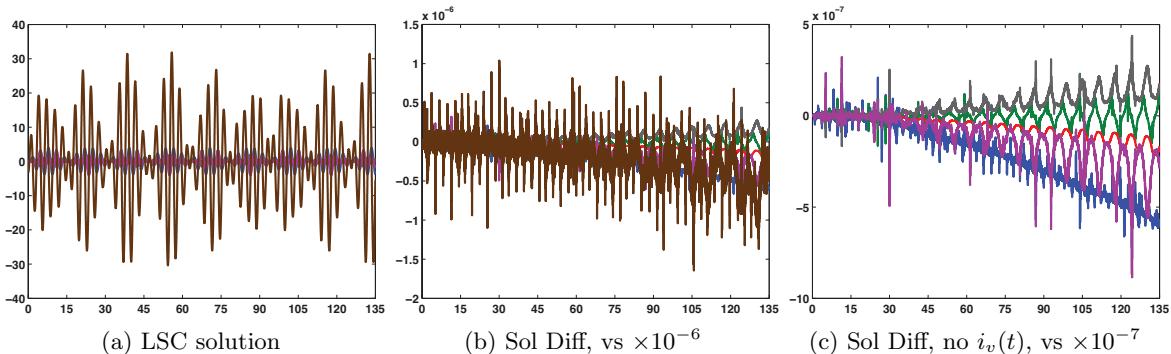


Figure 6.12: The LSC solution (Λ_L) is comparable to the SLSC solution in Figure 6.10. The solution difference is $SLSC - LSC$.

for the maximally reduced observer. $\bar{Q}(t)$ is taken to be ξI , with the identity matrix sized appropriately for the observer being used and ξ a scalar. Although $Q_L(t)$ is constant for the full-order observer since our choices for output matrix are time-invariant, $Q_L(t)$ may not be constant for the other observers if $A_{R_{12}}(t)$ or $A_{\bar{R}_{12}}(t)$ varies with time. The selection of δ , η , and ξ has been trial and error, but for the following results, $\delta = \xi = 0.1$ unless otherwise noted. The choices of η vary and are specified with each observer. The construction of the reduced-order and maximally reduced observers in the time-varying case also requires the selection of a nonsingular matrix $P_2(t)$. Since this matrix may be constant, we selected $P_2 = I$.

Case 1 (output matrix C_a) begins by assessing the full-order observers constructed using the stabilized least squares completion and the alternative stabilized completion. The effects of choosing different gain matrix parameters δ and ξ are also examined. Additionally, the alternative stabilized completions and their resulting full-order observers are shown to be comparable when using either the smooth decomposition method (SDM) or the symbolic method (SYM) in the ASC algorithm. The consequences from $\|\Phi(t, 0)\|$ failing to go to zero are realized while attempting to construct the full-order observer using the least squares completion. Next this case constructs reduced-order observers using each of the three completions, while the process for constructing the maximally reduced observer reveals the state of this circuit example can be calculated without an observer.

Case 2 (output matrices C_b and C_c) examines the results when the output is a linear combination of two state variables rather than two individual components. The observers presented for comparison are constructed using the stabilized least squares completion. This case concludes with a comment on the alternative stabilized completion and the selection of Λ_A .

Case 3 (output matrix C_d) attempts to construct observers using the stabilized completions when the systems to be observed are not observable.

Case 4 (output matrices C_e and C_f) investigates how outputting a state variable known from the input affects the estimates of the three observers. The results analysis for each completion, including a discussion on the detectability of the least squares completion, again reveals the benefits of estimating the state with the maximally reduced observer.

Case 1, C_a

From the results in Tables 6.2 and 6.5, the pairs $(\tilde{A}(t), C_a)_{SL}$ and $(\tilde{A}(t), C_a)_A$ are observable since their observability Gramians have rank 6 in finite time. Figure 6.13 presents the estimation error for the full-order observer constructed using the stabilized least squares completion (SLSC/FO observer) when $\eta = 1$. The SLSC/FO observer visibly fails to estimate $\tilde{x}(t)$ by $t_f = 45$ in Figure 6.13a. The difference between the SLSC solution and this observer at $t_f = 45$ is $\begin{bmatrix} 0.0 & -0.052800 & -0.036834 & -0.009260 & 0.0 & 0.009260 \end{bmatrix}^T$, so $e(t) \notin (-10^{-3}, 10^{-3})$ by the desired final time. However, this SLSC/FO observer appears to con-

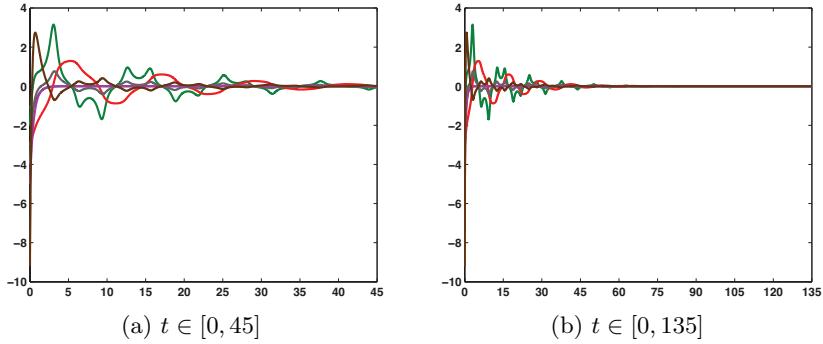


Figure 6.13: SLSC/FO observer's $\tilde{x}(t) - \hat{x}(t)$ on two intervals of time. $e(t) \rightarrow 0$ by $t = 135$. ($C_a, \Lambda_{4SL}, \delta = 0.1, \eta = 1, \xi = 0.1$)

verge to $\tilde{x}(t)$ on the extended time interval $t \in [0, 135]$ in Figure 6.13b and has estimation error $(1.0 \times 10^{-4}) * \begin{bmatrix} 0.0 & 0.381142 & 0.459229 & 0.091254 & 0.0 & -0.091254 \end{bmatrix}^T$ at $t = 135$.

Increasing η decreases the amount of time it takes the SLSC/FO observer to converge to $\tilde{x}(t)$. $e(t)$ reduces to $(1.0 \times 10^{-6}) * \begin{bmatrix} 0.0 & 0.765680 & -0.366451 & 0.134287 & 0.0 & -0.134287 \end{bmatrix}^T$ at $t_f = 45$ when $\eta = 100$, and Figure 6.14a visually confirms convergence by the desired final time. This larger η also improves the SLSC/FO observer's estimate of $\tilde{x}(t)$, indicated by the smooth difference in Figure 6.14a as opposed to the oscillating difference in Figure 6.13. Figure 6.14b is included to provide a closer view of the estimation error behavior on the $t \in [0, 5]$ interval. Another graph that can be used to review the observer's estimate of $\tilde{x}(t)$ plots the SLSC/FO observer and the SLSC solution together. Recall, $\hat{x}(0)$ is greater than $\tilde{x}(0)$ in each state variable, so a reasonable expectation would be for the observer to initially decrease toward the example solution. However, Figure 6.14c shows an initial increase in the observer's $e_1(t)$ component that does not mimic this state variable's true behavior during that interval of time.

The estimation error for the full-order observer constructed using the alternative stabilized completion (ASC/FO observer) when $\eta = 100$ appears in Figure 6.15a, and its associated difference at $t_f = 45$ is $(1.0 \times 10^{-6}) * [0.0 \ -0.586192 \ 0.329881 \ -0.102808 \ 0.0 \ 0.102808]^T$. A closer inspection in Figure 6.15b reveals $\tilde{x}(t) - \hat{x}(t)$ converges to zero monotonically for the ASC/FO observer. The estimation errors in the $e_1(t)$ and $i_v(t)$ components for this observer are clearly different from those for the SLSC/FO observer. Plotting the ASC/FO observer and the ASC solution together in Figure 6.15c helps explain why. For the estimate of $e_1(t)$, the ASC/FO observer does not initially increase away from the ASC solution like the SLSC/FO observer does from the SLSC solution in Figure 6.14c. For the estimate of $i_v(t)$, the ASC/FO observer does not decrease as quickly to the ASC solution, which means it takes longer for $\tilde{x}(t) - \hat{x}(t)$ to go to zero for the ASC/FO observer than for the SLSC/FO observer in this particular component.

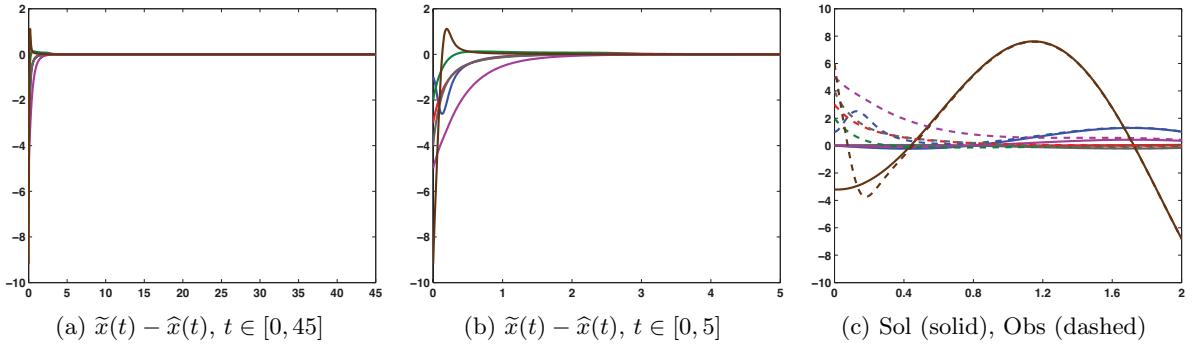


Figure 6.14: SLSC/FO observer's $\tilde{x}(t) - \hat{x}(t)$ on two intervals of time; SLSC solution and FO observer plotted together. $e(t) \rightarrow 0$ by $t_f = 45$. (C_a , Λ_{4SL} , $\delta = 0.1$, $\eta = 100$, $\xi = 0.1$)

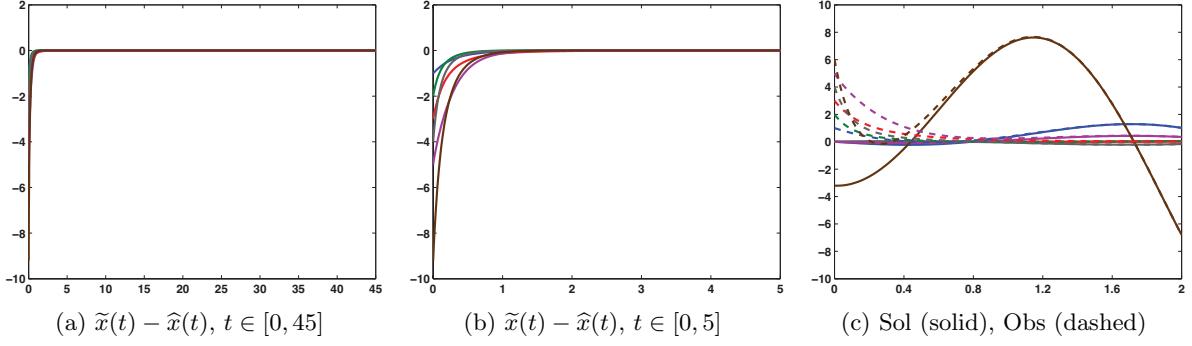


Figure 6.15: ASC/FO observer's $\tilde{x}(t) - \hat{x}(t)$ on two intervals of time; ASC solution and FO observer plotted together. $e(t) \rightarrow 0$ by $t_f = 45$. (C_a , Λ_{4A} , $\delta = 0.1$, $\eta = 100$, $\xi = 0.1$)

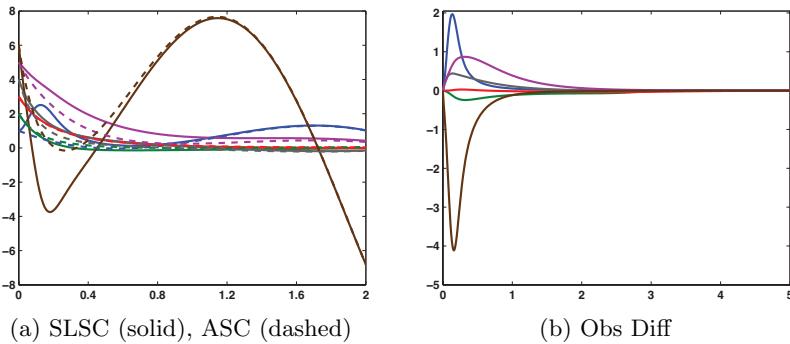


Figure 6.16: Observer comparison. SLSC/FO and ASC/FO observers plotted together; the observer difference is SLSC/FO – ASC/FO. (C_a , Λ_{4SL} , Λ_{4A} , $\delta = 0.1$, $\eta = 100$, $\xi = 0.1$)

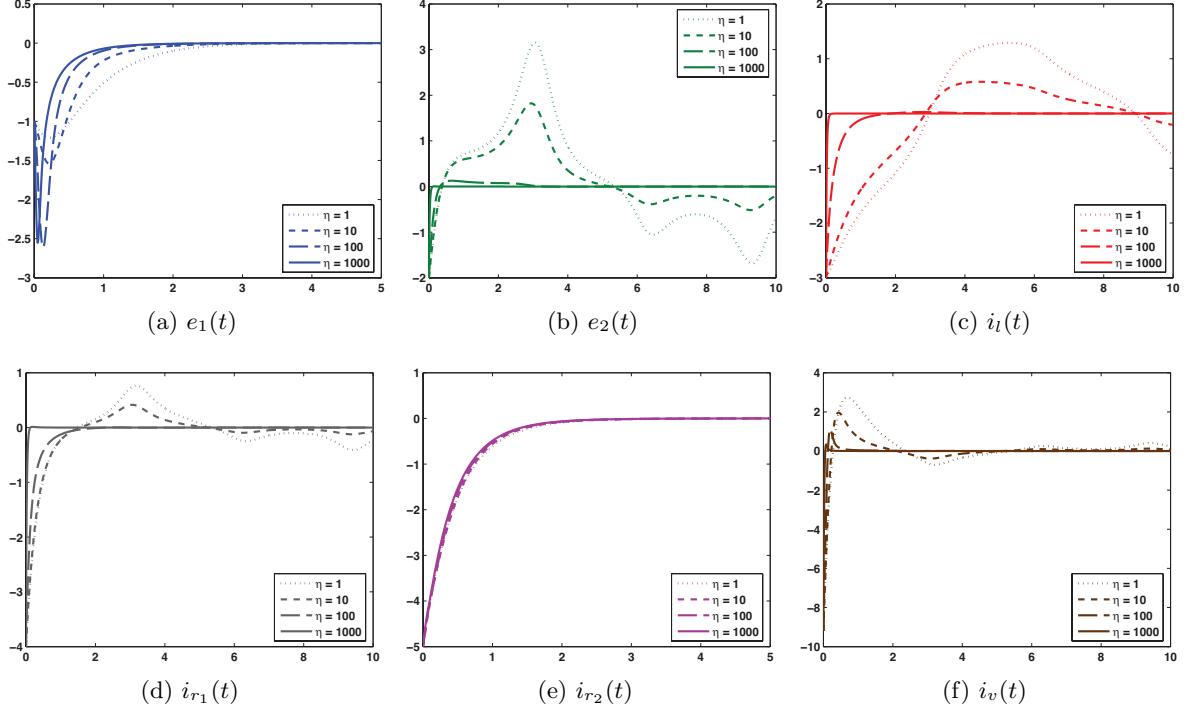


Figure 6.17: $\tilde{x}(t) - \hat{x}(t)$ progression as η is varied, SLSC/FO observer. Shortened time intervals provide closer views of initial differences. (C_a , $\Lambda_{4\text{SL}}$, $\delta = 0.1$, $\xi = 0.1$)

The SLSC/FO and the ASC/FO observers when $\eta = 100$ are compared directly in Figure 6.16. Plotting the observers together on the $t \in [0, 2]$ interval in 6.16a confirms the observations based on Figures 6.14c and 6.15c. Figure 6.16b displays the difference between these two observers (SLSC/FO – ASC/FO) in each state variable on the $t \in [0, 5]$ interval. This figure provides a measure of how much the estimates differ. The SLSC/FO and the ASC/FO observers are providing similar estimates of $\tilde{x}(t)$ by $t = 4$, with their closest estimate for all time being of component $i_l(t)$. Due to the similarity of the SLSC/FO and the ASC/FO observers when $\delta = 0.1$, $\eta = 100$, and $\xi = 0.1$, the following discussion uses the SLSC/FO observer to examine the effects from adjusting δ , η , and ξ .

As mentioned earlier, the increase in η from 1 to 100 allowed the observer to converge to $\tilde{x}(t)$ by the desired final time. Viewing the estimation error progression of an individual state variable presents another perspective on the effects from increasing η . Figure 6.17 includes the progressions as η is increased by a factor of 10 from 1 to 1000. The only state variable for which a greater η does not increase the observer's convergence rate is $i_{r_2}(t)$. Figure 6.17e shows the estimation error of this component changes very little. For state variables $e_2(t)$, $i_l(t)$, $i_{r_1}(t)$, and $i_v(t)$, a greater η also reduces the magnitude of $\tilde{x}(t) - \hat{x}(t)$. Although improved convergence rates and properties correspond with an increase in η for these four components, in Figure

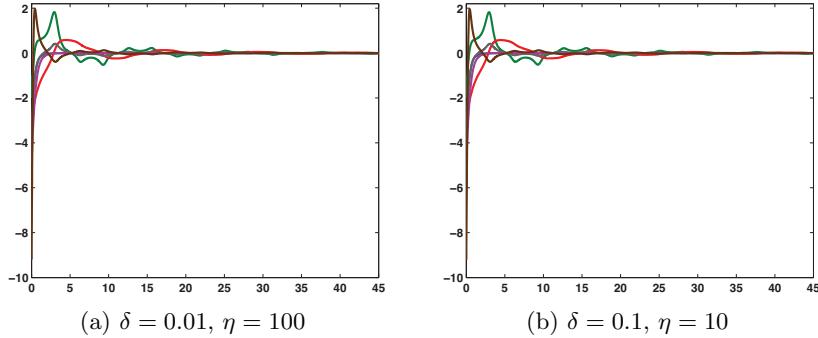


Figure 6.18: SLSC/FO observers' $\tilde{x}(t) - \hat{x}(t)$ as δ and η are varied. These estimation errors appear to be the same. (C_a , $\Lambda_{4\text{SL}}$, $\xi = 0.1$)

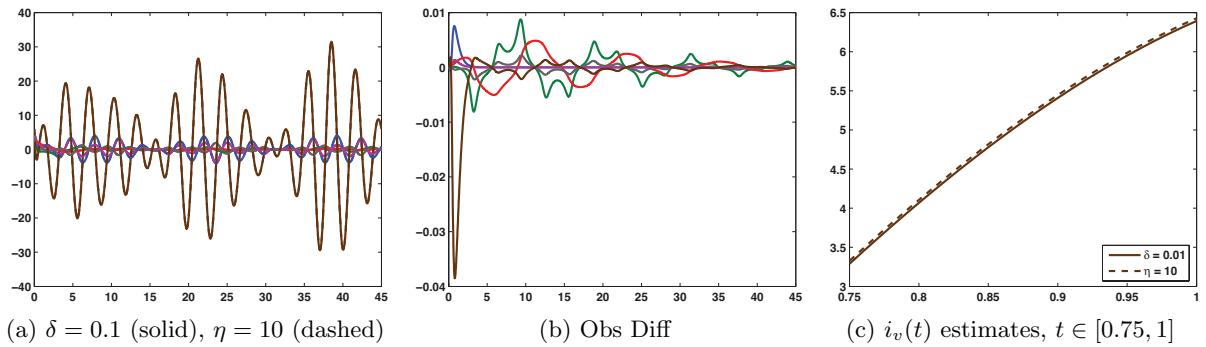


Figure 6.19: $\delta = 0.01$ SLSC/FO and $\eta = 10$ SLSC/FO observers plotted together; the SLSC/FO observer difference is $(\delta = 0.01) - (\eta = 10)$. Shortened time interval shows observers are not the same using $i_v(t)$ estimates as example. (C_a , Λ_{4SL} , $\xi = 0.1$)

6.17a the magnitude of component $e_1(t)$'s estimation error reaches a maximum when $\eta = 100$. These progression figures indicate a general conclusion on how changing η affects the observer's convergence may require a closer look at each state variable's response.

Decreasing δ to 0.01 when $\eta = 100$ and $\xi = 0.1$ also affects the convergence of the SLSC/FO observer. In Figure 6.18a, convergence by the desired final time appears to fail. The estimation error $\begin{bmatrix} 0.0 & -0.010609 & -0.006136 & -0.001861 & 0.0 & 0.001861 \end{bmatrix}^T$ at $t_f = 45$ confirms this observation. Figures 6.18a and 6.18b seem to be the same, but the estimation error on the right comes from estimating $\tilde{x}(t)$ with the SLSC/FO observer when $\delta = 0.1$, $\eta = 10$, and $\xi = 0.1$. The SLSC/FO observer constructed with gain matrix parameters $\delta = 0.01$, $\eta = 100$, and $\xi = 0.1$ will be identified by $\delta = 0.01$, while the other SLSC/FO observer will be identified by $\eta = 10$. Plotting the observers together in Figure 6.19a continues to give the impression that these SLSC/FO observers are close. However, the difference between the observers ($(\delta = 0.01) - (\eta = 10)$) in Fig-

ure 6.19b is not numerically zero. Since the largest difference between the observers is in their estimates of $i_v(t)$, this component is chosen from Figure 6.19a for a closer examination. Once a small enough interval is chosen, a dashed solution separate from the solid one can be identified in Figure 6.19c. Note these two observers have the same gain matrix at $t_0 = 0$. For the $\delta = 0.01$ SLSC/FO observer, $L(0) = \frac{1}{2}(0.01I)(C_a^T)(100I) = \frac{1}{2}C_a^T$, and for the $\eta = 10$ SLSC/FO observer, $L(0) = \frac{1}{2}(0.1I)(C_a^T)(10I) = \frac{1}{2}C_a^T$. Thus, since C_a and M are time-invariant, these observers' gain matrices remain the same if the ratio of their $S_L(t)$ matrices is constant.

Comparing Figures 6.20a and 6.14b, a decrease in ξ from 0.1 to 0.01 when $\delta = 0.1$ and $\eta = 100$ does not visibly alter the convergence of the SLSC/FO observer. Each of these SLSC/FO observers will be identified by its gain matrix parameter ξ . Figure 6.20b confirms the convergence observation by plotting the SLSC/FO observer difference $((\xi = 0.01) - (\xi = 0.1))$.

The stabilization parameter matrix can influence whether or not a particular combination of gain matrix parameters causes computational difficulties. Figure 6.21a displays the estimation error for the full-order observer constructed using the stabilized least squares completion with Λ_{2SL} , $\delta = 0.1$, $\eta = 100$, and $\xi = 0.1$. A close interval of $t \in [0, 2]$ in Figure 6.21b is required to view how this observer differs from the SLSC/FO observer with Λ_{4SL} and the same gain matrix parameters. When stabilizing with Λ_{2SL} , increasing either δ to 1 or ξ to 1 results in MATLAB ending the observer construction code prematurely at $t = 0.685944$. The step size of ode45 reaches a value smaller than allowed, which keeps the integration tolerance from being met. At this time t , one eigenvalue of the Riccati equation's $S_L(t)$ matrix has blown up to 3.960225×10^{13} if $\delta = 1$ and to 2.390417×10^{13} if $\xi = 1$. In contrast, the observer construction code for either $\delta = 1$ or $\xi = 1$ finishes running when using Λ_{4SL} . If $\delta = 1$, the largest eigenvalue of $S_L(t)$ is bounded above by 14 and is within $(0, 1)$ for most of $t \in [0, 45]$. The largest eigenvalue of $S_L(t)$ when $\xi = 1$ behaves similarly except its upper bound is 2. Figures 6.22a and 6.22b present the estimation errors when computing $L(t)$ with $\delta = 1$, $\eta = 100$, and $\xi = 0.1$ and with $\delta = 0.1$, $\eta = 100$, and $\xi = 1$, respectively. The estimation errors at $t_f = 45$ are $(1.0 \times 10^{-15}) * \begin{bmatrix} 0.0 & -0.149186 & 0.249800 & -0.027756 & 0.0 & 0.0 \end{bmatrix}^T$ in Figure 6.22a and $(1.0 \times 10^{-6}) * \begin{bmatrix} 0.0 & 0.223358 & -0.140031 & 0.039173 & 0.0 & -0.039173 \end{bmatrix}^T$ in Figure 6.22b. However, the observer construction code is not immune to ending prematurely when stabilizing with Λ_{4SL} . If $\delta = 1$, $\eta = 100$, and $\xi = 1$, the solver's integration tolerance error causes the program to end prematurely at $t = 0.260213$ with the largest eigenvalue of $S_L(t)$ equaling 4.884596×10^{13} . Although the choices of δ , η , and ξ may seem arbitrary, different combinations result in observers with a variety of convergence properties and rates, as well as observers that cannot be constructed due to computational instabilities. Some trial and error is necessary when deciding on the values of δ , η , and ξ for producing an acceptable observer.

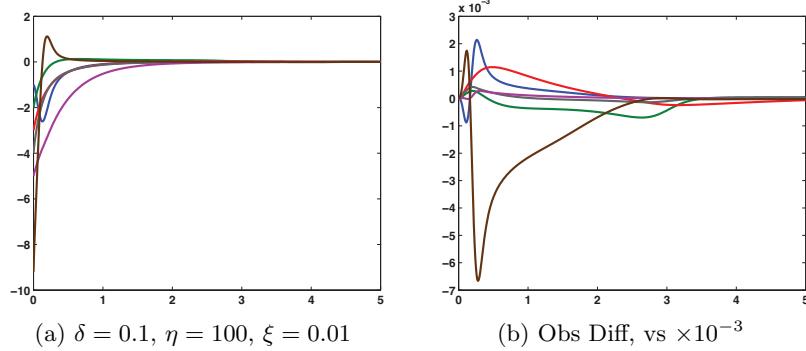


Figure 6.20: SLSC/FO observer's $\tilde{x}(t) - \hat{x}(t)$; the SLSC/FO observer difference is ($\xi = 0.01$) – ($\xi = 0.1$). Decreasing ξ does not visibly affect SLSC/FO observer convergence. (C_a, Λ_{4SL})

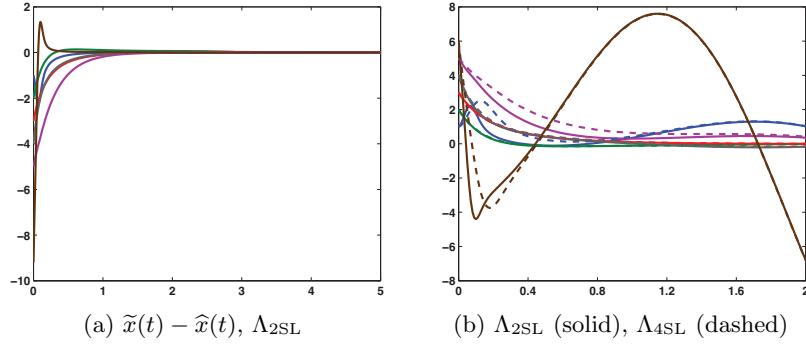


Figure 6.21: SLSC/FO observer's $\tilde{x}(t) - \hat{x}(t)$ for Λ_{2SL} ; $\Lambda_{2SL}, \Lambda_{4SL}$ SLSC/FO observers plotted together. Effects from changing Λ_{SL} are visible on $t \in [0, 2]$. ($C_a, \delta = 0.1, \eta = 100, \xi = 0.1$)

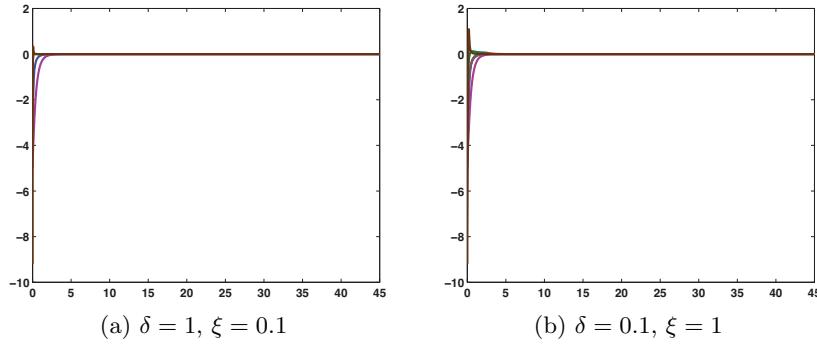


Figure 6.22: SLSC/FO observers' $\tilde{x}(t) - \hat{x}(t)$ as δ and ξ are varied. ($C_a, \Lambda_{4SL}, \eta = 100$)

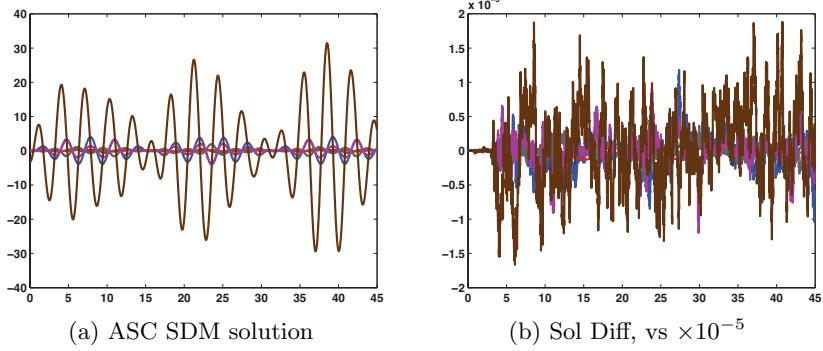


Figure 6.23: Solution comparison. The ASC solution difference is $\text{SDM} - \text{SYM}$. (Λ_{4A})

The description of the alternative stabilized completion in Section 2.3.2 mentioned using a smooth decomposition to find matrices $Z_2(t)^T$, $T_2(t)$, and $Z_{1,0}(t)^T$. In Section 3.3.2 we decided against implementing a method based on the smooth decomposition to avoid differentiating a numerically computed $\tilde{A}(t)$ at each time t when constructing the reduced-order observer, a decision that applies to the maximally reduced observer's construction as well. Since the full-order observer does not require $(\tilde{A}(t))'$, a comparison of the two alternative stabilized completions and their resulting full-order observers is being included in this case.

The algorithm executing the smooth decomposition method introduced in [109] computes $Z_2(t)^T$ and its first derivative at each time t . Based on the material in Appendix A, the process for calculating the smooth decomposition at current time t_c includes the decompositions of both $\mathcal{E}(t^*)$, where t^* is an initial time value, and $\mathcal{E}(t_c)$. If the computations at t_c for this t^* are ill-conditioned, then t^* is redefined to be t_c and the computations are repeated. However, any update to how t^* is defined is not remembered during the next iteration of MATLAB's ode45. To eliminate any possibilities of ill-conditioning, t^* is taken to be t_c , so only MATLAB's singular value decomposition of $\mathcal{E}(t_c)$ is used when calculating the smooth decomposition. As long as the first derivatives of $T_2(t)$ and $Z_{1,0}(t)^T$ are not needed, MATLAB's svd command is programmed for their computations.

The results associated with the smooth decomposition method are identified by SDM, while the results associated with the symbolic method (symbolically defined matrices) are identified by SYM. Recall, the results for this ASC comparison come from the VCL computer's version of MATLAB. The ASC SDM solution appears in Figure 6.23a. The difference between the ASC's SDM and SYM solutions ($\text{SDM} - \text{SYM}$) in Figure 6.23b is zero except for some numerical error.

The pair $(\tilde{A}(t), C_a)_{\text{SDM}}$ is observable. Its observability Gramian has rank 6 for $t \in (0, 135]$. Like the ASC/FO SYM observer, the ASC/FO SDM observer converges to $\tilde{x}(t)$ by $t_f = 45$ when $\eta = 100$: $e(45) = (1.0 \times 10^{-5}) * \begin{bmatrix} 0.0 & -0.113005 & 0.062107 & -0.019819 & 0.0 & 0.019819 \end{bmatrix}^T$. Fig-

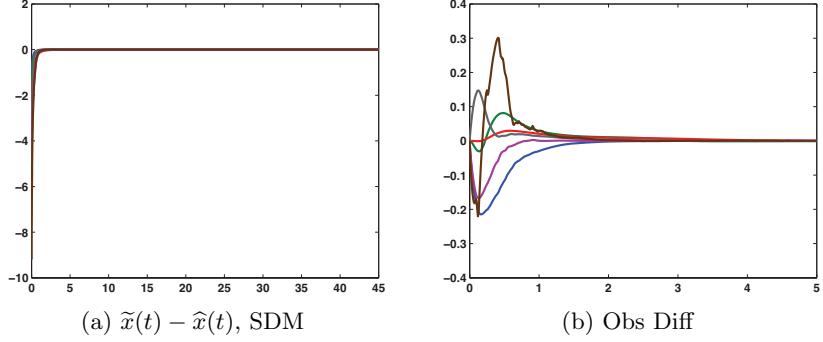


Figure 6.24: Observer comparison. ASC/FO SDM observer's $\tilde{x}(t) - \hat{x}(t)$; the ASC/FO observer difference is SDM – SYM. (C_a , Λ_{4SL} , $\delta = 0.1$, $\eta = 100$, $\xi = 0.1$)

ure 6.24a plots the estimation error for the ASC/FO SDM observer on $t \in [0, 45]$, and Figure 6.24b graphs the difference between the two full-order observers (SDM – SYM) on the shortened interval $t \in [0, 5]$ for a closer view. Although the two full-order observers initially differ in how they estimate the circuit example solution, their estimates are comparable and become equal by $t = 5$. These results help confirm the numerical smooth decomposition method is an acceptable alternative to the exact symbolic method when determining the alternative stabilized completion for use in constructing the full-order observer.

During the Λ selection analysis for Λ_L appearing earlier in Subsection 6.2.1, Criterion 4 revealed the least squares completion paired with output matrix C_a is observable. However, the ability to construct the full-order observer using this completion was put into question by Criterion 1. Matrix $\tilde{A}(t)$ is not exponentially asymptotically stable, which is affecting the computation of $S_L(t)$. Using gain matrix parameters $\delta = 0.1$, $\eta = 100$, and $\xi = 0.1$, for which both the SLSC/FO and the ASC/FO observers converged by the desired final time, the solver's integration tolerance error causes the observer construction code to end prematurely at $t = 10.359518$. The largest eigenvalue of $S_L(t)$ at this time t has blown up to 5.316134×10^{12} .

The same error did occur while attempting to construct the SLSC/FO observer with $\delta = 1$, $\eta = 100$, and $\xi = 1$, but decreasing δ to 0.1 successfully controlled the growth of $S_L(t)$. Decreasing δ for the full-order observer constructed using the least squares completion (LSC/FO observer) only delays when the solver's integration tolerance error causes the observer construction code to end. For example, Figure 6.25a plots the estimation error for the LSC/FO observer when $\delta = 0.0001$, $\eta = 100$, and $\xi = 0.1$. This combination of gain matrix parameters clearly fails to produce an LSC/FO observer that converges to $\tilde{x}(t)$ by $t_f = 45$. Even if the end time is postponed until $t = 90$, Figure 6.25b suggests the estimation error will never go to zero as time goes to infinity. The largest eigenvalues of $S_L(45)$ and $S_L(90)$ are 0.047340 and 2.782819, respectively, but when the solver's integration tolerance error terminates the code at $t = 92.380061$, the

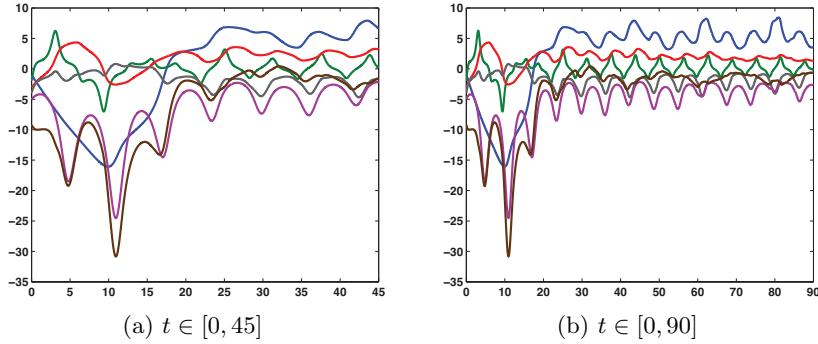


Figure 6.25: LSC/FO observer's $\tilde{x}(t) - \hat{x}(t)$ on two intervals of time. $e(t)$ fails to converge to zero by $t = 90$. ($C_a, \Lambda_L, \delta = 0.0001, \eta = 100, \xi = 0.1$)

largest eigenvalue of $S_L(t)$ has grown to 5.084107×10^{11} . Although decreasing δ postpones the solver's integration tolerance error, the effect on the gain matrix is comparable to decreasing η to 0.1 while keeping $\delta = 0.1$: $L(0) = \frac{1}{2}(0.0001I)(C_a^T)(100I) = \frac{1}{2}(0.1I)(C_a^T)(0.1I) = 0.005C_a^T$. Making η larger can be an effective means of improving the observer's convergence properties and rates, but program efficiency decreases as η increases.

Figure 6.26a plots $e(t)$ for the LSC/FO observer when $\delta = 0.1$, $\eta = 100$, and $\xi = 0.0001$. Letting $\bar{Q} = \xi I$ for a smaller value of ξ reduces the effect of the rightmost term in the Riccati equation

$$(S_L(t))' = \tilde{A}(t)S_L(t) + S_L(t)\tilde{A}(t)^T + S_L(t)\bar{Q}(t)S_L(t),$$

slowing the growth of $S_L(t)$. In addition to delaying the solver's integration tolerance error, decreasing ξ instead of δ increases $L(0)$ from $0.005C_a^T$ to $5C_a^T$ and improves the convergence properties of the LSC/FO observer. Figure 6.26 also displays the estimation errors when $\eta = 1000$ and $\eta = 10000$. The estimation errors

$$\begin{aligned}\eta = 100 : & \quad \left[\begin{matrix} 3.220219 & 0.038540 & -0.016702 & 0.008447 & 0.442855 & 0.018008 \end{matrix} \right]^T, \\ \eta = 1000 : & \quad \left[\begin{matrix} 0.335226 & 0.008157 & -0.001745 & -0.001554 & -0.051483 & 0.000320 \end{matrix} \right]^T, \\ \eta = 10000 : & \quad \left[\begin{matrix} -1.880890 & -0.000351 & 0.000251 & 0.000197 & -0.421216 & -0.000130 \end{matrix} \right]^T\end{aligned}$$

at $t_f = 45$ reveal $\tilde{x}(t) - \hat{x}(t)$ is within $(-10^{-3}, 10^{-3})$ for components $e_2(t)$, $i_l(t)$, $i_{r_1}(t)$, and $i_v(t)$ when $\eta = 10000$ but the best estimates of $e_1(45)$ and $i_{r_2}(45)$ occur when $\eta = 1000$. Unlike the differences in Figures 6.26a or 6.26c, the estimation error in Figure 6.26b appears to tend toward zero. Figure 6.27 reveals this tendency continues as the final time is extended to $t = 60$ and then to $t = 75$, but by $t = 90$, the difference is again visibly greater than zero for the LSC/FO observer when $\delta = 0.1$, $\eta = 1000$, and $\xi = 0.0001$.

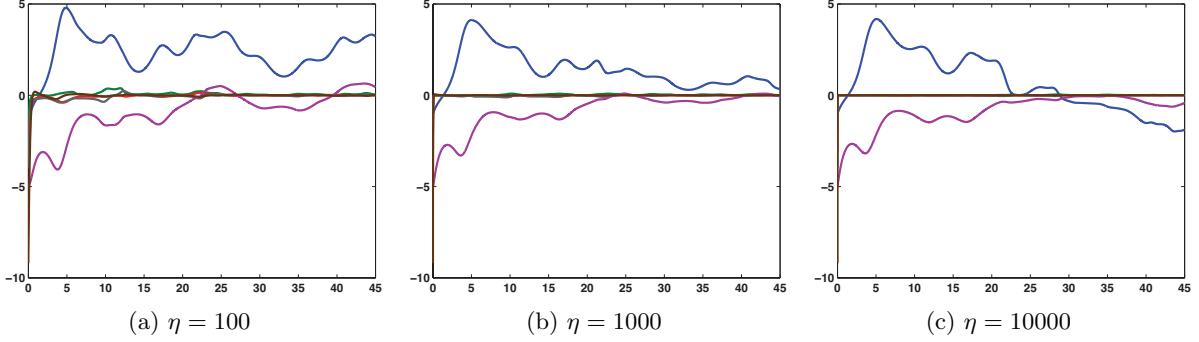


Figure 6.26: LSC/FO observers' $\tilde{x}(t) - \hat{x}(t)$ as η is varied. Increasing η does not result in $e(t) \rightarrow 0$ by $t_f = 45$. (C_a , Λ_L , $\delta = 0.1$, $\xi = 0.0001$)

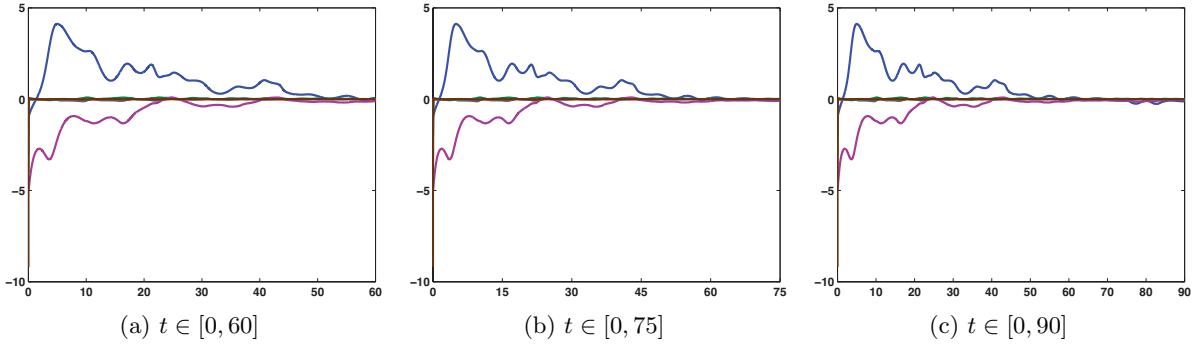


Figure 6.27: LSC/FO observer's $\tilde{x}(t) - \hat{x}(t)$ on three intervals of time. $e(t)$ fails to converge to zero by $t = 90$. (C_a , Λ_L , $\delta = 0.1$, $\eta = 1000$, $\xi = 0.0001$)

After noticing the estimation error's tendency toward zero in Figure 6.26b, the original desire was to determine if that LSC/FO observer would converge to $\tilde{x}(t)$ by $t = 135$. In order to monitor the progress of the observer construction code, the evaluation times of `ode45` were being output in MATLAB's command window, which increased the program's run time. Computing the estimation error on shorter intervals allowed for flexibility when designating MATLAB on the personal computer to run the code for this particular observer. Using MATLAB's `tic toc` command to output a program's total run time upon completion, the observer construction code took approximately 3 hours (`tic toc = 11073.332` seconds) to finish running for the interval $t \in [0, 60]$. Note for the personal computer's version of MATLAB, `tic toc` returns a time with three decimal places, which is why the time is not rounded to six decimal places. After redefining the initial conditions $\tilde{x}(0)$, $\hat{x}(0)$, and $S_L(0)$ as $\tilde{x}(60)$, $\hat{x}(60)$, and $S_L(60)$ from the saved .mat file of the previous run, the code took approximately 5 hours 30 minutes (`tic toc = 19833.509` seconds) to run from $t = 60$ to $t = 75$. Having again modified the necessary initial conditions accordingly,

the observer construction code ran for nearly 14 hours 50 minutes (tic toc = 53425.070 seconds) while completing its computations on the $t \in [75, 90]$ interval. A user-initiated Ctrl Break ended the code prematurely at $t = 91.888661$ during the $t \in [90, 105]$ interval after it had been running for almost 13 hours 30 minutes. Based on observations drawn after this case was completed, the solver's integration tolerance error would have occurred around $t = 92.380061$, the same time the code ended for gain matrix parameters $\delta = 0.0001$, $\eta = 100$, and $\xi = 0.1$. Manipulating η without altering δ and ξ does not change when the solver's integration tolerance error occurs.

Mentioned earlier, the solver's integration tolerance error is a result of the step size becoming so small that the integration tolerance cannot be met. This varying step size is dependent on the magnitude of the numbers involved in the solver's computations: as the magnitude increases, the step size decreases. A smaller step size translates into more iterations, increasing the program's run time no matter the computer. Another noted effect on the program's efficiency is the magnitude of η . Without outputting any information in MATLAB's command window except for the user-defined variables in RunLTV.m, the code takes approximately 30 minutes (tic toc = 1839.746 seconds) on the personal computer to construct the LSC/FO observer with $\delta = 0.1$, $\eta = 1000$, and $\xi = 0.0001$ on $t \in [0, 45]$. Using $\eta = 10000$ instead, the run time becomes 8 hours 20 minutes (tic toc = 30031.762 seconds), again a result of the solver's step size reacting to the computations' magnitudes.

Influenced by $\tilde{A}(t)$ not being exponentially asymptotically stable, the construction of the LSC/FO observer is not as straightforward as the construction of the two full-order observers using completions with stabilized additional dynamics. The attempted gain matrix parameter combinations failed to produce an LSC/FO observer that converged to $\tilde{x}(t)$ by the desired final time. Even if such a combination exists, the program's run time becomes a concern. When constructing a full-order observer to estimate the state of the circuit example using output matrix C_a , the LSC/FO observer is not recommended.

Output matrix C_a is not of the $\begin{bmatrix} I & 0 \end{bmatrix}$ structure, so the reduced-order observer is constructed using a transformed system. Since C_a is full row rank with rank 4, four of the six state variables of the transformed system are measurable. Thus, the reduced-order observer has order 2, meaning it is designed to estimate no more than the two unmeasurable components. The constant output matrix allows the transformation matrix $R = \begin{bmatrix} C_a \\ C_R \end{bmatrix}$ to be constant. MATLAB's

svd command returns $\begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$ for C_R , so the resulting transformation matrix is also a permutation matrix. Recall, the chosen method for defining C_R described in Section 3.3.2 utilizes the singular value decomposition of the output matrix. The pairs $(A_{R22}(t), A_{R12}(t))_{\text{SL}}$ and $(A_{R22}(t), A_{R12}(t))_{\text{A}}$ are observable since $\text{rank}(W_o(t)) = 2$, on $t \in (0, 135]$, for both.

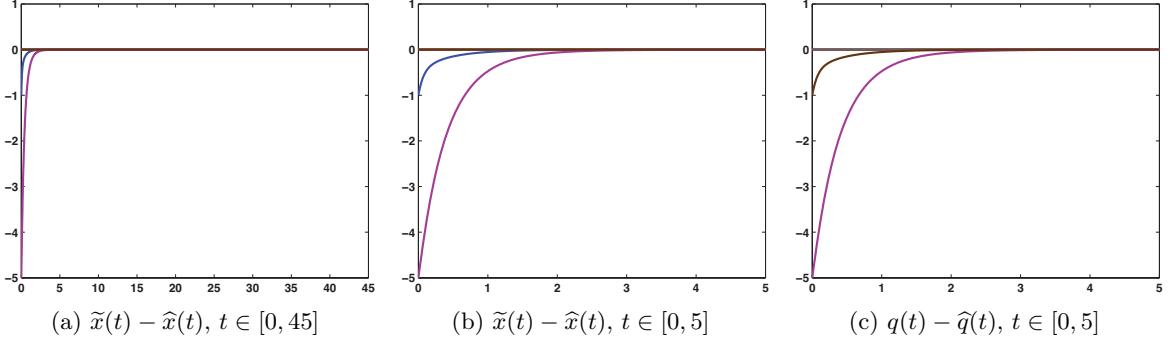


Figure 6.28: SLSC/RO observer's $\tilde{x}(t) - \hat{x}(t)$ on two intervals of time; SLSC/RO observer's $q(t) - \hat{q}(t)$. Transformation matrix R is a permutation matrix. (C_a , Λ_{4SL} , $\delta = 0.1$, $\eta = 1$, $\xi = 0.1$)

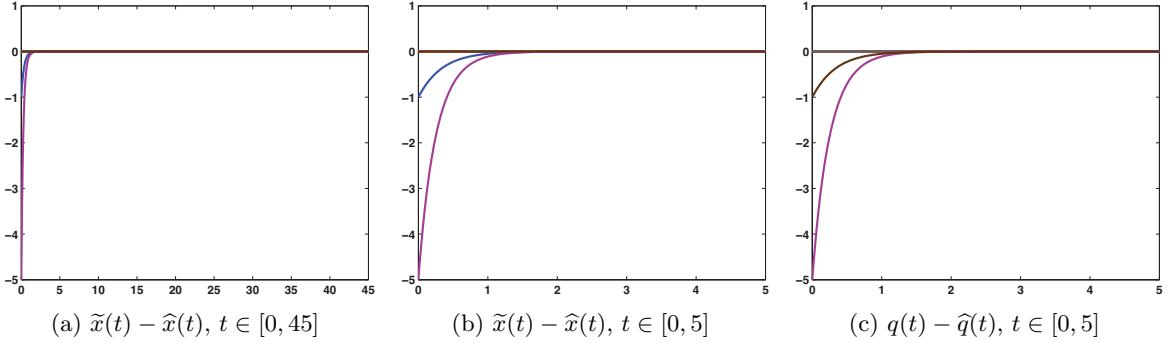


Figure 6.29: ASC/RO observer's $\tilde{x}(t) - \hat{x}(t)$ on two intervals of time; ASC/RO observer's $q(t) - \hat{q}(t)$. The order of the observer is reduced from 6 to 2. (C_a , Λ_{4A} , $\delta = 0.1$, $\eta = 1$, $\xi = 0.1$)

Gain matrix parameter η can be taken as 1 for the reduced-order observer constructed using the stabilized least squares completion (SLSC/RO observer) and for the reduced-order observer constructed using the alternative stabilized completion (ASC/RO observer). The difference $\tilde{x}(45) - \hat{x}(45)$ equals $(1.0 \times 10^{-13}) * [0.569544 \ 0.0 \ 0.0 \ 0.0 \ 0.199285 \ 0.0]^T$ for the SLSC/RO observer and $[0.0 \ 0.0 \ 0.0 \ 0.0 \ 0.0 \ 0.0]^T$ for the ASC/RO observer. Figures 6.28a and 6.29a display their respective estimation errors on the $t \in [0, 45]$ interval, and Figures 6.28b and 6.29b provide a closer view on the interval $t \in [0, 5]$. These figures indicate the SLSC/RO and the ASC/RO observers provide similar estimates of the circuit example's state.

Figures 6.28c and 6.29c are included to show how the estimation error in terms of the transformed system, $q(t) - \hat{q}(t)$, compares with $\tilde{x}(t) - \hat{x}(t)$ when the transformation matrix is simply a permutation matrix. The only two state variables for which $\tilde{y}(t) = C_a \tilde{x}(t)$ does not output any data are $e_1(t)$ and $i_{r_2}(t)$. The magenta solutions in Figures 6.28b and 6.28c and in

Figures 6.29b and 6.29c have not changed with the transformation because $i_{r_2}(t)$ in the fifth row of $\tilde{x}(t)$ is mapped to the fifth row of $q(t)$. The transformation maps $e_1(t)$ from the first row of $\tilde{x}(t)$ to the sixth row of $q(t)$, which is why the brown solutions in Figure 6.28c and 6.29c correspond with the blue estimation errors in Figures 6.28b and 6.29b.

One gain matrix parameter combination affects the construction of the SLSC/RO and the ASC/RO observers differently. Figure 6.30 presents $\tilde{x}(t) - \hat{x}(t)$ for the ASC/RO observer when $\delta = 1$, $\eta = 1$, and $\xi = 1$. This observer converges by the desired final time since $e(45) = [0.0 \ 0.0 \ 0.0 \ 0.0 \ 0.0 \ 0.0]^T$. When attempting to construct the SLSC/RO observer using these gain matrix parameters, the observer construction code ends prematurely at $t = 1.135412$ due to the solver's integration tolerance error. The largest eigenvalue of $S_L(t)$ at this time t has blown up to 8.143146×10^{12} , while the eigenvalues of $S_L(t)$ for the ASC/RO observer equal their maximum of 1 at $t_0 = 0$. This flexibility in the gain matrix parameters for the ASC/RO observer may indicate the alternative stabilized completion with Λ_{4A} does a better job of stabilizing the additional dynamics than the stabilized least squares completion with Λ_{4SL} .

For the least squares completion, reducing the order of the observer improves the stability of $\tilde{A}(t)$'s reduced-order counterpart $A_{R_{22}}(t)$. Figure 6.31a plots $\|\Phi_R(t, 0)\|$, where $\Phi_R(t, 0)$ is the solution to $\dot{\Phi}_R(t, 0) = A_{R_{22}}(t)\Phi_R(t, 0)$ with $\Phi_R(0, 0) = I$ or the state transition matrix of the reduced-order system. Although $A_{R_{22}}(t)$ is not exponentially asymptotically stable, the norm of $\Phi_R(t, 0)$ is bounded, which subdues the growth of $S_L(t)$ encountered during the construction of the LSC/FO observers. The largest eigenvalue of $S_L(t)$ is periodic, and each local maximum is bounded above by 0.92.

The observability Gramian for pair $(A_{R_{22}}(t), A_{R_{12}}(t))_L$ has rank 2 for $t > 0$. The most promising LSC/FO observer results occurred when $\xi = 0.0001$, so this gain matrix parameter is also chosen for the reduced-order observer constructed using the least squares completion (LSC/RO observer). When this ξ is combined with $\delta = 0.1$ and $\eta = 1000$, Figure 6.31b suggests the LSC/RO observer converges by the desired final time. This observation is confirmed by the estimation error at $t_f = 45$, $(1.0 \times 10^{-4}) * [-0.158245 \ 0.0 \ 0.0 \ 0.0 \ -0.026168 \ 0.0]^T$. Figure 6.31c highlights the improved convergence properties and desirable convergence rate resulting from reducing the order of the observer constructed using the least squares completion.

The process of extending the output matrix for the maximally reduced observer reveals the first two rows of C_a are linearly independent from the rows of $\mathcal{G}(t)\mathcal{F}(t)$ for all time. Thus, the extended output matrix $\begin{bmatrix} \mathcal{G}(t)\mathcal{F}(t) \\ C_x \end{bmatrix}$ for $C_x = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}$ is nonsingular. Using $y_x(t) = C_x\tilde{x}(t)$ and equation (6.13) to calculate $x(t)$, Figure 6.32 displays $\tilde{x}(t) - x(t)$ for each completion. The differences are zero except for some numerical error. The characterization of the solution manifold and the output contain enough information to determine the state of this circuit example without an observer and without a completion when output data exists.

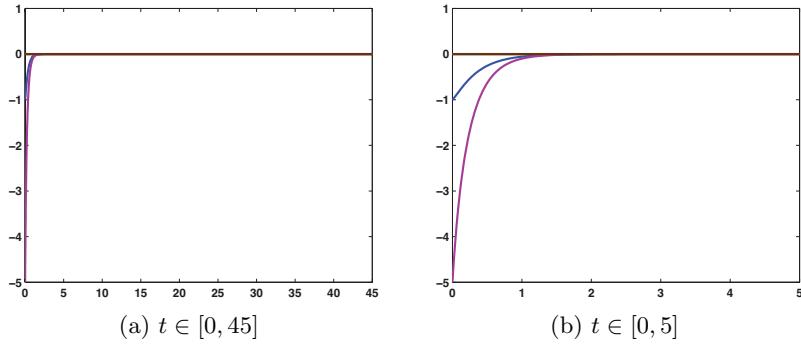


Figure 6.30: ASC/RO observer's $\tilde{x}(t) - \hat{x}(t)$ on two intervals of time. Integration tolerance error occurs for SLSC/RO observer ($\Lambda_{4\text{SL}}$) and same δ , η , and ξ . ($C_a, \Lambda_{4A}, \delta = 1, \eta = 1, \xi = 1$)

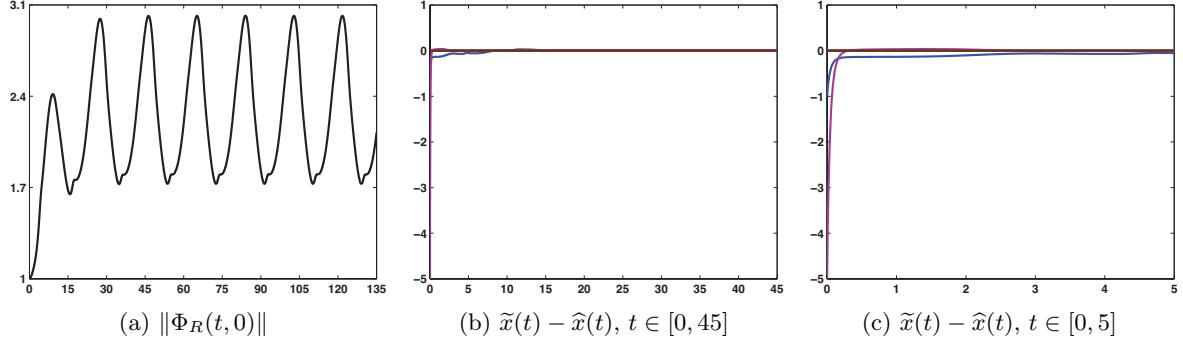


Figure 6.31: $\|\Phi_R(t, 0)\|$ is bounded; LSC/RO observer's $\tilde{x}(t) - \hat{x}(t)$ on two intervals of time. $e(t) \rightarrow 0$ by $t_f = 45$ for an LSC/RO observer. ($C_a, \Lambda_L, \delta = 0.1, \eta = 1000, \xi = 0.0001$)

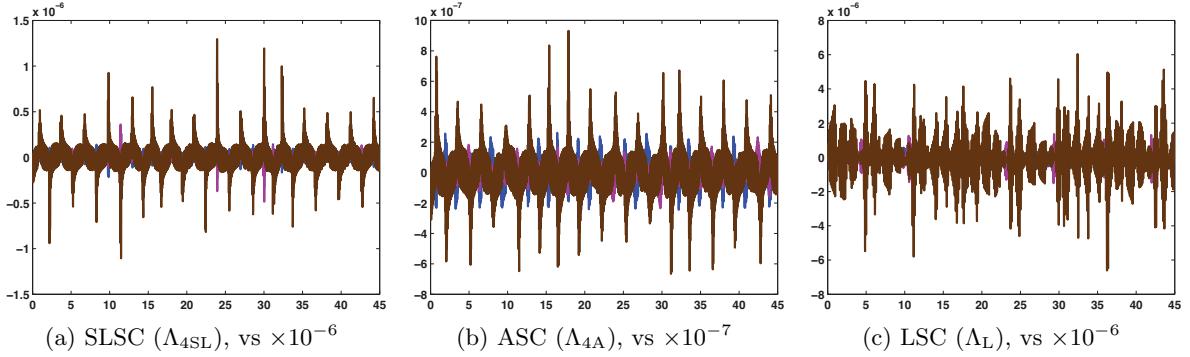


Figure 6.32: $\tilde{x}(t) - x(t)$; equation (6.13) computes $x(t)$ without observer. (C_a)

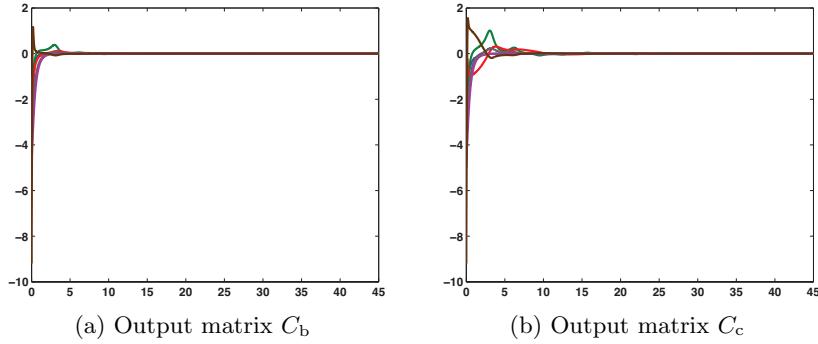


Figure 6.33: SLSC/FO observers' $\tilde{x}(t) - \hat{x}(t)$ for C_b and C_c . (Λ_{4SL} , $\delta = 0.1$, $\eta = 100$, $\xi = 0.1$)

Case 2, C_b and C_c

Independent measurements of state variables $i_l(t)$ and $i_v(t)$ are output by $\tilde{y}(t) = C_b \tilde{x}(t) = \begin{bmatrix} i_l(t) \\ i_v(t) \end{bmatrix}$, while the sum of their measurements is output by $\tilde{y}(t) = C_c \tilde{x}(t) = i_l(t) + i_v(t)$. Both pairs $(\tilde{A}(t), C_b)_{SL}$ and $(\tilde{A}(t), C_c)_{SL}$ are observable. The observability Gramian of the stabilized least squares completion with output matrix C_b has rank 6 after equaling 4 from 0.01 to 0.03 and 5 at $t = 0.04$ and $t = 0.05$. From the rank distribution of $W_o(t)$ for pair $(\tilde{A}(t), C_c)_{SL}$ in Table 6.3, the rank of the observability Gramian does not reach 6 until $t = 0.54$. When $\eta = 100$, the estimation error at $t_f = 45$ for the SLSC/FO observer with output matrix C_b is $(1.0 \times 10^{-3}) * \begin{bmatrix} 0.0 & -0.232732 & 0.083220 & -0.040817 & 0.0 & 0.040817 \end{bmatrix}^T$. Figure 6.33a plots this observer's estimation error. Although it appears the estimation error for the SLSC/FO observer with output matrix C_c in Figure 6.33b converges to zero by the desired final time, the difference $\tilde{x}(45) - \hat{x}(45) = \begin{bmatrix} 0.0 & -0.001244 & 0.000428 & -0.000218 & 0.0 & 0.000218 \end{bmatrix}^T$ is not within the desired interval $(-10^{-3}, 10^{-3})$.

In Case 1, increasing η either improved or maintained the convergence rate of the SLSC/FO observer depending on the state variable being considered. For the SLSC/FO observer with output matrix C_c , $e(45) = (1.0 \times 10^{-3}) * \begin{bmatrix} 0.0 & 0.118125 & -0.138815 & 0.020717 & 0.0 & -0.020717 \end{bmatrix}^T$ if η is increased to 1000. If η is increased again to 10000, the estimation error at $t_f = 45$ becomes $(1.0 \times 10^{-3}) * \begin{bmatrix} 0.0 & 0.022973 & -0.140320 & 0.004029 & 0.0 & -0.004029 \end{bmatrix}^T$. Figures 6.33b and 6.34 imply that if η continues to be increased, the SLSC/FO observer with output matrix C_c will not converge to $\tilde{x}(t)$ by the desired final time due to component $e_2(t)$. For a better understanding of how changing η influences each state variable's convergence, Figure 6.35 plots their estimation error progressions separately. As η increases, the rate of convergence decreases for the estimates of $e_2(t)$, $i_l(t)$, $i_{r_1}(t)$, and $i_v(t)$; increases for the estimate of $e_1(t)$; and stays the same for the estimate of $i_{r_2}(t)$. Although our analysis in Subsection 3.2.2 on the construction

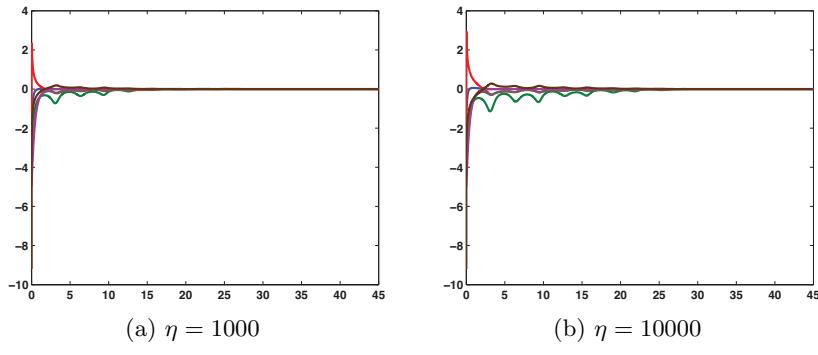


Figure 6.34: SLSC/FO observers' $\tilde{x}(t) - \hat{x}(t)$ as η is varied. For output matrix C_c , increasing η may not improve observer convergence. ($\Lambda_{4\text{SL}}$, $\delta = 0.1$, $\xi = 0.1$)

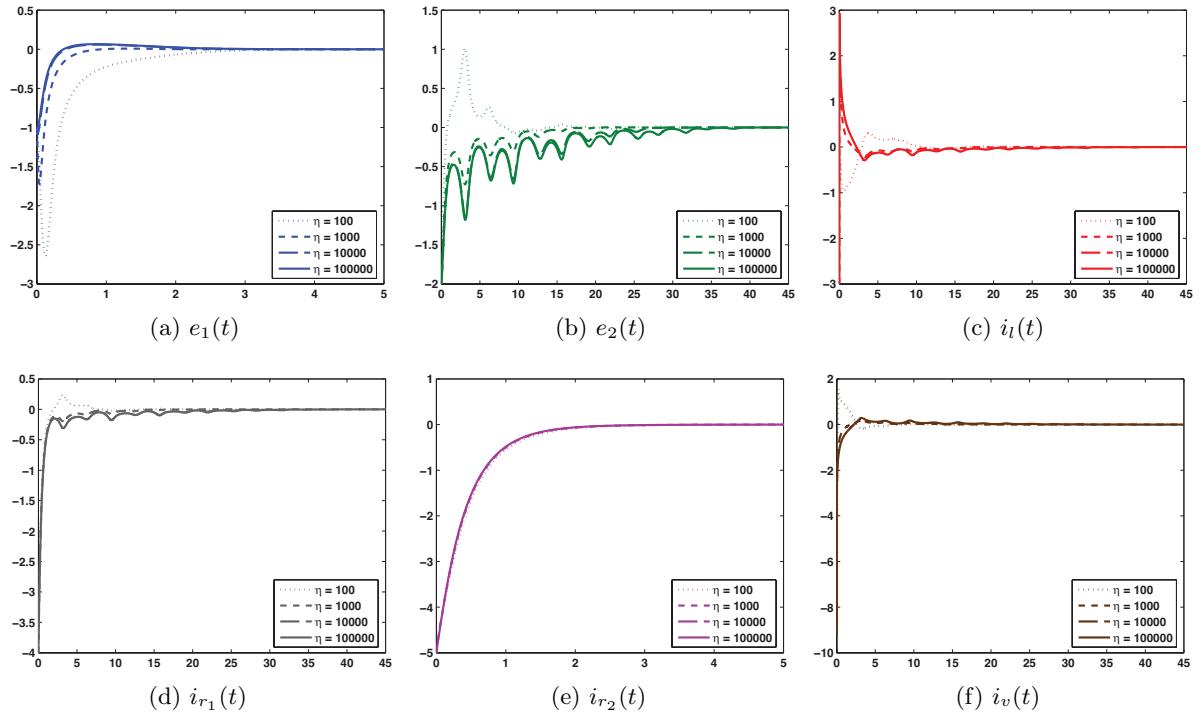


Figure 6.35: $\tilde{x}(t) - \hat{x}(t)$ progression as η is varied, SLSC/FO observer with output matrix C_c . Effects on state variable estimates are not uniform when increasing η . ($\Lambda_{4\text{SL}}$, $\delta = 0.1$, $\xi = 0.1$)

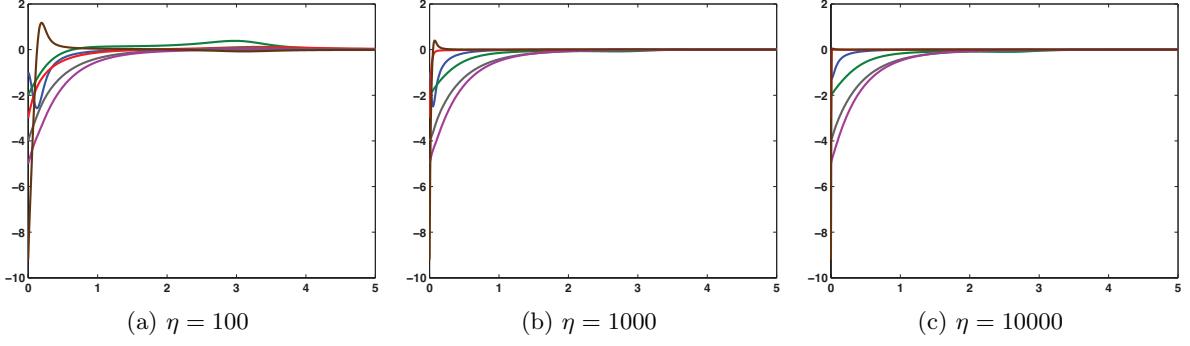


Figure 6.36: SLSC/FO observers' $\tilde{x}(t) - \hat{x}(t)$ as η is varied. For output matrix C_b , increasing η leads to $e(t) \rightarrow 0$ monotonically. ($\Lambda_{4\text{SL}}$, $\delta = 0.1$, $\xi = 0.1$)

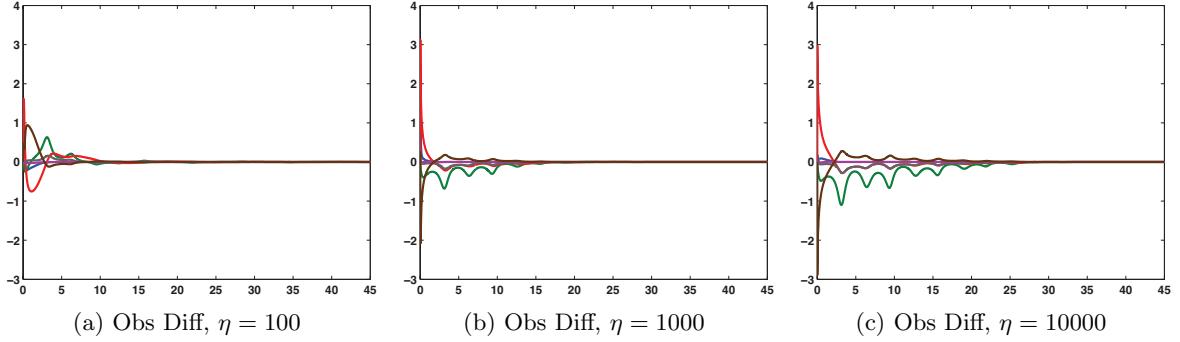


Figure 6.37: The SLSC/FO observer difference is $C_b - C_c$ as η is varied. The SLSC/FO observer with output matrix C_b converges to $\tilde{x}(t)$ faster. ($\Lambda_{4\text{SL}}$, $\delta = 0.1$, $\xi = 0.1$)

of the gain matrix produced the guideline that $M(t)$ should be chosen large, there may be an upper bound on how large $M(t)$ can be if there is a time by which the observer needs to converge. Further investigation into a possible upper bound of $M(t)$ is required.

In contrast, increasing η decreases the time it takes the SLSC/FO observer with output matrix C_b to converge to the circuit example solution. The estimation error at $t_f = 45$ decreases to $(1.0 \times 10^{-8}) * [0.0 \ -0.697970 \ 0.793215 \ -0.122412 \ 0.0 \ 0.122412]^T$ when $\eta = 1000$ and to $(1.0 \times 10^{-15}) * [-0.111022 \ 0.204697 \ -0.555112 \ 0.0 \ -0.055511 \ 0.888178]^T$ when $\eta = 10000$. Figure 6.36 presents the estimation errors on the shortened time interval $t \in [0, 5]$ for a closer view of how the state estimates improve when considering $\eta \in \{100, 1000, 10000\}$. The direct comparison of the SLSC/FO observers through their differences ($C_b - C_c$) in Figure 6.37 confirms the SLSC/FO observer with output matrix C_b provides a better estimate of the state as η increases. Figures 6.37b and 6.37c look like 6.34a and 6.34b, respectively, because the SLSC/FO observer with output matrix C_b converges to $\tilde{x}(t)$ faster.

Output matrix C_b provides information on two state variables. In particular, due to the structure of C_b , the reduced-order observer is estimating $e_1(t)$, $e_2(t)$, $i_{r_1}(t)$, and $i_{r_2}(t)$ since state variables $i_l(t)$ and $i_v(t)$ are measurable. Transformation matrix R is made nonsingular by

$$\text{defining } C_R = \begin{bmatrix} -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 \end{bmatrix}. \text{ During trial runs using the VCL computer's version}$$

of MATLAB, the returned C_R had the same structure but all ones were positive. Its resulting transformation matrix produces a different transformed system, but in terms of the original system's state variables, the observer results are the same. Output matrix C_c is 1×6 , so only one component of the transformed state vector is measurable. The transformation matrix

$$R = \begin{bmatrix} 0.000000 & 0.0 & 1.0 & 0.0 & 0.0 & 1.0 \\ 0.000000 & 1.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ -0.707107 & 0.0 & 0.5 & 0.0 & 0.0 & -0.5 \\ 0.000000 & 0.0 & 0.0 & 1.0 & 0.0 & 0.0 \\ 0.000000 & 0.0 & 0.0 & 0.0 & 1.0 & 0.0 \\ -0.707107 & 0.0 & -0.5 & 0.0 & 0.0 & 0.5 \end{bmatrix}$$

calculated by the personal computer's version of MATLAB for output matrix C_c transforms the output equation from equaling the linear combination $i_l(t) + i_v(t)$ to equaling the single measurable component $q_m(t)$ of the transformed system.

The observability Gramian for the stabilized least squares completion with output matrix C_b has rank 4 for the entire $t \in (0, 135]$ interval, so its pair $(A_{R22}(t), A_{R12}(t))_{SL}$ is observable. The reduced-order system for the stabilized least squares completion with output matrix C_c is also observable since $\text{rank}(W_o(t))$ is 5 after equaling 2 at $t = 0.01$, 3 from 0.02 to 0.09, and 4 from 0.10 to 0.23.

Both reduced-order observers converge to $\tilde{x}(t)$ by $t_f = 45$ when $\eta = 1$. The estimation error $(1.0 \times 10^{-3}) * \begin{bmatrix} -0.066678 & -0.164853 & 0.0 & -0.017218 & -0.023388 & 0.0 \end{bmatrix}^T$ for the SLSC/RO observer with output matrix C_b is just within $(-10^{-3}, 10^{-3})$. The difference $\tilde{x}(45) - \hat{x}(45)$ equals $(1.0 \times 10^{-6}) * \begin{bmatrix} 0.014239 & -0.102300 & -0.027909 & -0.020439 & 0.004995 & 0.027909 \end{bmatrix}^T$ for the SLSC/RO observer with output matrix C_c . Figure 6.38 displays the estimation error for each reduced-order observer in terms of its transformed system. For the SLSC/RO observer with output matrix C_b , the difference $q(t) - \hat{q}(t)$ in the first two state variables is 0.0, while the estimation error in the first state variable is 0.0 for the SLSC/RO observer with output matrix C_c . The closer views in Figures 6.39a and 6.39b confirm the reduced-order observers are estimating just the unmeasurable components of their respective transformed systems (a circular marker

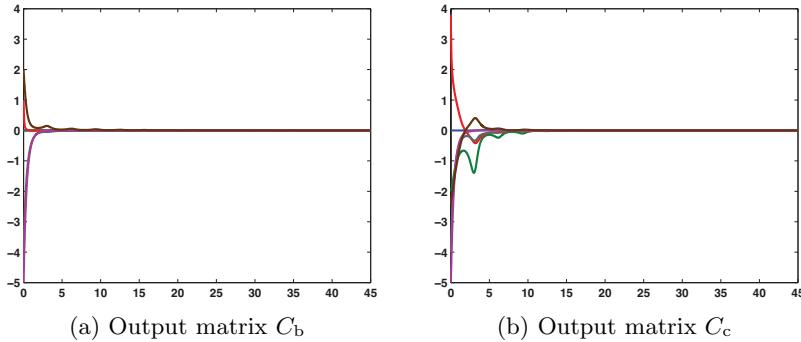


Figure 6.38: SLSC/RO observers' $q(t) - \tilde{q}(t)$ for C_b and C_c . ($\Lambda_{4\text{SL}}$, $\delta = 0.1$, $\eta = 1$, $\xi = 0.1$)

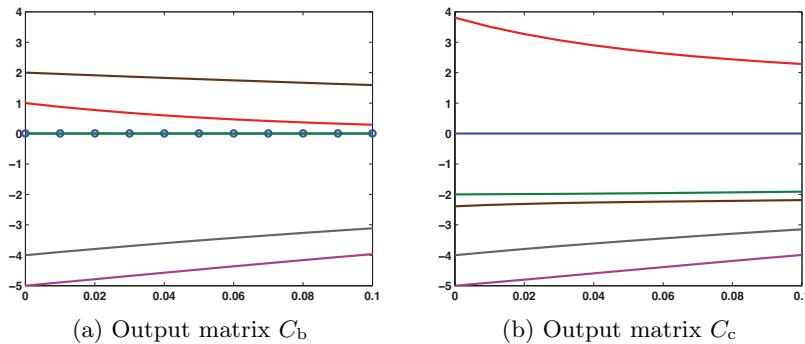


Figure 6.39: SLSC/RO observers' $q(t) - \hat{q}(t)$ for C_b and C_c . Shortened time intervals for results from Figure 6.38 show each reduced-order observer is estimating just the unmeasurable components of its transformed state vector. ($\Lambda_{4\text{SL}}, \delta = 0.1, \eta = 1, \xi = 0.1$)

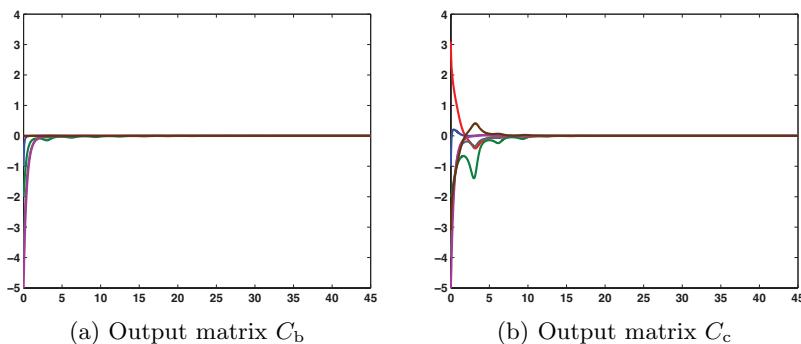


Figure 6.40: SLSC/RO observers' $\tilde{x}(t) - \hat{x}(t)$ for C_b and C_c . ($\Lambda_{4\text{SL}}$, $\delta = 0.1$, $\eta = 1$, $\xi = 0.1$)

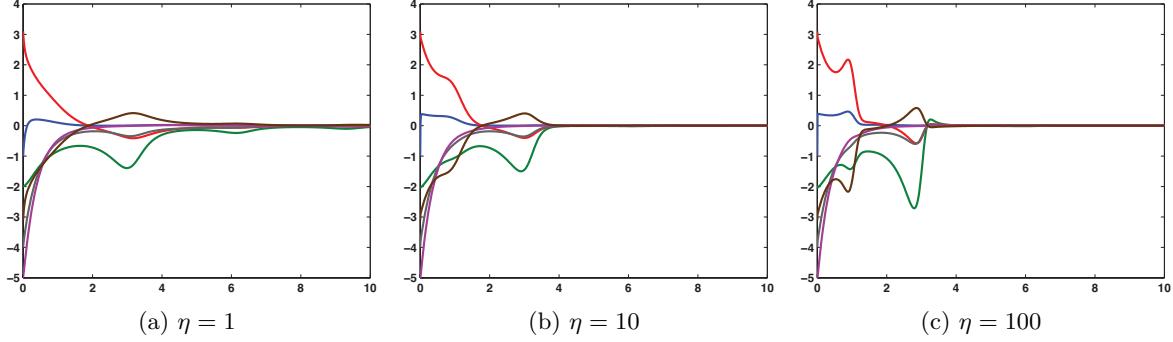


Figure 6.41: SLSC/RO observers' $\tilde{x}(t) - \hat{x}(t)$ as η is varied. For output matrix C_c , increasing η exchanges an estimate's accuracy for faster convergence. (Λ_{4SL} , $\delta = 0.1$, $\xi = 0.1$)

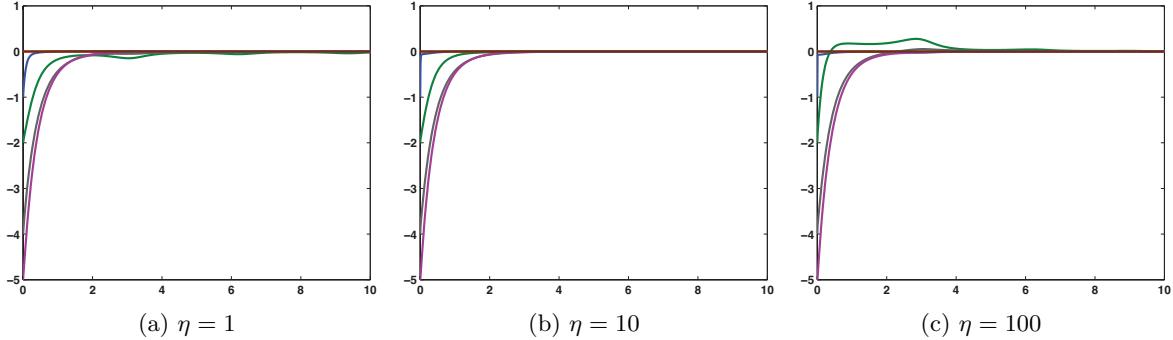


Figure 6.42: SLSC/RO observers' $\tilde{x}(t) - \hat{x}(t)$ as η is varied. For output matrix C_b , a maximum η may exist for convergence to occur by a specific time. (Λ_{4SL} , $\delta = 0.1$, $\eta = 0.1$)

identifies the blue solution underneath the green one in Figure 6.39a). The estimation errors for these reduced-order observers in terms of $\tilde{x}(t)$ and $\hat{x}(t)$ appear in Figure 6.40.

When considering output matrix C_c , increasing η does not affect the SLSC/RO observer the same way it did the SLSC/FO observer. Figure 6.41 shows increasing η for the SLSC/RO observer with output matrix C_c decreases the amount of time it takes the estimation error to go to zero. However, the magnitude of the difference before convergence increases. Even though the observer is taking less time to converge to $\tilde{x}(t)$, its initial estimate becomes less accurate.

As η is increased to 10 and then to 100, the estimation error at $t_f = 45$ improves to $(1.0 \times 10^{-5}) * \begin{bmatrix} 0.071566 & 0.176938 & 0.0 & 0.018481 & 0.025103 & 0.0 \end{bmatrix}^T$ and then to $(1.0 \times 10^{-6}) * \begin{bmatrix} 0.086574 & 0.214163 & 0.0 & 0.022386 & 0.030371 & 0.0 \end{bmatrix}^T$ for the SLSC/RO observer with output matrix C_b . The estimation errors in Figure 6.42 show a progression in state variable $e_2(t)$ that implies its best estimate out of $\eta \in \{1, 10, 100\}$ occurs when $\eta = 10$. Similar to the results for

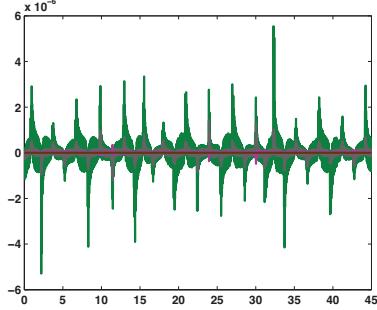


Figure 6.43: $\tilde{x}(t) - x(t)$, SLSC ($\Lambda_{4\text{SL}}$) and $C_x = C_b$, vs $\times 10^{-6}$; equation (6.13) computes $x(t)$ without observer.

the SLSC/FO observer with output matrix C_c , a large η or unbounded M may also keep this SLSC/RO observer from converging to $\tilde{x}(t)$ by the desired final time. Case 2 helps illustrate the importance of studying both numerical and visual estimation error results when drawing conclusions about how changing η affects the observers.

Both rows of C_b are linearly independent from the rows of (6.11) for all time. The output matrix for the maximally reduced observer can be extended by letting $C_x = C_b$, so the extended output matrix is invertible. Again the state of this circuit example can be determined without an observer since enough information is provided about $x(t)$ through the output equation and the constraints characterizing the solution manifold. The difference $\tilde{x}(t) - x(t)$ comparing the solutions of the stabilized least squares completion and equation (6.13) appears in Figure 6.43.

Output matrix C_c and the rows of (6.11) are also linearly independent for all time. With $C_x = C_c$, the extended output matrix is 5×6 . Thus, the known information about five state variables reduces the order of the maximally reduced observer to 1. While attempting to construct the transformation matrix $\bar{R}(t)$, MATLAB's svd command returns $C_{\bar{R}}(t) = [0 \ 4 + 2 \sin(\text{conj}(t)) \ 1 \ 1 \ 0 \ -1]$. After using its diff command to determine $(C_{\bar{R}}(t))'$ for $(\bar{R}(t))'$, MATLAB does not know how to evaluate $\text{diff}(\text{conj}(t), t)$ at each time t . Instead, $C_{\bar{R}}(t) = [0 \ 4 + 2 \sin(t) \ 1 \ 1 \ 0 \ -1]$ is hard coded for this specific example since all of the time values are real numbers.

The rank of the observability Gramian for pair $(A_{\bar{R}_{22}}(t), A_{\bar{R}_{12}}(t))_{\text{SL}}$ is 1 for $t \in (0, 135]$. The maximally reduced observer constructed using the stabilized least squares completion (SLSC/MR observer) converges by the desired final time when $\eta = 1$: $e(45) = (1.0 \times 10^{-3}) * [0.000019 \ -0.119688 \ -0.020872 \ -0.020995 \ 0.000007 \ 0.020872]^T$. The only nonzero solution from $q(t) - \hat{q}(t)$ in Figure 6.44a is in the sixth component of the transformed system's state vector ($q_6(t)$) since the first five are known and do not need to be estimated by the maximally reduced observer.

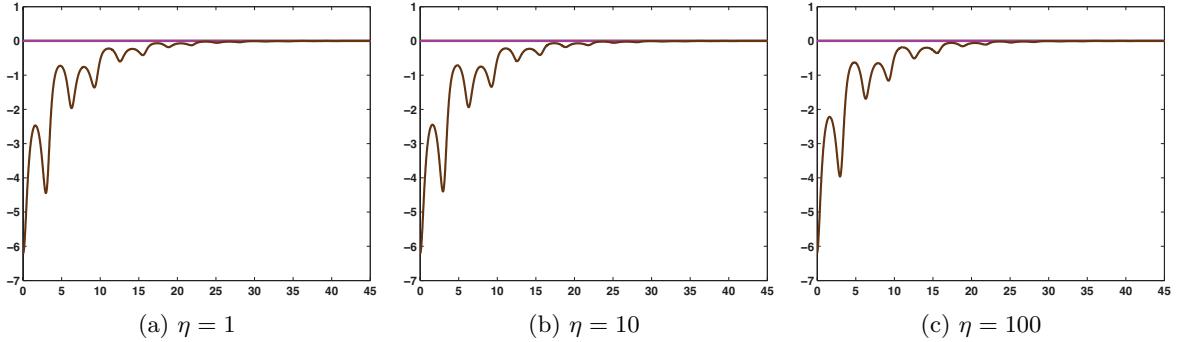


Figure 6.44: SLSC/MR observers' $q(t) - \hat{q}(t)$ as η is varied. For output matrix C_c , the SLSC/MR observer estimates just one component of $q(t)$. ($\Lambda_{4\text{SL}}$, $\delta = 0.1$, $\xi = 0.1$)

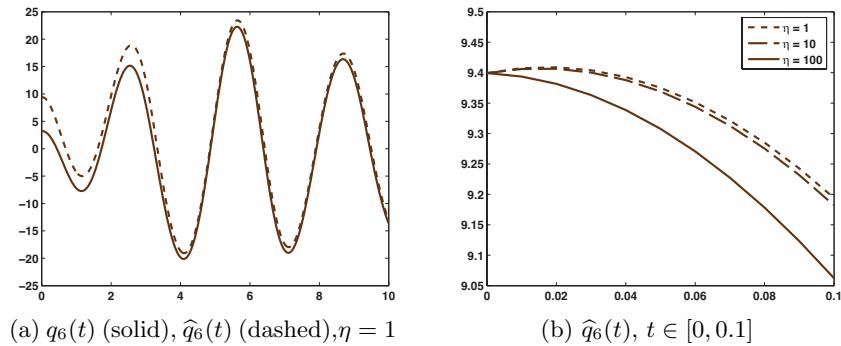


Figure 6.45: $q_6(t)$ plotted with $\hat{q}_6(t)$ estimated by the SLSC/MR observer when $\eta = 1$; shortened time interval shows how much varying η affects $\hat{q}_6(t)$. ($C_c, \delta = 0.1, \xi = 0.1$)

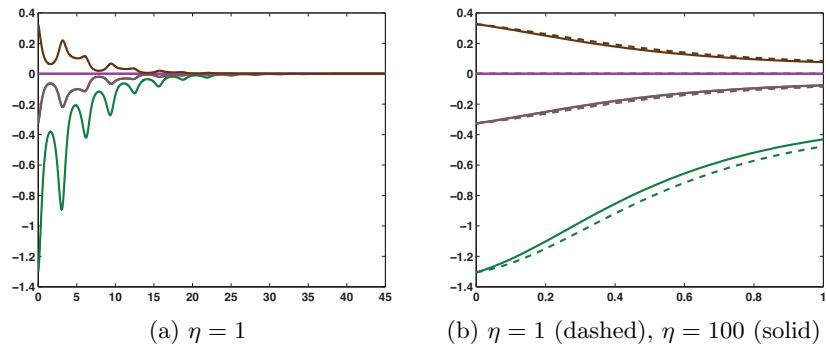


Figure 6.46: SLSC/MR observers' $\tilde{x}(t) - \hat{x}(t)$ as η is varied. Without plotting the estimation errors together on a shortened interval, increasing η would not appear to affect convergence. ($C_c, \Lambda_{4\text{SL}}, \delta = 0.1, \xi = 0.1$)

The visually comparable estimation errors in Figures 6.44a, 6.44b, and 6.44c indicate increasing η from 1 to 10 and then to 100 does not have a significant effect on the maximally reduced observer's convergence rate. Each of the SLSC/MR observers in Figure 6.44 overestimates the sixth component. Viewing $\hat{q}_6(t)$ in relation to $q_6(t)$ in Figure 6.45a provides a reference for comparing the SLSC/MR observer estimates in Figure 6.45b, which improve slightly as η is increased. Figure 6.46a displays the estimation error in terms of $\tilde{x}(t)$ and $\hat{x}(t)$ for the SLSC/MR observer when $\eta = 1$. At first glance, the oscillations may make the estimate look unsatisfactory, but the vertical scale reveals this order 1 observer provides a close estimate of $\tilde{x}(t)$. Plotting the estimation errors together for the SLSC/MR observers when $\eta = 1$ (dashed) and $\eta = 100$ (solid) in Figure 6.46b confirms in terms of the original state vector that the estimate improves as η is made larger.

Overall, the convergence properties and rates for the SLSC/FO and SLSC/RO observers are better with output matrix C_b than with output matrix C_c . Also, when $\tilde{y}(t) = C_b\tilde{x}(t)$ is considered together with the equation characterizing the solution manifold, enough information is known about the state to make observer construction unnecessary. The comparisons made in Case 2 reveal advantages of observing with an output from which data about multiple state variables is collected.

During the selection of Λ_A , stabilization parameter matrix Λ_{4A} was the only option for which the observability Gramian of pair $(\tilde{A}(t), C_c)_A$ was well-conditioned. However, the rank of $W_o(t)$ for pair $(\tilde{A}(t), C_b)_A$ is 4 from 0.01 to 0.02, increases to 5 at $t = 0.03$, decreases to 4 from 0.04 to 0.05, and then increases to 6 for the rest of the interval. The rank of $W_o(t)$ for the full-order system and output matrix C_b does not fluctuate when stabilizing with Λ_{3A} and is 6 after equaling 4 at $t = 0.01$ and 5 from 0.02 to 0.07. The ASC solution difference ($\Lambda_{4A} - \Lambda_{3A}$) in Figure 6.47b shows these two alternative stabilized completions produce comparable solutions of the circuit example system. Knowing how close the Λ_{4A} and Λ_{3A} ASC solutions are helps validate the results when comparing their ASC/FO observers.

The estimation errors of both ASC/FO observers are within the desired interval by $t_f = 45$ when $\eta = 100$: $(1.0 \times 10^{-3}) * \begin{bmatrix} 0.0 & 0.160091 & -0.041469 & 0.028077 & 0.0 & -0.028077 \end{bmatrix}^T$ if stabilizing with Λ_{4A} and $(1.0 \times 10^{-4}) * \begin{bmatrix} 0.0 & 0.793825 & -0.221013 & 0.139223 & 0.0 & -0.139223 \end{bmatrix}^T$ if stabilizing with Λ_{3A} . Figures 6.48a and 6.48b display the estimation errors for these two ASC/FO observers on the $t \in [0, 10]$ interval. The greatest difference appears to be in their estimates of state variable $e_2(t)$. The similarity of the estimation errors and the observation about component $e_2(t)$ are confirmed by the ASC/FO observer difference ($\Lambda_{4A} - \Lambda_{3A}$) in Figure 6.48c. Although the observability Gramian for pair $(\tilde{A}(t), C_b)_A$ is ill-conditioned when stabilizing with Λ_{4A} , in this instance, the fluctuation in the unobservable subspace's dimension does not negatively impact the ASC/FO observer's estimate of the state.

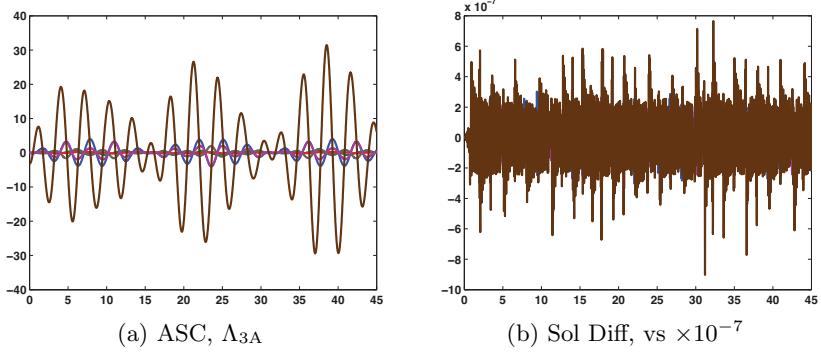


Figure 6.47: Solution comparison. The ASC solution difference is $\Lambda_{4A} - \Lambda_{3A}$.

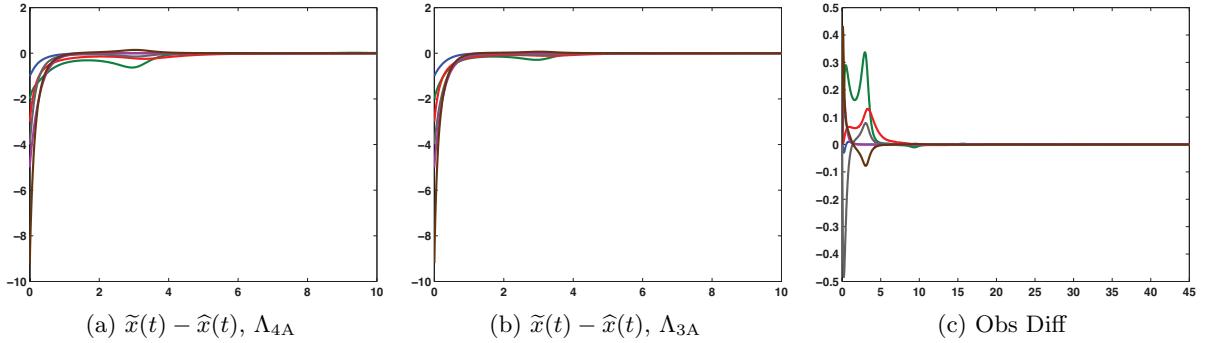


Figure 6.48: ASC/FO observers' $\tilde{x}(t) - \hat{x}(t)$ for Λ_{4A} and Λ_{3A} ; the ASC/FO observer difference is $\Lambda_{4A} - \Lambda_{3A}$. (C_b , $\delta = 0.1$, $\eta = 100$, $\xi = 0.1$)

Case 3, C_d

Neither pair $(\tilde{A}(t), C_d)_{SL}$ nor $(\tilde{A}(t), C_d)_A$ is observable. Found in Tables 6.2 and 6.5 from the Λ selection discussion on Criterion 4, the rank of the observability Gramian reaches its maximum of 5 at $t = 0.32$ for the stabilized least squares completion and is 4 on $t \in (0, 135]$ for the alternative stabilized completion.

However, it is possible to construct SLSC/FO and ASC/FO observers to estimate the state of the circuit example. The estimation errors for these observers when $\eta = 100$ are plotted separately in Figures 6.49a and 6.49b and together (SLSC/FO solid; ASC/FO dashed) in Figure 6.49c for inspection on the $t \in [0, 5]$ interval. For the SLSC/FO observer, $e(45) = (1.0 \times 10^{-4}) * [0.0 \ 0.560883 \ -0.176601 \ 0.098369 \ 0.0 \ -0.098369]^T$, and for the ASC/FO observer, $e(45) = (1.0 \times 10^{-4}) * [0.0 \ 0.390747 \ -0.062374 \ 0.068530 \ 0.0 \ -0.068530]^T$.

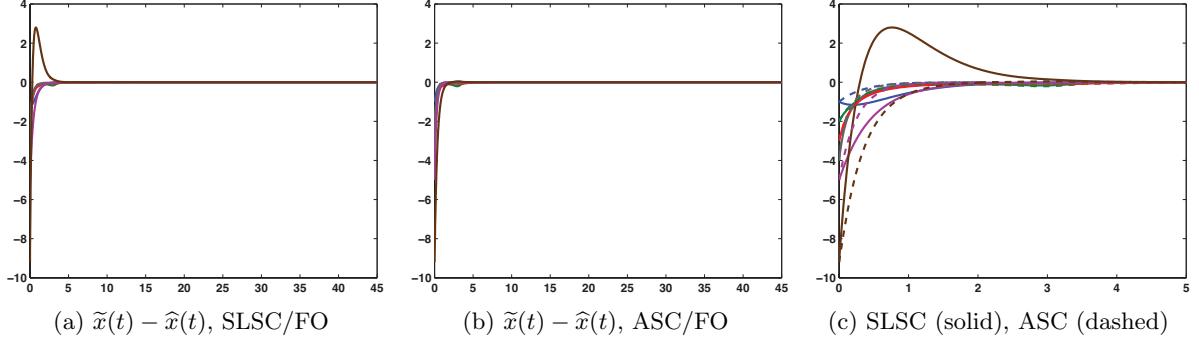


Figure 6.49: SLSC/FO and ASC/FO observers' $\tilde{x}(t) - \hat{x}(t)$ plotted separately and together. ($C_d, \Lambda_{4SL}, \Lambda_{4A}, \delta = 0.1, \eta = 100, \xi = 0.1$)

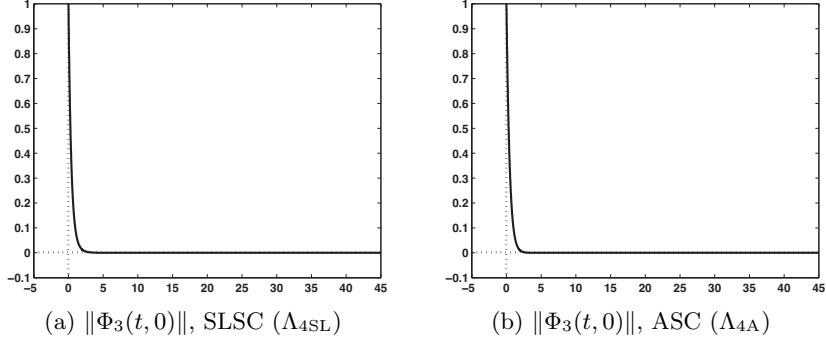


Figure 6.50: $\|\Phi_3(t, 0)\| \rightarrow 0$, so the stabilized completions are detectable. (C_d)

From Section 3.1, a linear time-varying system of ODEs is detectable if its unobservable block from the standard form for the unobservable system is uniformly asymptotically stable. Our observer construction code obtains this standard form at each time t by computing $Q^{-1}\tilde{A}Q + Q^{-1}(Q')$, where transformation matrix Q and its first derivative are defined using the processes described in [109] for finding a smooth decomposition and its first derivative. Designating $\Phi_3(t, 0)$ as the unobservable system's state transition matrix, if $\|\Phi_3(t, 0)\| \rightarrow 0$ as $t \rightarrow \infty$, then the observer can be designed to estimate the state even when the system to be observed is not observable. If an appropriate combination of gain matrix parameters had not already been selected, the convergence of $\|\Phi_3(t, 0)\|$ to zero in Figure 6.50 would reveal the stabilized completions with output matrix C_d are detectable.

The transformed systems from which Case 3's reduced-order systems come are determined using transformation matrix $R = \begin{bmatrix} -\vec{e}_3 & -\vec{e}_4 & \vec{e}_1 & \vec{e}_2 & \vec{e}_5 & \vec{e}_6 \end{bmatrix}$, where \vec{e}_i is a unit vector with the 1 in the i th row. The observability Gramian of the reduced-order system for the stabilized least squares completion has rank 2 from 0.01 to 0.03 and then rank 3 for the rest of the interval.

The pair $(A_{R22}(t), A_{R12}(t))_A$ has rank $(W_o(t)) = 2$ for $t \in (0, 135]$. In the linear time-invariant case, a reduced-order system is detectable if the full-order system is detectable, and the proof of this relationship in Section 4.2 shows the systems have the same unobservable eigenvalues. This eigenvalue proof does not carry over to the linear time-varying case, but a theoretical explanation in Section 4.3 focuses on the systems' zero dynamics.

First, a reduced-order system's minimum dimension of $N(W_o(t))$ ($\dim(N(W_o(t)))$) should equal that of its associated full-order system. For this example, $\dim(N(W_o(t)))$ eventually equals 1 for both systems of the stabilized least squares completion and $\dim(N(W_o(t))) = 2$ for both systems of the alternative stabilized completion. Second, at a specific time t , basis vectors b_f for the full-order system's $N(W_o(t))$ and vectors $b_{rx} = \begin{bmatrix} 0_{(n-m) \times 1} \\ b_r \end{bmatrix}$, an extension of the basis vectors b_r for the reduced-order system's $N(W_o(t))$, should be linearly independent. This extension is described in Section 4.3. Our algorithm in MATLAB program CompUnobs.m finds the time at which the minimum dimension of $N(W_o(t))$ is attained for both systems and selects the later time as the starting time for checking linear independence. Then at each time t , the basis vectors for the reduced-order system's $N(W_o(t))$ are extended; an augmented matrix $\begin{bmatrix} b_f & b_{rx} \end{bmatrix}$ is formed; and the singular value decomposition of this basis vector matrix is computed. Once the entries of the singular value matrix within $\pm 10^{-10}$ of zero are defined as 0 by program Rounding.m, the rank of the rounded matrix is calculated. The rounded basis vector matrix for the stabilized least squares completion's systems has rank 1 for $t \in [0.33, 135.00]$, while the rounded augmented matrix for the ASC full-order and reduced-order systems has rank 2 for $t \in (0, 135]$. With these rank results, the SLSC and ASC reduced-order systems are expected to be detectable. Checking the rank of the rounded singular value matrix rather than $\begin{bmatrix} b_f & b_{rx} \end{bmatrix}$ is important; without correcting MATLAB's internal rounding errors, a rank of 2 is returned for the stabilized least squares completion's basis vector matrix.

The estimation errors for the SLSC/RO and ASC/RO observers when $\eta = 100$ confirm the detectability of their reduced-order systems. Figures 6.51a and 6.51b visually suggest each observer converges to $\tilde{x}(t)$ by the desired final time, and Figure 6.51c is included for a comparison of the estimation errors on the shorter interval $t \in [0, 5]$. At $t_f = 45$, $\tilde{x}(t) - \hat{x}(t)$ equals $(1.0 \times 10^{-7}) * \begin{bmatrix} 0.013135 & -0.155350 & 0.0 & 0.0 & 0.003778 & -0.057906 \end{bmatrix}^T$ for the SLSC/RO observer and $(1.0 \times 10^{-7}) * \begin{bmatrix} 0.0 & 0.711236 & 0.0 & 0.0 & 0.0 & -0.399325 \end{bmatrix}^T$ for the ASC/RO observer.

By letting $C_x = C_d$, the extended output matrix is nonsingular for all time. Again, with the help of equation (6.13), state vector $x(t)$ of this circuit example is available without constructing a state estimator. Figure 6.52 exhibits the near zero difference $\tilde{x}(t) - x(t)$ for each stabilized completion.

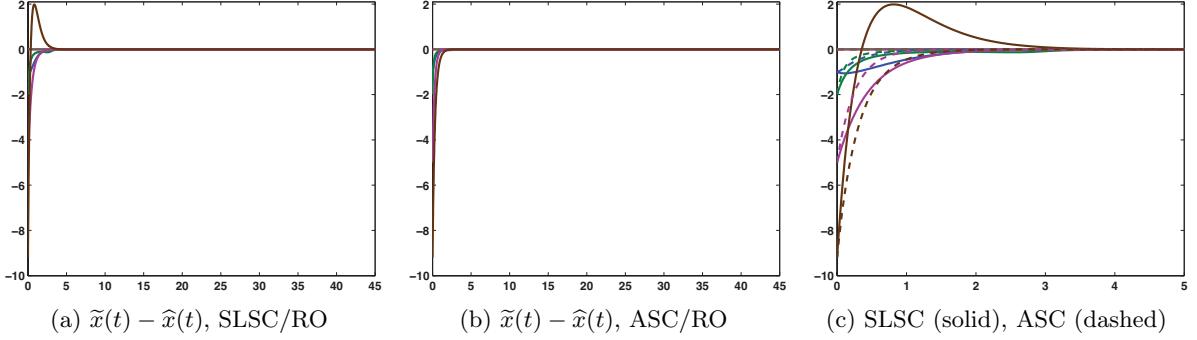


Figure 6.51: SLSC/RO and ASC/RO observers' $\tilde{x}(t) - \hat{x}(t)$ plotted separately and together. ($C_d, \Lambda_{4SL}, \Lambda_{4A}, \delta = 0.1, \eta = 100, \xi = 0.1$)

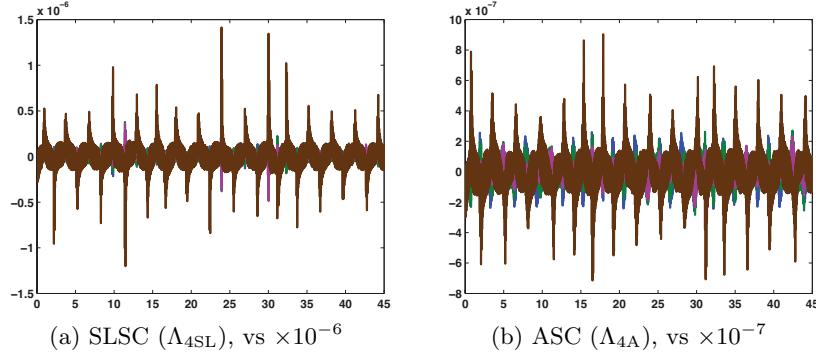


Figure 6.52: $\tilde{x}(t) - x(t)$; equation (6.13) computes $x(t)$ without observer. (C_d)

Case 4, C_e and C_f

Recall from equations (6.1f) and (6.9f), state variable $e_1(t)$ is known because it equals the negative of the input voltage source. Having an output equal this known value does not provide new information about the state but provides a way to detect if an error has occurred in the physical system. Case 4 considers output matrices C_e and C_f together to compare how the additional row in output matrix C_e changes the observers' estimates.

The stabilized least squares completion with output matrix C_e has rank $(W_o(t)) = 4$ from 0.01 to 0.04 and rank $(W_o(t)) = 5$ from 0.05 to the end of the interval. The observability Gramian for $(\tilde{A}(t), C_f)_{SL}$ eventually has rank 5 as well after having rank 2 from 0.01 to 0.04, rank 3 from 0.05 to 0.18, and rank 4 from 0.19 to 1.30. Although neither pair is observable, the estimation errors in Figures 6.53a and 6.53b indicate there exist gain matrices such that $\tilde{x}(t) - \hat{x}(t) \rightarrow 0$ as $t \rightarrow \infty$. At the desired final time, both SLSC/FO observers are within $\pm 10^{-3}$ of $\tilde{x}(t)$ when $\eta = 1000$. For C_e , the estimation error at $t_f = 45$ is

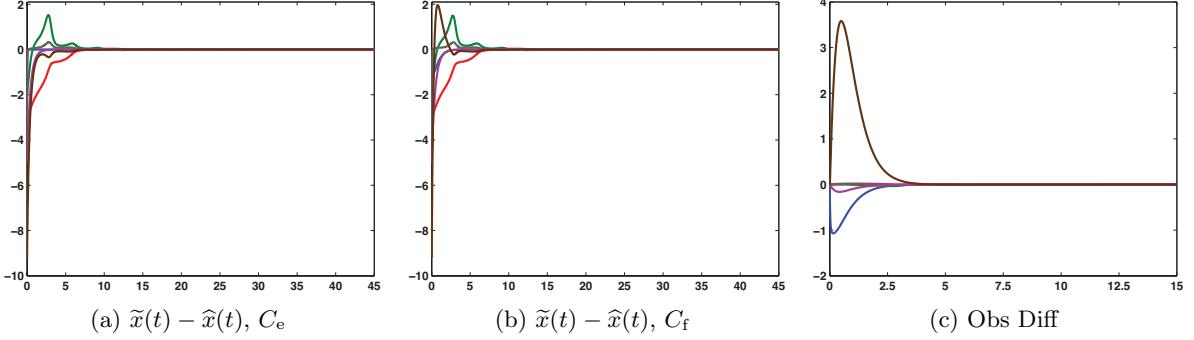


Figure 6.53: SLSC/FO observers' $\tilde{x}(t) - \hat{x}(t)$ for C_e and C_f ; the SLSC/FO observer difference is $C_e - C_f$. ($\Lambda_{4\text{SL}}$, $\delta = 0.1$, $\eta = 1000$, $\xi = 0.1$)

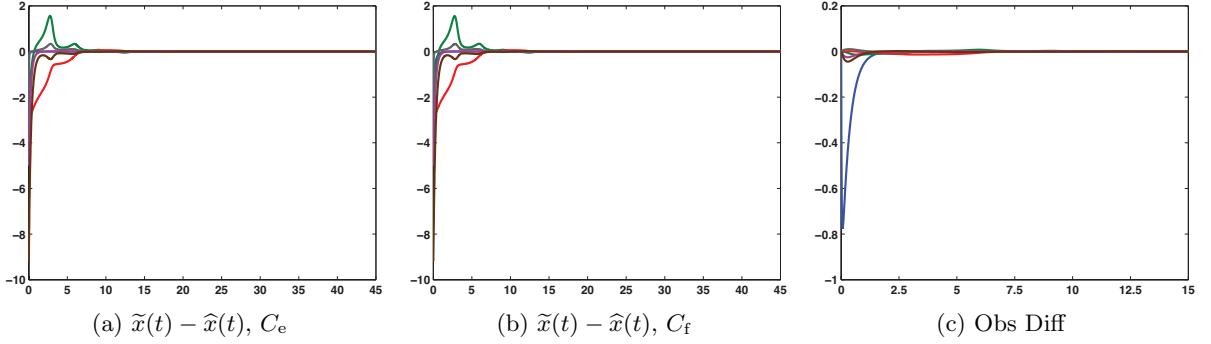


Figure 6.54: ASC/FO observers' $\tilde{x}(t) - \hat{x}(t)$ for C_e and C_f ; the ASC/FO observer difference is $C_e - C_f$. ($\Lambda_{4\text{A}}$, $\delta = 0.1$, $\eta = 1000$, $\xi = 0.1$)

$(1.0 \times 10^{-3}) * \begin{bmatrix} 0.0 & 0.424164 & 0.229966 & 0.074391 & 0.0 & -0.074391 \end{bmatrix}^T$, and for C_f , the estimation error at $t_f = 45$ is $(1.0 \times 10^{-3}) * \begin{bmatrix} 0.0 & 0.440149 & 0.232012 & 0.077195 & 0.0 & -0.077195 \end{bmatrix}^T$. A visible dissimilarity between these observers is in their estimates of state variable $i_v(t)$. This component's estimation error in Figure 6.53b exhibits polynomial behavior typically associated with the stabilized least squares completion's Jordan blocks. The structure of output matrix C_e is able to affect the Jordan blocks and their associated observability to eliminate this behavior. The initial difference between the SLSC/FO observers ($C_e - C_f$) in Figure 6.53c shows the estimates of $e_1(t)$ and $i_{r_2}(t)$ also differ but the estimates of $e_2(t)$, $i_l(t)$, and $i_{r_1}(t)$ are comparable.

The alternative stabilized completion with output matrix C_e has $\text{rank}(W_o(t)) = 4$ after having $\text{rank}(W_o(t)) = 3$ at $t = 0.01$ and $t = 0.02$. The observability Gramian for the pair $(\tilde{A}(t), C_f)_A$ has rank 2 at $t = 0.01$, rank 3 from 0.02 to 0.22, and then rank 4 for the remaining interval. Again, both pairs are detectable and the ASC/FO observers' esti-

mation errors meet the convergence criteria of being within $(-10^{-3}, 10^{-3})$ when $\eta = 1000$: $e(45) = (1.0 \times 10^{-3}) * \begin{bmatrix} 0.0 & 0.224257 & 0.496601 & 0.039331 & 0.0 & -0.039331 \end{bmatrix}^T$ for output matrix C_e and $e(45) = (1.0 \times 10^{-3}) * \begin{bmatrix} 0.0 & 0.238107 & 0.508213 & 0.041760 & 0.0 & -0.041760 \end{bmatrix}^T$ for output matrix C_f . Unlike the SLSC/FO observer estimation errors in Figures 6.53a and 6.53b, there is no visible distinction between the ASC/FO observer estimation errors in Figures 6.54a and 6.54b. The initial difference between the ASC/FO observers ($C_e - C_f$) in Figure 6.54c shows their estimates differ the most in component $e_1(t)$.

The least squares completion paired with either of Case 4's output equations also fails its observability check. The observability Gramian of pair $(\tilde{A}(t), C_e)_L$ has rank 4 at $t = 0.01$ and then 5 for the remainder of the interval. For the full-order system with output matrix C_f , $\text{rank}(W_o(t)) = 5$ after equaling 2 at $t = 0.01$, 3 from 0.02 to 0.18, and 4 from 0.19 to 20.92. Although the SLSC/FO and ASC/FO observers with output matrices C_e and C_f converge to the state of the circuit example by $t_f = 45$ when $\eta = 1000$, the observer construction code ends prematurely at $t = 10.359518$ for the LSC/FO observer with these two output matrices and the same gain matrix parameters. At this time, the largest eigenvalue of $S_L(t)$ equals 6.109183×10^{11} when $\tilde{y}(t) = C_e \tilde{x}(t)$ and 1.497614×10^{13} when $\tilde{y}(t) = C_f \tilde{x}(t)$. However, if ξ is decreased to 0.0001, the code finishes executing and the largest eigenvalues of $S_L(45)$ equal 47.340389 for both pairs $(\tilde{A}(t), C_e)_L$ and $(\tilde{A}(t), C_f)_L$.

Figure 6.55a displays the estimation error for the LSC/FO observer with output matrix C_e and gain matrix parameters $\delta = 0.1$, $\eta = 1000$, and $\xi = 0.0001$. Although this figure implies $\tilde{x}(t) - \hat{x}(t)$ will eventually converge to zero for state variables $e_1(t)$, $e_2(t)$, $i_l(t)$, and $i_{r_1}(t)$, the same cannot be said for either state variable $i_{r_2}(t)$ or $i_v(t)$. The progressions of these two components' estimation errors are shown in Figures 6.55b and 6.55c. Changing η does not appear to manipulate the observer in a way that improves the estimates of these state variables.

Similarly, the LSC/FO observer with output matrix C_f does not converge to $\tilde{x}(t)$ by the desired final time when $\delta = 0.1$, $\eta = 1000$, and $\xi = 0.0001$. In addition to the discernible estimation errors of state variables $i_{r_2}(t)$ and $i_v(t)$ in Figure 6.56a, this LSC/FO observer is also having trouble estimating state variable $e_1(t)$. Viewing the estimation error on the $t \in [0, 90]$ interval in Figure 6.56b does not conclusively show the observer will never converge to the state because the difference $\tilde{x}(t) - \hat{x}(t)$ for these three components is slowly decreasing. However, the observer construction code ends prematurely at 92.380061 due to the solver's integration tolerance error. The largest eigenvalue of $S_L(t)$ has increased from 2.782819×10^3 at $t = 90$ to 6.789507×10^{11} in under 3 units of time. The estimation error progression in Figure 6.56c demonstrates increasing η from 100 to 1000 is an incorrect response when trying to decrease the difference between the SLSC solution and the observer in component $e_1(t)$ on $t \in [0, 45]$.

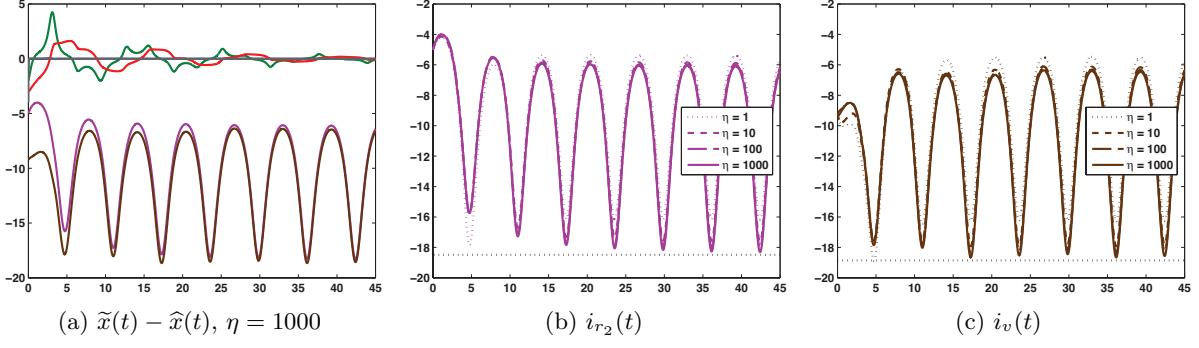


Figure 6.55: LSC/FO observer's $\tilde{x}(t) - \hat{x}(t)$ for $\eta = 1000$; $\tilde{x}(t) - \hat{x}(t)$ progression in $i_{r_2}(t)$ and $i_v(t)$ as η is varied. ($C_e, \Lambda_L, \delta = 0.1, \xi = 0.0001$)

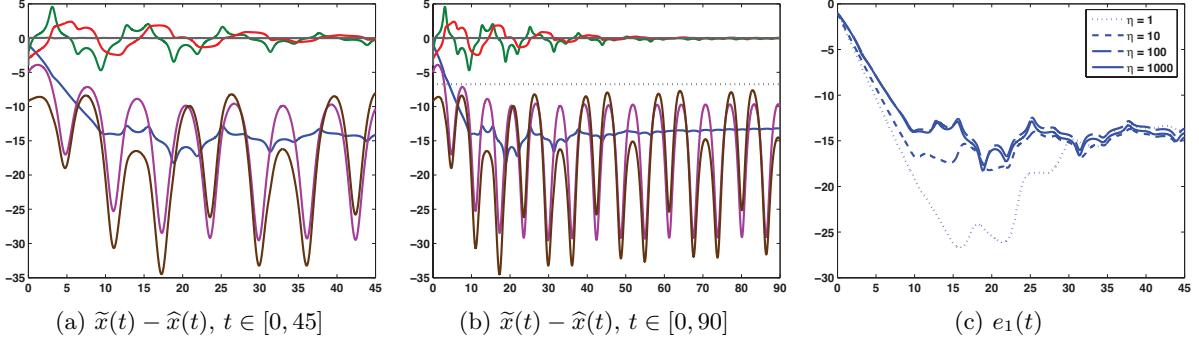


Figure 6.56: LSC/FO observer's $\tilde{x}(t) - \hat{x}(t)$ for $\eta = 1000$ on two intervals of time; $\tilde{x}(t) - \hat{x}(t)$ progression in $e_1(t)$ as η is varied. ($C_f, \Lambda_L, \delta = 0.1, \xi = 0.0001$)

Figures 6.55 and 6.56 give the impression that the least squares completion is not detectable with either output matrix C_e or C_f . To confirm this observation, the norm of state transition matrix $\Phi_3(t, 0)$ is computed for both pairs' unobservable systems. In Figure 6.57a, $\|\Phi_3(t, 0)\|$ is periodic, failing to converge to zero like the norms associated with the stabilized completions in Figures 6.57b and 6.57c. Figures 6.58b and 6.58c also support the detectable conclusion for the stabilized least squares completion and the alternative stabilized completion with output matrix C_f . Unexpectedly, $\|\Phi_3(t, 0)\|$ has a longer period for the least squares completion with output matrix C_f , so the norm's calculation is extended to $t = 135$ in Figure 6.58a. The amplitude changes but not consistently. Since the norm of this state transition matrix for the unobservable system does not converge to zero by $t = 135$, the least squares completion with output matrix C_f is categorized as unobservable on the interval under consideration. Therefore, neither the LSC/FO observer with output matrix C_e nor output matrix C_f can be constructed to estimate the state of the circuit example.

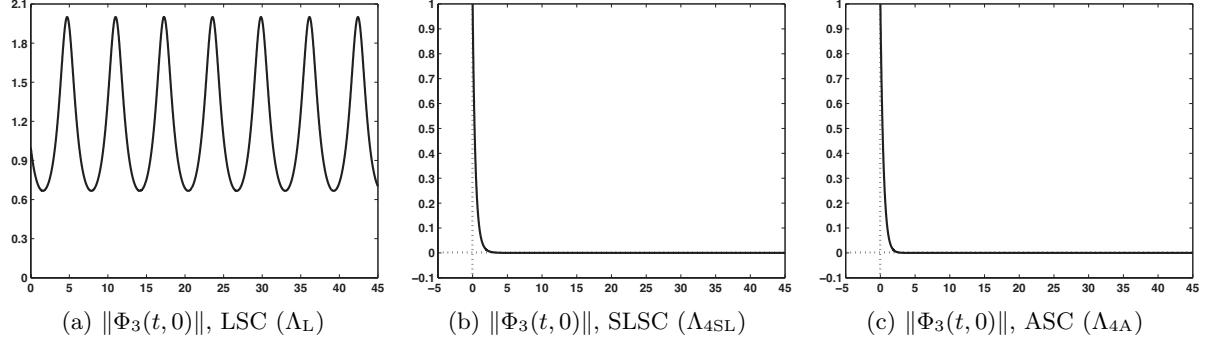


Figure 6.57: $\|\Phi_3(t, 0)\| \rightarrow 0$ for the stabilized completions but not for the LSC. For output matrix C_e , the LSC is not detectable.

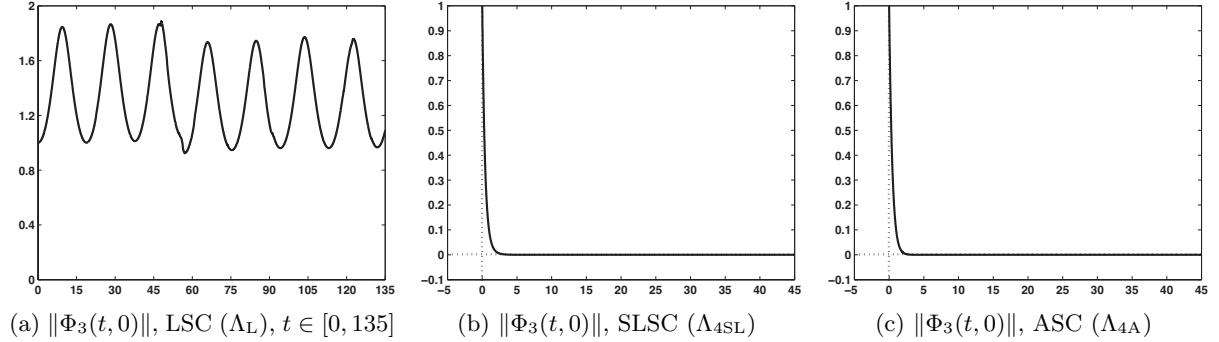


Figure 6.58: $\|\Phi_3(t, 0)\| \rightarrow 0$ for the stabilized completions but not for the LSC. For output matrix C_f , the LSC is not detectable.

The full-order observer results provide a preview of what to expect from the reduced-order observers. With an appropriate combination of gain matrix parameters, the estimation errors should converge to within $\pm 10^{-3}$ of zero by desired final time $t_f = 45$ for this case's SLSC/RO and ASC/RO observers but not for its LSC/RO observers. The transformation matrices R are

$$\begin{bmatrix} C_e \\ C_R \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \quad \begin{bmatrix} C_f \\ C_R \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

The resulting reduced-order systems are not observable when either the least squares completion or its stabilized version is considered with output matrices C_e and C_f . The observability Grami-

ans for both pairs $(A_{R_{22}}(t), A_{R_{12}}(t))_{\text{SL}}$ and $(A_{R_{22}}(t), A_{R_{12}}(t))_{\text{L}}$, original output matrix C_e , have rank 3 on $t \in (0, 135]$. The minimum dim $(N(W_o(t)))$ is 2 for the reduced-order system resulting from the stabilized least squares completion with output matrix C_f since the rank of $W_o(t)$ is 2 at $t = 0.01$, 3 from 0.02 to 0.37, and 4 from 0.38 to 135.00. The observability Gramian's rank for the reduced-order system from pair $(\tilde{A}(t), C_f)_{\text{L}}$ remains at 3 after equaling 2 from 0.01 to 0.04. The resulting reduced-order system from the alternative stabilized completion with either output matrix is also not observable. Pair $(A_{R_{22}}(t), A_{R_{12}}(t))_{\text{A}}$ has rank $(W_o(t)) = 2$ on $t \in (0, 135]$ for output matrix C_e and rank $(W_o(t)) = 3$ after equaling 2 at $t = 0.01$ for output matrix C_f .

The detectability of the two reduced-order systems for both stabilized completions is apparent with the seemingly successful convergence of the observers to $\tilde{x}(t)$ in Figures 6.59a, 6.59b, 6.60a, and 6.60b. When $\eta = 100$, the estimation errors at $t_f = 45$ are

$$\begin{aligned} \text{SLSC, } C_e : & (1.0 \times 10^{-3}) * \begin{bmatrix} 0.0 & -0.149811 & 0.417293 & 0.0 & 0.000696 & 0.016194 \end{bmatrix}^T; \\ \text{SLSC, } C_f : & (1.0 \times 10^{-3}) * \begin{bmatrix} 0.001676 & -0.130070 & 0.364905 & 0.0 & 0.000588 & 0.014365 \end{bmatrix}^T; \\ \text{ASC, } C_e : & (1.0 \times 10^{-4}) * \begin{bmatrix} 0.0 & 0.073577 & -0.380043 & 0.0 & 0.0 & -0.009414 \end{bmatrix}^T; \\ \text{ASC, } C_f : & (1.0 \times 10^{-4}) * \begin{bmatrix} 0.0 & 0.077769 & -0.401815 & 0.0 & 0.0 & -0.009940 \end{bmatrix}^T. \end{aligned}$$

The polynomial effects from the stabilized least squares completion's Jordan blocks do not disappear when estimating component $i_v(t)$ using the SLSC/RO observer with output matrix C_f . The difference in all components takes longer to go to zero in Figures 6.59c and 6.60c than in Figures 6.53c and 6.54c, respectively. That is, the SLSC/FO and ASC/FO observers with output matrices C_e and C_f reach the same estimate of $\tilde{x}(t)$ faster than the SLSC/RO and ASC/RO observers.

At $t = 13.712793$ and $t = 9.929004$, the observer construction code ends prematurely when constructing the LSC/RO observers with output matrices C_e and C_f , respectively, and with gain matrix parameters $\delta = 0.1$, $\eta = 100$, and $\xi = 0.1$. Decreasing ξ to 0.00001 keeps the code running until $t = 55.221733$ with output matrix C_e and until $t = 49.375845$ with output matrix C_f . As expected, the estimation errors in Figures 6.61a and 6.61b do not give the impression that these LSC/RO observers can be designed to estimate the state. Accepting that the reduced-order observers are not detectable because the full-order observers are not detectable, the maximally reduced observer provides a means of constructing an observer to estimate $\tilde{x}(t)$ when using the least squares completion.

In Chapter 5, the maximally reduced observer's estimate of the linear time-invariant example's state was shown to be independent of the completions' additional dynamics. Neither the completion nor the choice of λ altered the convergence properties or rate. Only parameter ρ and

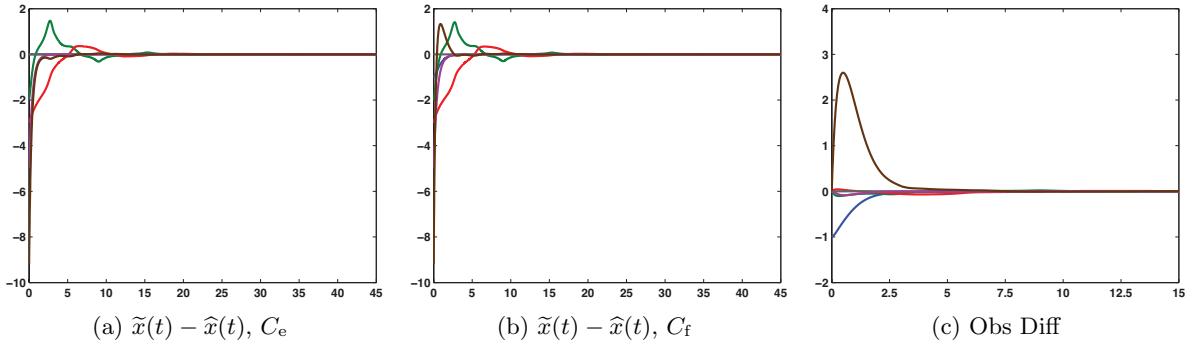


Figure 6.59: SLSC/RO observers' $\tilde{x}(t) - \hat{x}(t)$ for C_e and C_f ; the SLSC/RO observer difference is $C_e - C_f$. (Λ_{4SL} , $\delta = 0.1$, $\eta = 100$, $\xi = 0.1$)

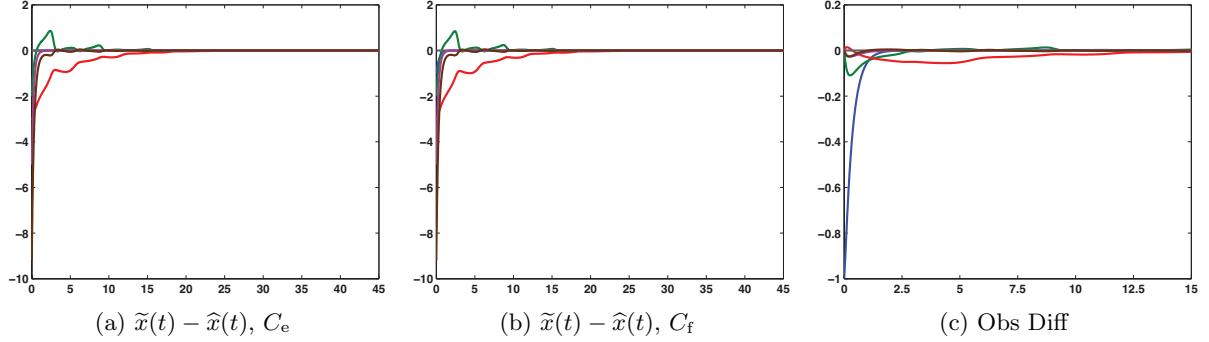


Figure 6.60: ASC/RO observers' $\tilde{x}(t) - \hat{x}(t)$ for C_e and C_f ; the ASC/RO observer difference is $C_e - C_f$. (Λ_{4A} , $\delta = 0.1$, $\eta = 100$, $\xi = 0.1$)

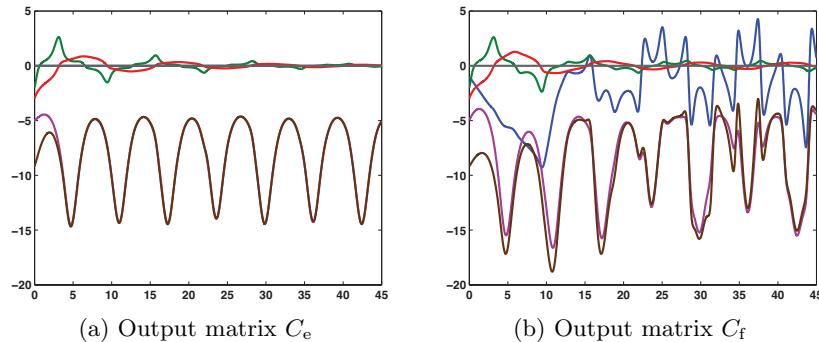


Figure 6.61: LSC/RO observers' $\tilde{x}(t) - \hat{x}(t)$ for C_e and C_f . $e(t)$ fails to converge to zero by $t_f = 45$. (Λ_L , $\delta = 0.1$, $\eta = 100$, $\xi = 10^{-5}$)

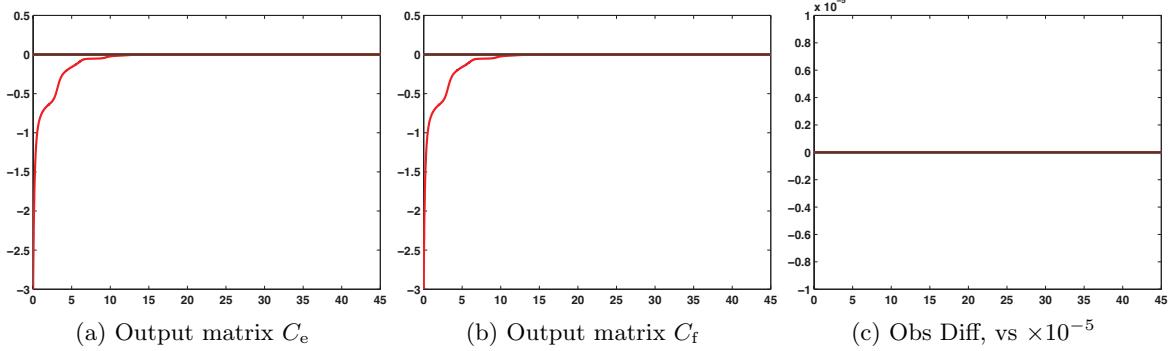


Figure 6.62: SLSC/MR observers' $\tilde{x}(t) - \hat{x}(t)$ for C_e and C_f ; the SLSC/MR observer difference is $C_e - C_f$. ($\Lambda_{4\text{SL}}$, $\delta = 0.1$, $\eta = 1000$, $\xi = 0.1$)

the placement of an observable eigenvalue by the gain matrix influenced how quickly $e(t) \rightarrow 0$. Comparable results are observed in this linear time-varying case.

First, though, C_x in the extension of the output matrix is the same for output matrices C_e and C_f . The first row of C_e was knowingly chosen as one that provided information on a state variable available from the input. Therefore, when tested for linear independence with the rows of $\mathcal{G}(t)\mathcal{F}(t)$, only the second row of C_e is saved. The resulting transformation matrix is

$$\bar{R}(t) = \begin{bmatrix} \mathcal{G}(t)\mathcal{F}(t) \\ C_x \\ C_{\bar{R}} \end{bmatrix} = \begin{bmatrix} -1 & 1 & 0 & -(4 + 2\sin(t)) & 0 & 0 \\ 1 & 0 & 0 & 0 & -(2 + \sin(t)) & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ -(1/3)\sin(t/3) & 0 & 0 & -1 & 1 & -1 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}.$$

Figures 6.53c, 6.54c, 6.59c, and 6.60c demonstrate output matrices C_e and C_f produce similar but not exact results when considering a specific observer (full-order or reduced-order) and stabilized completion. With no distinction between their extended output equations, the transformed systems from which the maximally reduced observers are constructed are the same for these output matrices. For example, consider the SLSC/MR observer. The observability Gramians have rank 1 on $t \in (0, 135]$ for both pairs $(A_{\bar{R}_{22}}(t), A_{\bar{R}_{12}}(t))_{\text{SL}}$. The estimation error when $\delta = 0.1$, $\eta = 1000$, $\xi = 0.1$ is displayed in Figure 6.62a for output matrix C_e and in Figure 6.62b for output matrix C_f . In Figure 6.62c, the difference between these maximally reduced observers ($C_e - C_f$) is zero (no noise appears).

For the same gain matrix parameters from the previous comparison, the differences between the SLSC/MR observer and the maximally reduced observer constructed using the alterna-

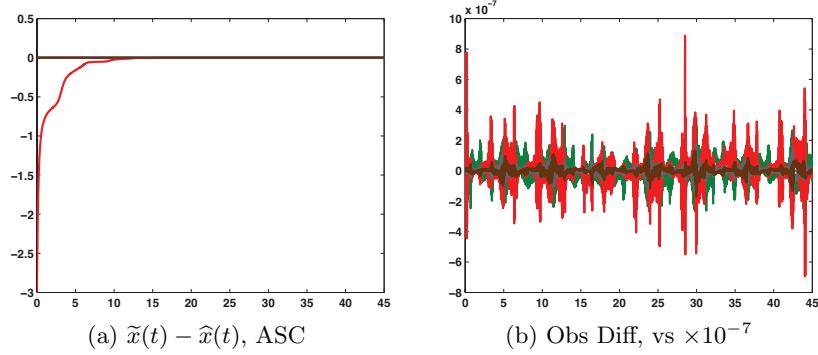


Figure 6.63: ASC/MR observer's $\tilde{x}(t) - \hat{x}(t)$; the observer difference is SLSC/MR – ASC/MR. (Case 4, Λ_{4A} , Λ_{4SL} , $\delta = 0.1$, $\eta = 1000$, $\xi = 0.1$)

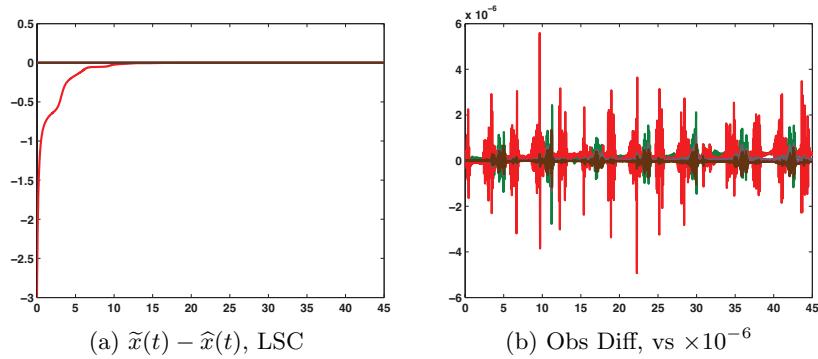


Figure 6.64: LSC/MR observer's $\tilde{x}(t) - \hat{x}(t)$; the observer difference is SLSC/MR – LSC/MR. (Case 4, Λ_{4L} , Λ_{4SL} , $\delta = 0.1$, $\eta = 1000$, $\xi = 0.1$)

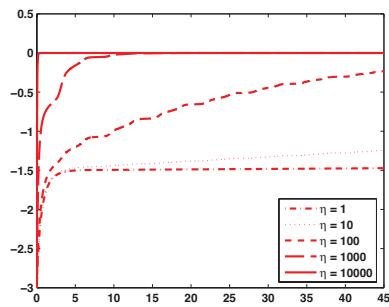


Figure 6.65: $\tilde{x}(t) - \hat{x}(t)$ progression in $i_l(t)$ as η is varied, MR observer. (Case 4, $\delta = 0.1$, $\xi = 0.1$)

tive stabilized completion (ASC/MR observer) in Figure 6.63b and between the SLSC/MR observer and the maximally reduced observer constructed using the least squares completion (LSC/MR observer) in Figure 6.64b are zero except for some numerical error. The pairs $(A_{\bar{R}_{22}}(t), A_{\bar{R}_{12}}(t))_A$ and $(A_{\bar{R}_{22}}(t), A_{\bar{R}_{12}}(t))_L$ are also observable since $\text{rank}(W_o(t)) = 1$ in finite time (by $t = 0.01$).

The convergence of the maximally reduced observer is affected by the gain matrix, demonstrated in Figure 6.65 by the estimation error progression in state variable $i_l(t)$ as η is varied. For $\eta = 1000$, $e(t) = (1.0 \times 10^{-6}) * \begin{bmatrix} 0.020666 & 0.022225 & -0.007757 & 0.0 & 0.007819 & -0.129750 \end{bmatrix}^T$ is the estimation error at $t_f = 45$, but overall, increasing η decreases the amount of time it takes the maximally reduced observer to estimate $\tilde{x}(t)$.

Case 4 is important for two reasons. First, it supports the claim that the maximally reduced observer is the same no matter the completion or stabilization parameter matrix, extending the experimental results from linear time-invariant to linear time-varying systems of DAEs. Second, this case provides a linear time-varying example where the LSC/MR observer can be constructed even though the full-order and reduced-order systems are unobservable. The benefits of the maximally reduced observer will continue to be seen in the next subsection.

6.2.2 Example with Negative Resistors

System Introduction

Switching the resistors from positive to negative adds energy into the system, affecting the computation of the gain matrix and, hence, the ability to construct the observers. The linear time-varying system of DAEs for this subsection is the same as system (6.9) except for equations (6.9d) and (6.9e), which are now

$$\begin{aligned} -e_1(t) + e_2(t) + (4 + 2 \sin(t)) i_{r_1}(t) &= 0, \\ e_1(t) + (2 + \sin(t)) i_{r_2}(t) &= 0. \end{aligned}$$

Output matrices C_a , C_b , C_d , and C_f are selected to construct the output equations with this negative resistor system. This modified linear time-varying system of DAEs is observable for each of these output matrices.

Changing the resistors does not alter the consistent initial conditions but does modify $\mathcal{G}(t)\mathcal{F}(t)$ from the constraints characterizing the solution manifold. Since the resistors do not appear in the development of the consistent initial condition from Subsection 6.2.1, vector $\tilde{x}(0)$ remains $\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & -16/5 \end{bmatrix}^T$. The initial condition for the full-order observer is again taken as $\hat{x}(0) = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 & 6 \end{bmatrix}^T$ and is transformed accordingly for the other two ob-

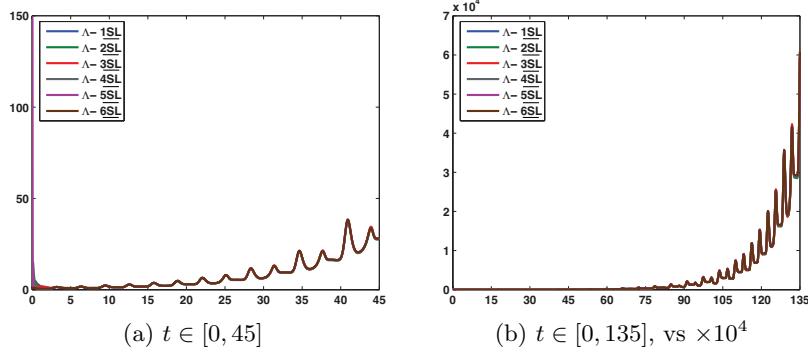


Figure 6.66: $\|\Phi(t, 0)\|$, on two intervals of time, fails to converge to zero for the six $\Lambda_{\underline{\text{SL}}}$ options.

servers. With the resistors being negative instead of positive,

$$\mathcal{G}(t)\mathcal{F}(t) = \begin{bmatrix} -1 & 1 & 0 & (4 + 2\sin(t)) & 0 & 0 \\ 1 & 0 & 0 & 0 & (2 + \sin(t)) & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ -(1/3)\sin(t/3) & 0 & 0 & -1 & 1 & -1 \end{bmatrix} \quad (6.14)$$

from MATLAB's null command (terms $(4+2\sin(t))$ and $(2+\sin(t))$ are now without the negative sign). Note, $\mathcal{G}(t)\mathcal{B}(t)$ for this example with negative resistors is unchanged from (6.12).

In order to distinguish notation between the two time-varying examples, an underline is added to identify the pairs and stabilization parameter matrices for this example with negative resistors: $(\cdot, \cdot)_{\underline{\text{SL}}}$ and $\Lambda_{\underline{\text{SL}}}$ for the stabilized least squares completion; $(\cdot, \cdot)_{\underline{\text{A}}}$ and $\Lambda_{\underline{\text{A}}}$ for the alternative stabilized completion; and $(\cdot, \cdot)_{\underline{\text{L}}}$ and $\Lambda_{\underline{\text{L}}}$ for the least squares completion.

Λ Selection

The selection of the stabilization parameter matrices for the stabilized least squares completion and the alternative stabilized completion of this subsection follows the criteria outlined in Subsection 6.2.1.

STABILIZED LEAST SQUARES COMPLETION:

$$\begin{aligned} \Lambda_{1\underline{\text{SL}}} &= \{1.2, 1.4, 1.6, 1.8, 2.0, 2.2\}, & \Lambda_{2\underline{\text{SL}}} &= \{1.0, 1.5, 2.0, 2.5, 3.0, 3.5\}, \\ \Lambda_{3\underline{\text{SL}}} &= \{3.5, 3.0, 2.5, 2.0, 1.5, 1.0\}, & \Lambda_{4\underline{\text{SL}}} &= \{35, 30, 25, 20, 15, 10\}, \\ \Lambda_{5\underline{\text{SL}}} &= \{350, 300, 250, 200, 150, 100\}, & \Lambda_{6\underline{\text{SL}}} &= \{3500, 3000, 2500, 2000, 1500, 1000\} \end{aligned}$$

- Figure 6.66a shows $\|\Phi(t, 0)\|$ fails to converge to zero by $t_f = 45$ for the six stabilized least squares completions being considered. The plots of these norms on the extended time interval $t \in [0, 135]$ in Figure 6.66b imply that each norm of $\Phi(t, 0)$ will continue to grow as $t \rightarrow \infty$,

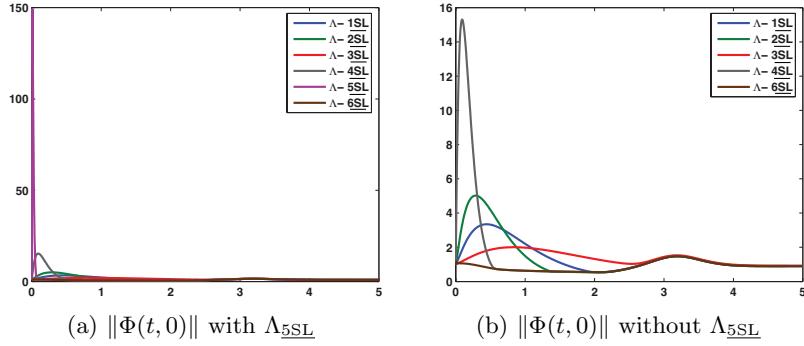


Figure 6.67: $\|\Phi(t, 0)\|$ comparison. Shortened time intervals provide closer views of norms' initial behaviors with and without result for component $\Lambda_{5\text{SL}}$.

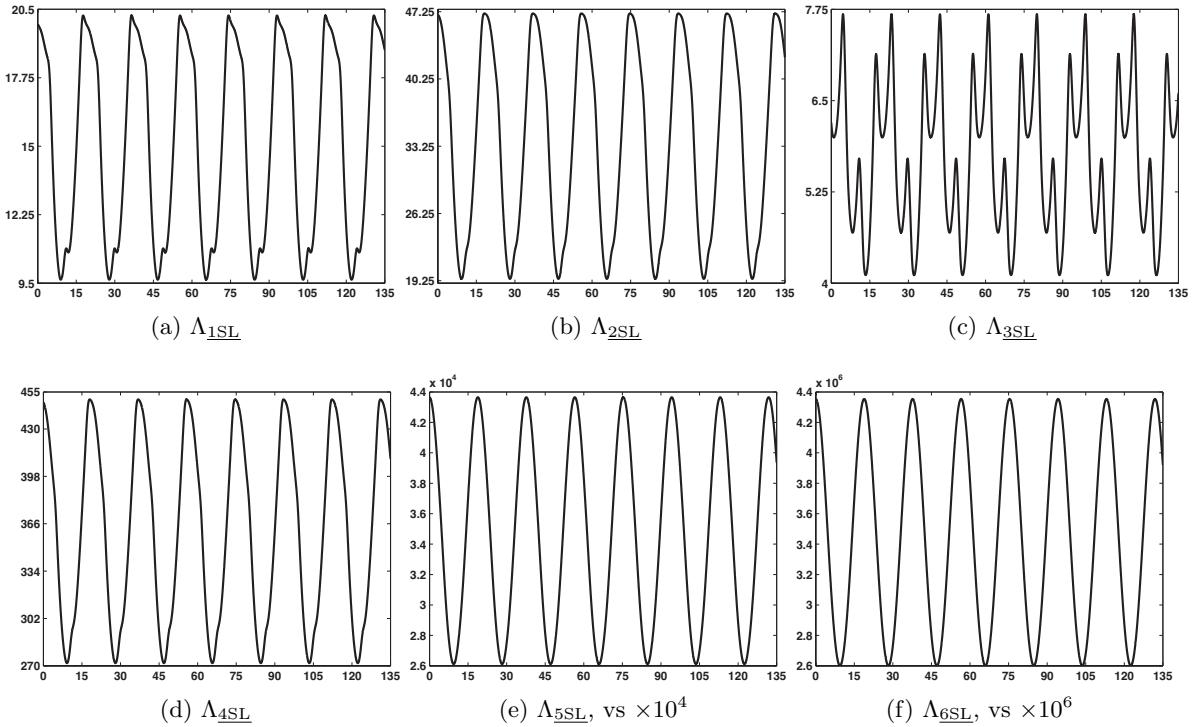


Figure 6.68: $\|\tilde{A}(t)\|$ for six choices of Λ_{SL} . Each $\|\tilde{A}(t)\|$ is bounded.

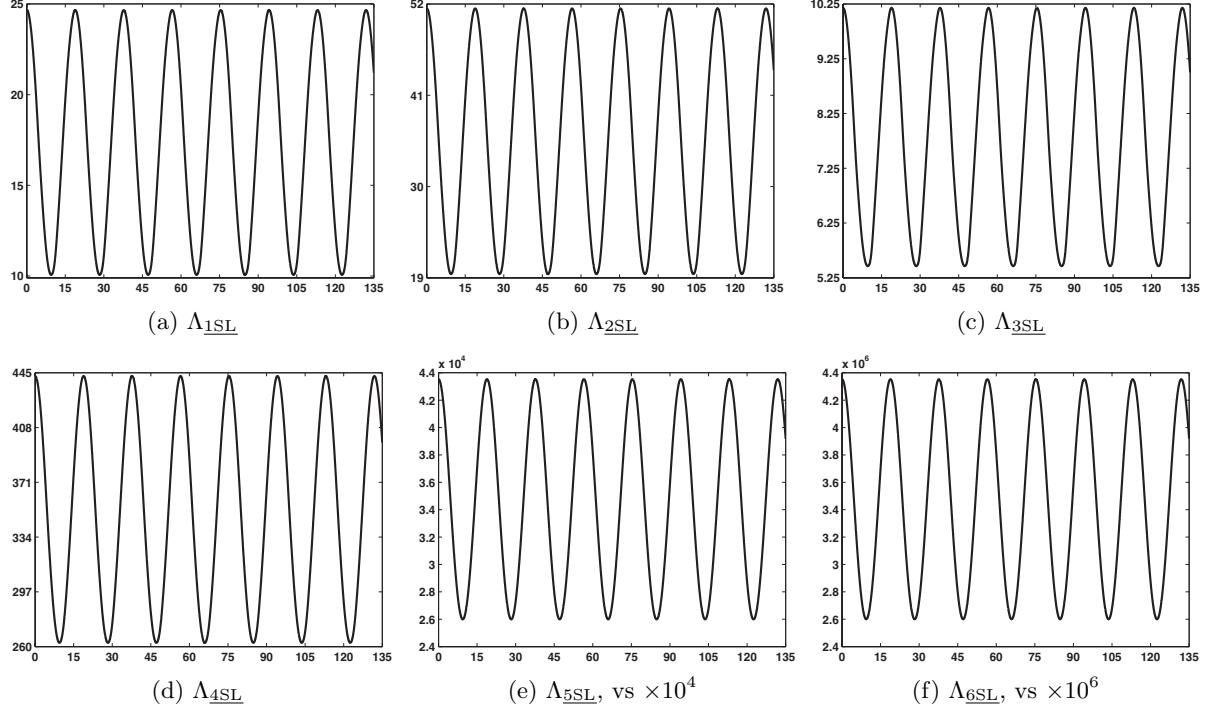


Figure 6.69: $\|\tilde{B}(t)\|$ for six choices of $\Lambda_{\underline{SL}}$. Each $\|\tilde{B}(t)\|$ is bounded.

which is expected since energy is being added to the system by the negative resistors.

A peculiarity in the behaviors of $\|\Phi(t, 0)\|$ near $t_0 = 0$ is slightly visible in Figure 6.66a, so closer views are included in Figure 6.67. The norm with the greatest initial growth does not correspond with Λ_{6SL} , the stabilization parameter matrix with the largest diagonal entries, but with Λ_{5SL} . Once the norm of $\Phi(t, 0)$ in Figure 6.67a resulting from Λ_{5SL} is removed, Figure 6.67b reveals stabilizing with Λ_{6SL} eliminates the initial growth seen in the other norms.

2. Although matrices $\tilde{A}(t)$ and $\tilde{B}(t)$ are bounded for each choice of $\Lambda_{\underline{SL}}$, the vertical scales of Figures 6.68 and 6.69 reveal significant growth in the bounds as the magnitude of the $\Lambda_{\underline{SL}}$ entries is increased. These large bounds are indicative of growth in $\tilde{A}(t)$ that may cause computational inefficiency or an integration tolerance error while computing the Riccati equation's $S_L(t)$. The smallest upper and lower bounds of $\|\tilde{A}(t)\|$ and their smallest difference occur for Λ_{3SL} in Figure 6.68c but the solution is not smooth. Also note how the order of the $\Lambda_{\underline{SL}}$ entries affects $\|\tilde{A}(t)\|$. By reversing the order of the Λ_{3SL} entries in Λ_{2SL} , Figure 6.68b shows the upper and lower bounds and their difference increase but the solution is smooth.

3. Table 6.7 presents the number of eigenvalues with a negative real part that $\tilde{A}(t)$ has for each $\Lambda_{\underline{SL}}$. Not one of the eigenvalues has a constant real part but all real parts have oscillatory

Table 6.7: For each $\Lambda_{\underline{SL}}$, the number of $\tilde{A}(t)$'s eigenvalues with a negative real part.

	$t \in [0, 45]$	$t \in [0, 135]$
$\Lambda_{1\underline{SL}}$	1	1
$\Lambda_{2\underline{SL}}$	1	1
$\Lambda_{3\underline{SL}}$	2	2
$\Lambda_{4\underline{SL}}$	1	1
$\Lambda_{5\underline{SL}}$	2	2
$\Lambda_{6\underline{SL}}$	4	4

behavior that is periodic. The only stabilization parameter matrix producing the desired four eigenvalues with a negative real part is $\Lambda_{6\underline{SL}}$. Two of these real parts are bounded above by -1000 and below by -2000 , again causing concern about this $\tilde{A}(t)$'s growth and possible computational implications for the observer construction code. Two eigenvalues' real parts have the desired sign when stabilizing with either $\Lambda_{3\underline{SL}}$ or $\Lambda_{5\underline{SL}}$, while the additional dynamics resulting from the three remaining choices of $\Lambda_{\underline{SL}}$ have just one such eigenvalue.

4. The rank check of $W_o(t)$ for pairs $(\tilde{A}(t), C_a)_{\underline{SL}}$, $(\tilde{A}(t), C_d)_{\underline{SL}}$, and $(\tilde{A}(t), C_f)_{\underline{SL}}$ on the $t \in (0, 135]$ interval is summarized in Tables 6.8 and 6.9. Recall, the rank is calculated after solving system (3.4), so the times at which the rank is calculated correlate with the 0.01 output time step for MATLAB's ode45 solver.

Table 6.8 presents unexpected rank ($W_o(t)$) results for pair $(\tilde{A}(t), C_a)_{\underline{SL}}$ when stabilizing with either $\Lambda_{5\underline{SL}}$ or $\Lambda_{6\underline{SL}}$. The pairs are initially observable, but after $t = 128.25$ for $\Lambda_{5\underline{SL}}$ and $t = 115.72$ for $\Lambda_{6\underline{SL}}$, the ranks of their observability Gramians decrease to 5 and remain there until the end of the interval. This rank decrease lasts longer than any rank fluctuation seen in the example with positive resistors. Notice, if the rank had been checked on just the $t \in (0, 45]$ interval, this phenomenon would not have been detected. As a response to these numerical uncertainties about future ranks, nothing is assumed about the systems' observability past $t = 135$. Only stabilization parameter matrix $\Lambda_{3\underline{SL}}$ produces an observable completion with output matrix C_a from 0.01 to 135.00. The three remaining stabilized least squares completions are also observable on the interval under consideration since their observability Gramian ranks reach 6 in finite time (by $t = 0.02$) and no rank drops occur.

These six stabilized least squares completions are not observable with output equation $\tilde{y}(t) = C_d \tilde{x}(t)$ on the $t \in (0, 135]$ interval. Again, some of the observability Gramians are not well-conditioned, but for output matrix C_d , these undesirable results correspond with stabilization parameter matrices $\Lambda_{1\underline{SL}}$, $\Lambda_{2\underline{SL}}$, and $\Lambda_{4\underline{SL}}$. For matrix $\Lambda_{1\underline{SL}}$, the rank of $W_o(t)$ is 4 from 0.01 to 0.31 and 105.29 to 135.00 and is 5 from 0.32 to 105.28. Similarly, on the intermediate intervals of time 0.17 to 116.85 for $\Lambda_{2\underline{SL}}$ and 0.08 to 94.70 for $\Lambda_{4\underline{SL}}$, the observability Gramian has rank

Table 6.8: For each $\Lambda_{\underline{\text{SL}}}$, the ranks of $W_o(t)$ and their time intervals for two output matrices.

rank ($W_o(t)$)	C_a		C_d	
	5	6	4	5
$\Lambda_{1\underline{\text{SL}}}$	0.01	0.02 to 135.00	*	*
$\Lambda_{2\underline{\text{SL}}}$	0.01	0.02 to 135.00	*	*
$\Lambda_{3\underline{\text{SL}}}$		0.01 to 135.00	0.01 to 0.15	0.16 to 135.00
$\Lambda_{4\underline{\text{SL}}}$	0.01	0.02 to 135.00	*	*
$\Lambda_{5\underline{\text{SL}}}$	128.25 to 135.00	0.01 to 128.24	0.01 to 135.00	
$\Lambda_{6\underline{\text{SL}}}$	115.72 to 135.00	0.01 to 115.71	0.01 to 135.00	

Table 6.9: For each $\Lambda_{\underline{\text{SL}}}$, the ranks of $W_o(t)$ and their time intervals for output matrix C_f .

rank ($W_o(t)$)	C_f			
	2	3	4	5
$\Lambda_{1\underline{\text{SL}}}$	0.01 to 0.04	0.05 to 0.17	*	*
$\Lambda_{2\underline{\text{SL}}}$	0.01 to 0.03	0.04 to 0.14	*	*
$\Lambda_{3\underline{\text{SL}}}$	*	*	0.18 to 0.61	0.62 to 135.00
$\Lambda_{4\underline{\text{SL}}}$	0.01	0.02 to 0.08	*	*
$\Lambda_{5\underline{\text{SL}}}$		0.01 to 0.02	0.03 to 135.00	
$\Lambda_{6\underline{\text{SL}}}$			0.01 to 135.00	

5 rather than 4. Since the dimension of the unobservable subspace decreases from 2 to 1 and remains there when stabilizing with matrix $\Lambda_{3\underline{\text{SL}}}$, it is a better choice for this system than either $\Lambda_{5\underline{\text{SL}}}$ or $\Lambda_{6\underline{\text{SL}}}$, which have unobservable subspaces with dimension 2 on the entire interval.

The observability Gramian's rank decrease repeats for pairs $(\tilde{A}(t), C_f)_{\underline{\text{SL}}}$ stabilized with $\Lambda_{1\underline{\text{SL}}}$, $\Lambda_{2\underline{\text{SL}}}$, and $\Lambda_{4\underline{\text{SL}}}$. After equaling 4 from 0.18 to 1.30 for stabilization parameter matrix $\Lambda_{1\underline{\text{SL}}}$, the rank of $W_o(t)$ becomes 5 at $t = 1.31$ but reduces to 4 for $t \in [122.69, 135.00]$. Although $\text{rank}(W_o(t)) = 4$ from 0.15 to 0.69 and increases to 5 at $t = 0.70$ for matrix $\Lambda_{2\underline{\text{SL}}}$, the rank of the observability Gramian does not remain at 5 and changes to 4 at $t = 129.33$ for the rest of the interval. When stabilizing with $\Lambda_{4\underline{\text{SL}}}$, the rank of $W_o(t)$ is 5 from 0.32 to 114.20 in between 4 from 0.09 to 0.31 and 114.21 to 135.00. For stabilization parameter matrix $\Lambda_{3\underline{\text{SL}}}$, the rank of the observability Gramian decreases to 2 at $t = 0.04$ after equaling 2 at $t = 0.01$ and $t = 0.02$ and 3 at $t = 0.03$ but increases to 3 again for $t \in [0.05, 0.17]$. This observability Gramian's rank eventually equals 5. Following the analysis in Case 2 on the ASC/FO observers resulting from stabilization parameters Λ_{3A} and Λ_{4A} , this fluctuation, although undesirable, is not expected to affect observer construction. The observability Gramians' ranks do not decrease when stabilizing with either $\Lambda_{5\underline{\text{SL}}}$ or $\Lambda_{6\underline{\text{SL}}}$ but their maximum values are 4.

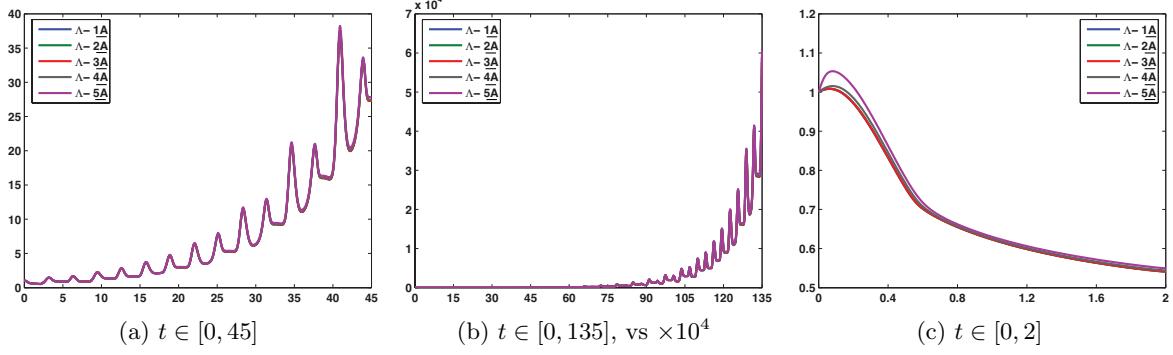


Figure 6.70: $\|\Phi(t, 0)\|$ comparison. $\|\Phi(t, 0)\|$ fails to converge to zero for the five Λ_A options, and a shortened time interval provides a closer view of norms' initial behaviors.

Λ_{SL} SELECTION: Focusing on Criterion 4 alone, stabilization parameter matrix $\Lambda_{3\text{SL}}$ is the most desirable choice. This matrix is the only one to produce a matrix $\tilde{A}(t)$ that, when paired with either of the three output matrices, results in an observability Gramian without a rank drop. Even though the most favorable results of Criteria 1 and 3 occur when stabilizing with $\Lambda_{6\text{SL}}$, the results arising from Criterion 2's norm check suggest this stabilization parameter matrix will pose numerical challenges to the algorithm for computing the gain matrix. Considering Criteria 1 and 3 together, the next best results occur when stabilizing with matrix $\Lambda_{3\text{SL}}$. The norm of $\tilde{A}(t)$ resulting from this stabilization parameter matrix is not a smooth curve but its bounds out of the six options are the smallest. Using the guidance of these criteria, our chosen stabilization parameter matrix for the stabilized least squares completion is $\Lambda_{3\text{SL}}$.

ALTERNATIVE STABILIZED COMPLETION:

$$\begin{aligned}\Lambda_{1A} &= \{4.0, 3.5, 3.0, 2.5\}, & \Lambda_{2A} &= \{3.2, 3.4, 3.6, 3.8\}, \\ \Lambda_{3A} &= \{3.8, 3.6, 3.4, 3.2\}, & \Lambda_{4A} &= \{8.0, 7.0, 6.0, 5.0\}, \\ \Lambda_{5A} &= \{40, 35, 30, 25\}\end{aligned}$$

1. For each of the five Λ_A being considered, Figure 6.70a shows the growth of $\|\Phi(t, 0)\|$ on the $t \in [0, 45]$ interval and Figure 6.70b implies the norm of $\Phi(t, 0)$ will continue to grow as time goes to infinity. The initial growth in the norms of $\Phi(t, 0)$ is unlike the behavior seen in Figure 6.67 for the stabilized least squares completions. The $t \in [0, 2]$ interval in Figure 6.70c reveals how close the norms are for these five alternative stabilized completions. A Λ_A option with entries of larger magnitude is not included since manipulating the entries to make the initial growth vanish was unnecessary.

2. The largest upper and lower bounds and difference between bounds in Figure 6.71 correspond with Λ_{5A} . Out of the five options, the entries of this stabilization parameter matrix have the largest magnitude. The only norm with a visibly distinct and perhaps numerically

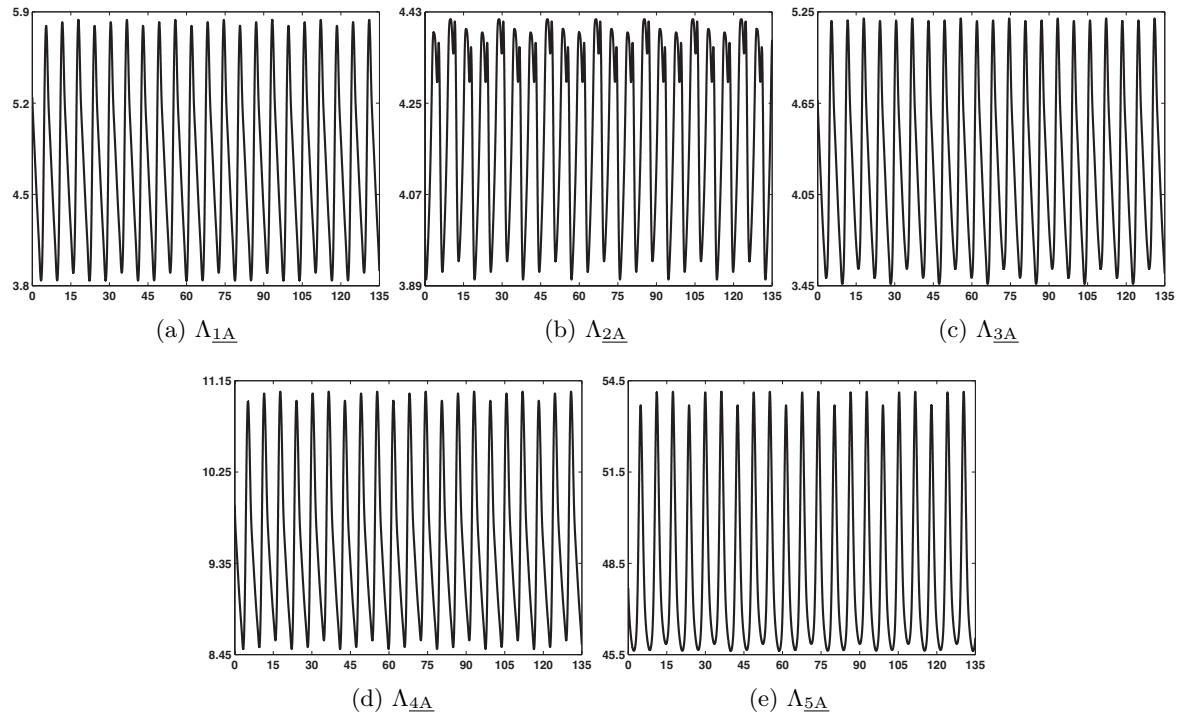


Figure 6.71: $\|\tilde{A}(t)\|$ for five choices of $\Lambda_{\underline{A}}$. Each $\|\tilde{A}(t)\|$ is bounded.

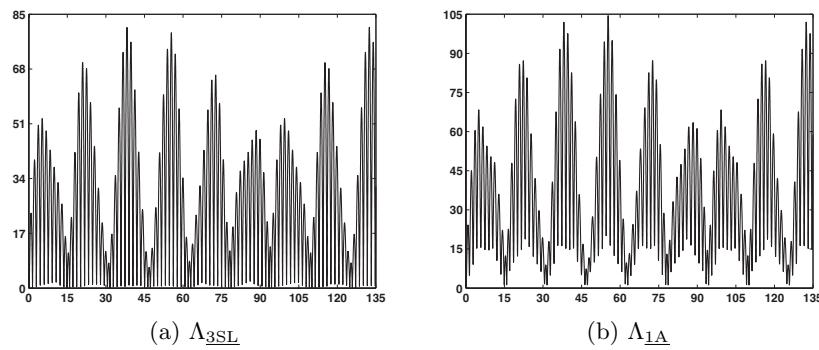


Figure 6.72: $\|\tilde{B}(t)v(t)\|$ for an SLSC and an ASC. $\|\tilde{B}(t)\|$ cannot be computed for the ASC, but comparing $\|\tilde{B}(t)v(t)\|$ for the stabilized completions suggests the ASC $\|\tilde{B}(t)\|$ is bounded.

Table 6.10: For each $\Lambda_{\underline{A}}$, the number of $\tilde{A}(t)$'s eigenvalues with a negative real part.

	$t \in [0, 45]$	$t \in [0, 135]$
$\Lambda_{1\underline{A}}$	3	3
$\Lambda_{2\underline{A}}$	3	3
$\Lambda_{3\underline{A}}$	3	3
$\Lambda_{4\underline{A}}$	4	4
$\Lambda_{5\underline{A}}$	4	4

undesirable oscillation appears in Figure 6.71b for the alternative stabilized completion with matrix $\Lambda_{2\underline{A}}$. In terms of the stabilization parameter matrices, the smallest to largest order of $\|\tilde{A}(t)\|$ based on the upper and lower bounds and on the difference between the bounds is $\{\Lambda_{2\underline{A}}, \Lambda_{3\underline{A}}, \Lambda_{1\underline{A}}, \Lambda_{4\underline{A}}, \Lambda_{5\underline{A}}\}$. It is interesting to note Figures 6.71b and 6.71c and how the norm changes when the order of the entries of $\Lambda_{2\underline{A}}$ are reversed.

As with the Criterion 2 results in Subsection 6.2.1 for selecting Λ_A , the norm of $\tilde{B}(t)$ cannot be computed when considering different $\Lambda_{\underline{A}}$ due to how the linear time-varying alternative stabilized completion is derived. With the negative resistors, coefficient matrix $F(t)$ becomes

$$\begin{bmatrix} -(1/3) \sin(t/3) & 0 & 0 & -1 & 1 & -1 \\ 0 & 2 \sin(2t) & 1 & 1 & 0 & 0 \\ 0 & -1 & \exp(-t) & 0 & 0 & 0 \\ -1 & 1 & 0 & (4 + 2 \sin(t)) & 0 & 0 \\ 1 & 0 & 0 & 0 & (2 + \sin(t)) & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

The bounded matrix and constant rank conditions hold for this modified example's time-varying system of DAEs, so matrix $\tilde{B}(t)$ is bounded. A figure is included again for comparing norms of $\|\tilde{B}(t)v(t)\|$ for the two stabilized completions (see Figure 6.72), knowing the behavior and magnitude of $\|\tilde{B}(t)\|$ for the stabilized least squares completion from Figure 6.69.

3. Table 6.10 displays the number of $\tilde{A}(t)$'s eigenvalues with a negative real part for each $\Lambda_{\underline{A}}$. Two of the eigenvalues resulting from the additional dynamics equal $-\lambda_3$ and $-\lambda_4$ if generally $\Lambda_{\underline{A}} = \{\lambda_1, \lambda_2, \lambda_3, \lambda_4\}$. The remaining eigenvalues have oscillating real parts that are periodic. One other eigenvalue has a negative real part when stabilizing with $\Lambda_{1\underline{A}}$, $\Lambda_{2\underline{A}}$, and $\Lambda_{3\underline{A}}$, but the desired eigenvalue behavior occurs for stabilization parameter matrices $\Lambda_{4\underline{A}}$ and $\Lambda_{5\underline{A}}$.

4. The observability check for these alternative stabilized completions with output matrices C_a , C_d , and C_f returns only one rank drop and no rank fluctuations. Tables 6.11 and 6.12 summarize the results.

Table 6.11: For each $\Lambda_{\underline{A}}$, the ranks of $W_o(t)$ and their time intervals for two output matrices.

	C_a		C_d
rank $(W_o(t))$	5	6	4
$\Lambda_{1\underline{A}}$		0.01 to 135.00	0.01 to 135.00
$\Lambda_{2\underline{A}}$	131.95 to 135.00	0.01 to 131.94	0.01 to 135.00
$\Lambda_{3\underline{A}}$		0.01 to 135.00	0.01 to 135.00
$\Lambda_{4\underline{A}}$		0.01 to 135.00	0.01 to 135.00
$\Lambda_{5\underline{A}}$		0.01 to 135.00	0.01 to 135.00

Table 6.12: For each $\Lambda_{\underline{A}}$, the ranks of $W_o(t)$ and their time intervals for output matrix C_f .

	C_f		
rank $(W_o(t))$	2	3	4
$\Lambda_{1\underline{A}}$	0.01	0.02 to 0.22	0.23 to 135.00
$\Lambda_{2\underline{A}}$	0.01 to 0.02	0.03 to 0.28	0.29 to 135.00
$\Lambda_{3\underline{A}}$	0.01	0.02 to 0.28	0.29 to 135.00
$\Lambda_{4\underline{A}}$	0.01	0.02 to 0.11	0.12 to 135.00
$\Lambda_{5\underline{A}}$		0.01 to 0.03	0.04 to 135.00

The rank drop occurs for pair $(\tilde{A}(t), C_a)_{\underline{A}}$ when stabilizing with $\Lambda_{2\underline{A}}$. This pair's observability Gramian has rank 6 from 0.01 to 131.94 then 5 from 131.95 to 135.00. When considering the $t \in (0, 135]$ interval, the other alternative stabilized completions are observable with output matrix C_a . Each pair $(\tilde{A}(t), C_d)_{\underline{A}}$ has rank 4 on the entire interval, and the minimum dimension of the unobservable subspace for each pair $(\tilde{A}(t), C_f)_{\underline{A}}$ is 2.

$\Lambda_{\underline{A}}$ SELECTION: Matrix $\Lambda_{2\underline{A}}$ is not selected due to Criterion 4 and the rank drop, and matrix $\Lambda_{5\underline{A}}$ is not selected due to Criterion 2 and the potential effects of an $\tilde{A}(t)$ with larger bounds on computational efficiency. Although stabilization parameter matrix $\Lambda_{4\underline{A}}$ has the desired number of eigenvalues with negative real part, its $\|\tilde{A}(t)\|$ has the second largest upper and lower bounds. Our chosen stabilization parameter matrix to compute the alternative stabilized completion is $\Lambda_{4\underline{A}}$, but the decision takes into consideration additional analysis in the upcoming Case 1.

LEAST SQUARES COMPLETION:

Recall, Λ is a zero matrix ($\Lambda_{\underline{L}} = \{0, 0, 0, 0, 0, 0\}$) for the least squares completion since the differential polynomial used to determine the derivative array equations is not stabilized.

- As with the stabilized least squares completion and the alternative stabilized completion of the circuit example system with negative resistors, coefficient matrix $\tilde{A}(t)$ from the least

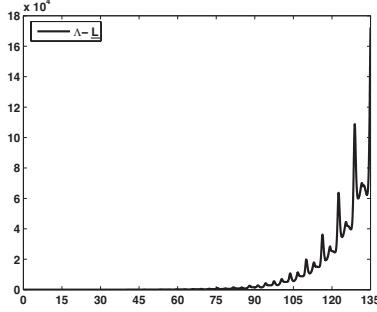


Figure 6.73: $\|\Phi(t, 0)\|$ fails to converge to zero for the LSC ($\Lambda_{\underline{L}}$). vs $\times 10^4$

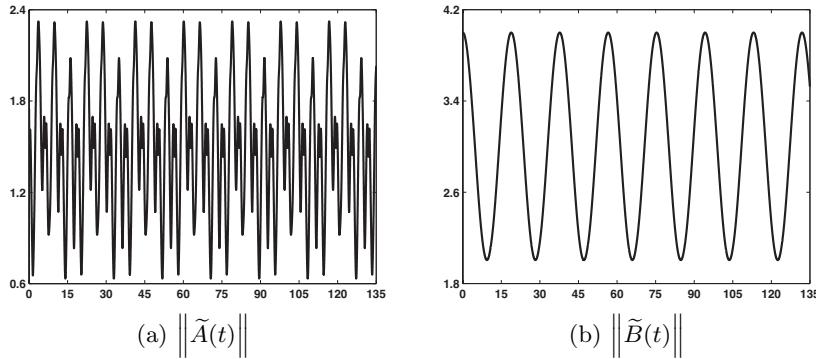


Figure 6.74: $\|\tilde{A}(t)\|$ and $\|\tilde{B}(t)\|$ are bounded for the LSC ($\Lambda_{\underline{L}}$).

squares completion is not exponentially asymptotically stable. Figure 6.73 shows $\|\Phi(t, 0)\|$ fails to converge to zero on the extended time interval.

2. Figure 6.74 displays $\|\tilde{A}(t)\|$ and $\|\tilde{B}(t)\|$ are bounded for the least squares completion.
3. None of $\tilde{A}(t)$'s eigenvalues have real parts that remain negative for all of $t \in [0, 45]$.
4. The observability Gramians of pairs $(\tilde{A}(t), C_a)_{\underline{L}}$, $(\tilde{A}(t), C_d)_{\underline{L}}$, and $(\tilde{A}(t), C_f)_{\underline{L}}$ are all well-conditioned. Even though the algorithms for determining the least squares completion and the stabilized least squares completion are interchangeable, the rank drops seen earlier for some choices of $\Lambda_{\underline{SL}}$ do not occur when differentiating with polynomial $\frac{d}{dt} + \Lambda_{\underline{L}} = \frac{d}{dt}$. Pair $(\tilde{A}(t), C_a)_{\underline{L}}$ is observable because $\text{rank}(W_o(t))$ equals 6 in finite time (by $t = 0.02$). The rank of the observability Gramian for the least squares completion with output matrix C_d is 4 for $t \in (0, 135]$. The rank of $W_o(t)$ for pair $(\tilde{A}(t), C_f)_{\underline{L}}$ reaches 4 after equaling 2 at $t = 0.01$ and 3 from 0.02 to 0.18.

Results

In the remaining figures, the color and state variable combinations are blue for $e_1(t)$, green for $e_2(t)$, red for $i_l(t)$, gray for $i_{r_1}(t)$, magenta for $i_{r_2}(t)$, and brown for $i_v(t)$ (the same as originally listed in Subsection 6.2.1). Also, black is used when plotting a norm. In captions, vs represents vertical scale; Sol stands for solution; Obs designates observer; and Diff identifies difference.

Figure 6.75a presents the SLSC solution when stabilizing with $\Lambda_{3\text{SL}}$. Although the results in Figures 6.75a and 6.10a are similar, the extended time interval reveals how different the solutions are for the two time-varying examples. Unlike the solution of system (6.9), the solution of the linear time-varying example with negative resistors is not periodic and the oscillations grow in amplitude as time progresses. Figure 6.76a shows the ASC solution when stabilizing with $\Lambda_{4\text{A}}$ on $t \in [0, 135]$. The difference between the SLSC and ASC solutions ($\text{SLSC} - \text{ASC}$) appears bounded on the $t \in [0, 45]$ interval in Figure 6.76b but is growing, which can be seen in Figure 6.76c. Similarly, the difference between the SLSC and LSC solutions ($\text{SLSC} - \text{LSC}$) on the shorter interval suggests the solutions are the same except for some numerical error. However, once Figure 6.77c is considered instead of Figure 6.77b, the difference between the solutions is growing. The stabilization of the additional dynamics may not be as effective for longer intervals of time, which requires further investigation.

Introduced in the **Results** preliminaries for the example with positive resistors, δ , η , and ξ represent the gain matrix parameters. In the following cases, the values of δ , η , and ξ are specified with each observer unless the context is clear when discussing the gain matrix parameters. Also, P_2 is kept as an identity matrix.

Case 1 (output matrix C_a) focuses on constructing observers using the stabilized completions. Stabilization parameter matrix $\Lambda_{4\text{A}}$ rather than $\Lambda_{1\text{A}}$ is selected after examining solutions and full-order observer results for the alternative stabilized completion. This case reveals the failure of $\|\Phi_R(t, 0)\|$ to converge to zero for one completion's reduced-order system does not imply convergence failure for another completion's reduced-order system.

Case 2 (output matrix C_b) begins by studying the SLSC/RO observer and investigates different methods for influencing observer convergence when the growth of $\|\Phi_R(t, 0)\|$ is disconcerting. After commenting on the ASC/RO observer, this case explains why the full-order observer and especially the process for constructing the maximally reduced observer are preferable to the SLSC/RO observer.

Case 3 (output matrix C_d) attempts to construct full-order and reduced-order observers using each completion for systems that are not observable. Even though a system may be unobservable, this case shows a nonsingular extended output matrix C_Ξ allows for the state to be determined.

Case 4 (output matrix C_f) examines the effects on estimation and convergence as the order of the observer is reduced.

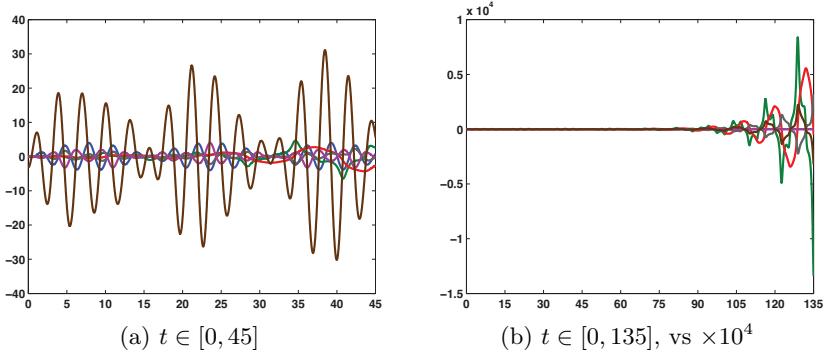


Figure 6.75: SLSC solution (Λ_{SL}) on two intervals of time. The solution of the linear time-varying example with negative resistors is neither periodic nor bounded.

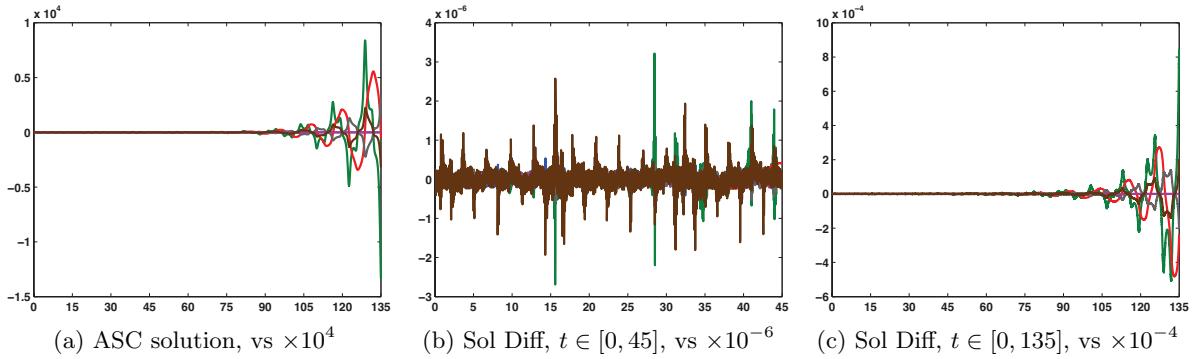


Figure 6.76: The ASC solution (Λ_{4A}) and the SLSC solution (Λ_{3SL}) grow apart in some state variables. The solution difference is $SLSC - ASC$.

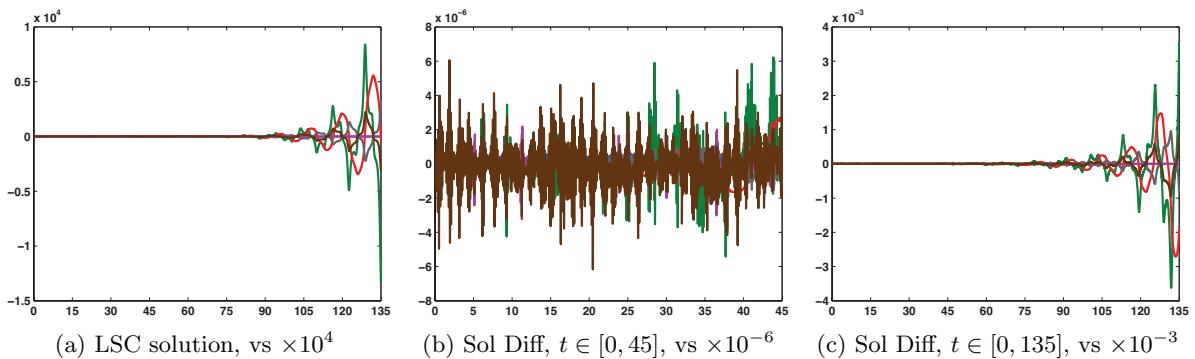


Figure 6.77: The LSC solution (Λ_L) and the SLSC solution (Λ_{SLSC}) grow apart in some state variables. The solution difference is $\text{SLSC} - \text{LSC}$.

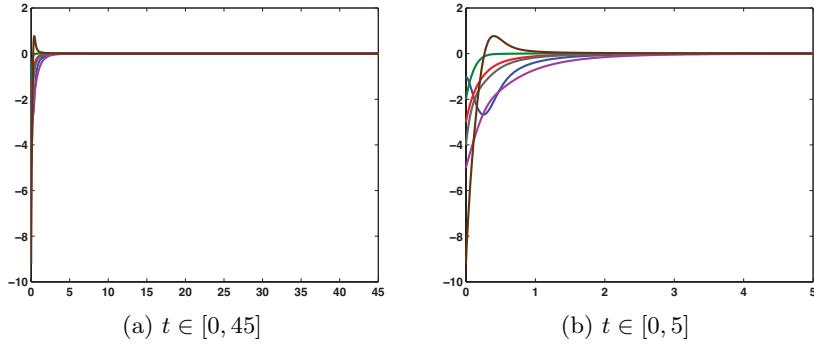


Figure 6.78: SLSC/FO observer's $\tilde{x}(t) - \hat{x}(t)$ on two intervals of time. $e(t) \rightarrow 0$ by $t_f = 45$. ($C_a, \Lambda_{3SL}, \delta = 0.1, \eta = 100, \xi = 0.001$)

Case 1, C_a

From the results summarized in Table 6.8, the stabilized least squares completion with output matrix C_a is observable for $t \in (0, 135]$. A reasonable starting guess for the gain matrix parameters δ , η , and ξ are the values that resulted in observers with desirable convergence properties for Case 1 of the example with positive resistors. For Case 1 of this example with negative resistors, the solver's integration tolerance error occurs at $t = 22.677650$ when $\delta = 0.1$, $\eta = 100$, and $\xi = 0.1$ for the SLSC/FO observer. At this time, the largest eigenvalue of $S_L(t)$ equals 7.412632×10^{12} and the step size cannot be made any smaller than 5.684342×10^{-14} . Decreasing ξ to 0.01 postpones the error until $t = 34.879051$, and the observer construction code ends at $t = 47.437914$ when $\xi = 0.001$. The largest eigenvalue of $S_L(47.437914)$ has become 5.845897×10^{11} after equaling 1.826645×10^2 at $t_f = 45$.

Figure 6.78 presents the estimation error on two intervals for the SLSC/FO observer when $\delta = 0.1$, $\eta = 100$, and $\xi = 0.001$. On the $t \in [0, 45]$ interval, the observer appears to converge to the completion by the desired final time, which is confirmed with $e(45) = (1.0 \times 10^{-6}) * \begin{bmatrix} 0.0 & 0.036767 & -0.347390 & -0.006448 & 0.0 & 0.006448 \end{bmatrix}^T$, and the $t \in [0, 5]$ interval shows the estimation error monotonically converges to zero in four of the six state variables. Pulling from generalizations formed during plot analysis for the example with positive resistors, this observer's convergence properties are indicative of an η that is larger than necessary for $e(45) \in (-10^{-3}, 10^{-3})$ to occur. Figures 6.79b and 6.79c display the estimation errors for the SLSC/FO observers when $\eta = 10$ and $\eta = 1$, respectively. These estimation errors reveal the observer has a harder time estimating the state as η is decreased, but at $t_f = 45$, the difference $\tilde{x}(t) - \hat{x}(t)$ equals $(1.0 \times 10^{-6}) * \begin{bmatrix} 0.0 & -0.037168 & 0.351831 & 0.006513 & 0.0 & -0.006513 \end{bmatrix}^T$ when $\eta = 10$ and $(1.0 \times 10^{-6}) * \begin{bmatrix} 0.0 & 0.011092 & -0.110917 & -0.001915 & 0.0 & 0.001915 \end{bmatrix}^T$ when $\eta = 1$. Although the observer takes fewer units of time to converge to the completion when $\eta = 100$

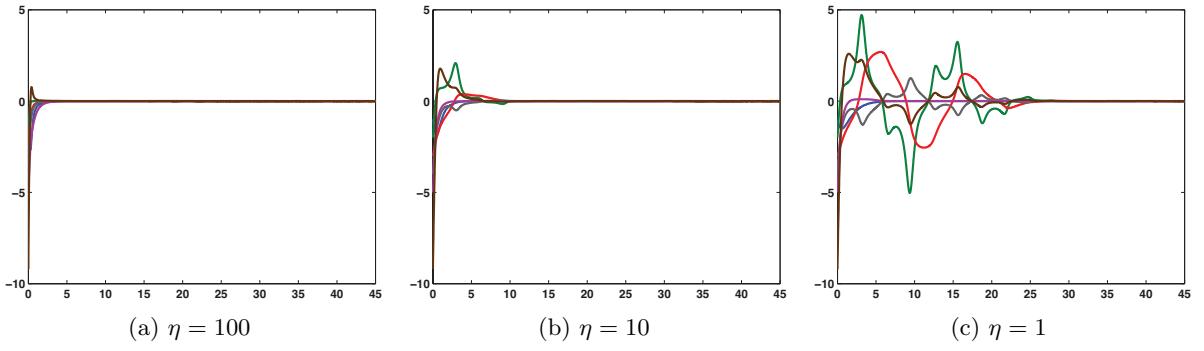


Figure 6.79: SLSC/FO observers' $\tilde{x}(t) - \hat{x}(t)$ as η is varied. $e(t) \rightarrow 0$ by $t_f = 45$ even as η is decreased. (C_a , Λ_{3SL} , $\delta = 0.1$, $\xi = 0.001$)

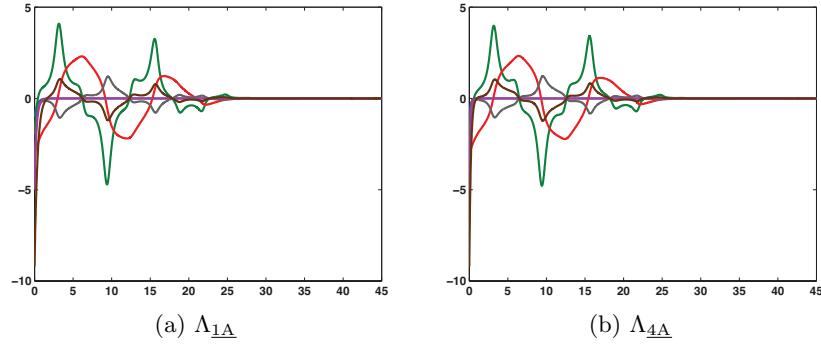


Figure 6.80: ASC/FO observers' $\tilde{x}(t) - \hat{x}(t)$ for Λ_{1A} and Λ_{4A} . These estimation errors appear to be the same. (C_a , $\delta = 0.1$, $\eta = 1$, $\xi = 0.001$)

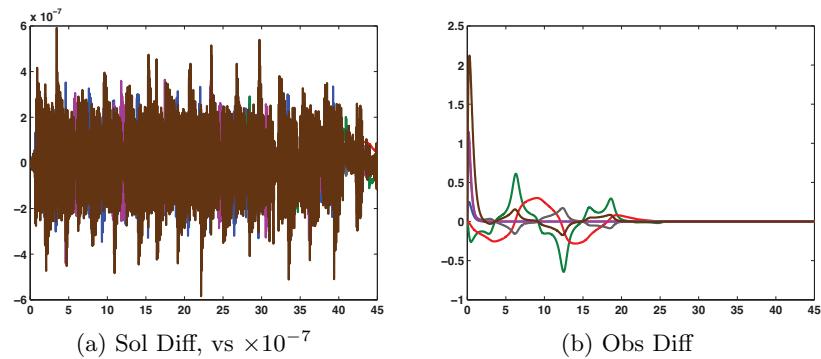


Figure 6.81: Solution and observer comparisons. The ASC solution and ASC/FO observer differences are $\Lambda_{1A} - \Lambda_{4A}$. (C_a , $\delta = 0.1$, $\eta = 1$, $\xi = 0.001$)

than when $\eta = 1$, the run time of the observer construction code (outputs are user defined inputs) is greater: approximately 24.775 seconds (MATLAB's tic toc value) when $\eta = 1$ and almost 6 minutes (tic toc = 358.986 seconds) when $\eta = 100$. This contrast in execution times highlights potential concerns with algorithm efficiency.

The four criteria applied to selecting a stabilization parameter matrix identified two possible contenders for Λ_A . Their resulting alternative stabilized completions have a difference ($\Lambda_{1A} - \Lambda_{4A}$) in Figure 6.81a equal to zero except for some numerical error. Both Λ_{1A} and Λ_{4A} produce observable alternative stabilized completions with output matrix C_a on $t \in (0, 135]$. Figure 6.80 displays the estimation errors for the ASC/FO observers when $\delta = 0.1$, $\eta = 1$, and $\xi = 0.001$, showing no visible discrepancies in the observers' estimates of the state. At $t_f = 45$, $e(t)$ equals $(1.0 \times 10^{-12}) * [0.0 \ -0.036415 \ 0.313527 \ 0.006495 \ 0.0 \ -0.006217]^T$ when stabilizing with Λ_{1A} and MATLAB returns the estimation error $[0.0 \ 0.0 \ 0.0 \ 0.0 \ 0.0 \ 0.0]^T$ when stabilizing with Λ_{4A} . The difference between these ASC/FO observers ($\Lambda_{1A} - \Lambda_{4A}$) is plotted in Figure 6.81b.

Although the norm of $\tilde{A}(t)$ stabilized with Λ_{1A} has an upper bound less than the lower bound of $\|\tilde{A}(t)\|$ for $\tilde{A}(t)$ stabilized with Λ_{4A} , the largest eigenvalues of $S_L(45)$ equal 1.611898×10^2 for Λ_{1A} and 1.659034×10^2 for Λ_{4A} . Thus, the growth of these ASC/FO observers' $S_L(t)$ matrices is comparable. It is difficult to study the effects from the magnitudes of the stabilization parameter matrices' entries on the observer construction code's execution time when $\eta = 1$, but MATLAB's tic toc command returns 13.740 seconds and 16.714 seconds for Λ_{1A} and Λ_{4A} , respectively, when the only outputs are the user defined inputs. A larger η may reveal a less efficient algorithm if stabilizing with Λ_{4A} , but our interest in a stabilization parameter matrix is to provide the best possible circumstances for observer construction. Therefore, Λ_{4A} is our choice for stabilizing the alternative stabilized completion due to its Criterion 3 result (four eigenvalues of $\tilde{A}(t)$ with negative real parts).

The stabilized least squares completion and the alternative stabilized completion are transformed to construct the reduced-order observer using transformation matrix $R = \begin{bmatrix} C_a \\ C_R \end{bmatrix}$, where C_R again equals $\begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$ for output matrix C_a . The reduced-order systems of both stabilized completions with output matrix C_a are observable on $t \in (0, 135]$; that is, the observability Gramians of pairs $(A_{R_{22}}(t), A_{R_{12}}(t))_{\underline{SL}}$ and $(A_{R_{22}}(t), A_{R_{12}}(t))_{\underline{A}}$ have rank 2 by $t = 0.01$. When the gain matrix parameters are $\delta = 0.1$, $\eta = 1$, and $\xi = 0.001$, the SLSC/RO observer converges to the completion by the desired final time. At $t_f = 45$, the estimation error equals $(1.0 \times 10^{-6}) * [0.270466 \ 0.0 \ 0.0 \ 0.0 \ -0.094863 \ 0.0]^T$. Due to the structure of output matrix C_a , only two of the state variables have a nonzero estimation error for the

interval of time under consideration. Figure 6.82a shows how well this reduced-order observer estimates state variables $e_1(t)$ and $i_{r_2}(t)$. The code constructing this observer terminates at $t = 48.805029$ due to the solver's integration tolerance error. The reduced-order system is not exponentially asymptotically stable on the extended time interval, demonstrated by the plot in Figure 6.82b of $\|\Phi_R(t, 0)\|$, where $\Phi_R(t, 0)$ is the state transition matrix of the reduced-order system. This growth translates into an eigenvalue of $S_L(t)$ equaling 3.704447×10^{12} by the time the error occurs.

The ASC/RO observer with output matrix C_a also converges to the completion by $t_f = 45$ when using the same gain matrix parameters as the SLSC/RO observer. For $\delta = 0.1$, $\eta = 1$, and $\xi = 0.001$, Figure 6.83a compared with Figure 6.82a shows the ASC/RO observer estimates the state faster than the SLSC/RO observer but the estimation errors of the $e_1(t)$ and $i_{r_2}(t)$ components converge to zero monotonically for both. The difference at $t_f = 45$ in Figure 6.83a is $\begin{bmatrix} 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \end{bmatrix}^T$. Unlike the SLSC/RO observer, this ASC/RO observer can be constructed to estimate the state on the entire extended time interval, as seen in Figure 6.83b. The norm of $\Phi_R(t, 0)$ in Figure 6.83c goes to zero for the alternative stabilized completion's reduced-order system. The eigenvalues of $S_L(135)$ returned by MATLAB are both 0.0, contradictory to the theoretical statement by O'Reilly in [135] that $S_L(t)^{-1} = P_L(t)$ must be real symmetric, positive definite. Further investigation is required to justify a formal modification to the conditions on $S_L(t)$, but a relaxation to real symmetric, positive semi-definite seems appropriate. This relaxation has already been suggested in [1], [20] for detectable systems.

The first two rows of output matrix C_a are linearly independent from the rows of (6.14) for all time and are defined as C_x to build a nonsingular extended output matrix. Since enough information is provided about the state through the output equation and the characterization of the solution manifold, an observer does not need to be constructed to estimate the state. Instead, using equation (6.13), the state can be determined explicitly. Also, when physical data is available for output $y(t)$, finding a completion of a system of DAEs to compute $\tilde{y}(t)$ becomes unnecessary. For the two stabilized completions, Figure 6.84 shows only noise is the difference between $\tilde{x}(t)$ and $x(t)$, where $x(t)$ is the solution to equation (6.13).

Case 2, C_b

For numerical algorithms, reducing the order of the observer may improve computational efficiency since state variables already known through the output are no longer being estimated. Results in this chapter and in Chapter 5 have also shown a linear ODE system's reduced-order observer can have better convergence properties and rates than its full-order observer. From Criterion 1's results, the completions of the circuit example with negative resistors are not

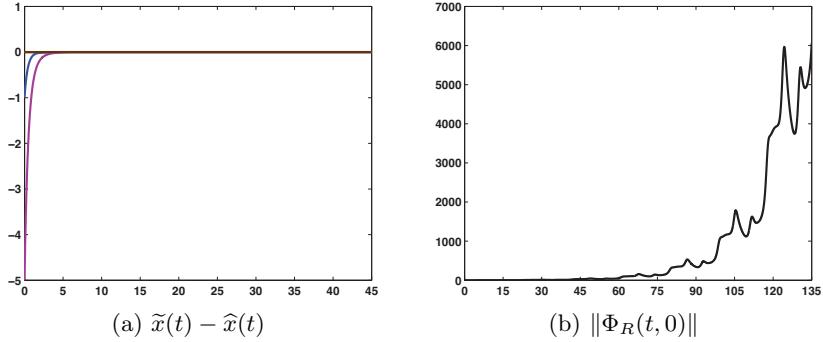


Figure 6.82: SLSC/RO observer's $\tilde{x}(t) - \hat{x}(t)$; $e(t) \rightarrow 0$ by $t_f = 45$ but $\|\Phi_R(t, 0)\|$ does not go to zero by $t = 135$. (C_a , $\Lambda_{3\text{SL}}$, $\delta = 0.1$, $\eta = 1$, $\xi = 0.001$)

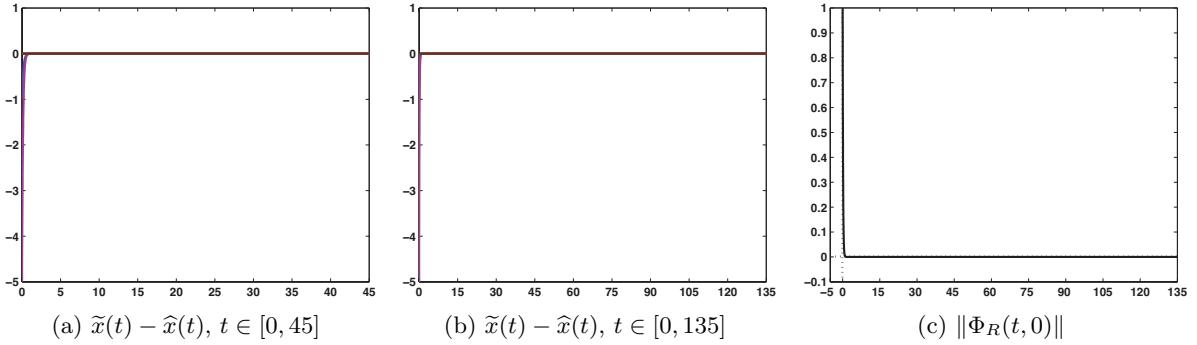


Figure 6.83: ASC/RO observer's $\tilde{x}(t) - \hat{x}(t)$ on two intervals of time; $\|\Phi_R(t, 0)\| \rightarrow 0$ by $t = 135$. ($C_a, \Lambda_{4A}, \delta = 0.1, \eta = 1, \xi = 0.001$)

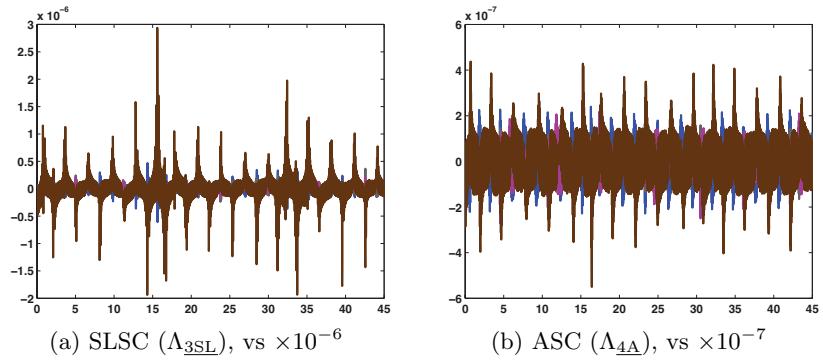


Figure 6.84: $\tilde{x}(t) - x(t)$; equation (6.13) computes $x(t)$ without observer. (C_a)

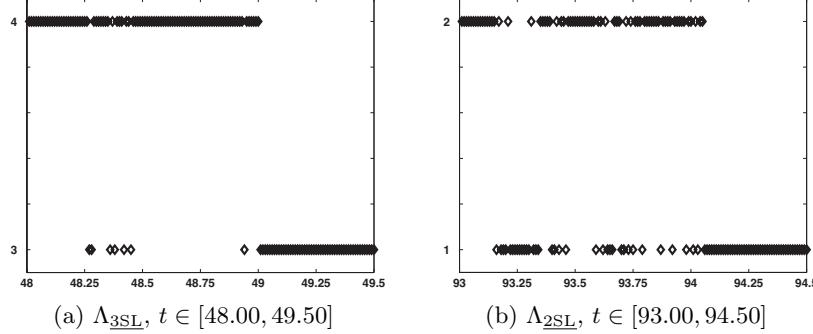


Figure 6.85: SLSC reduced-order systems' rank ($W_o(t)$) examination for $\Lambda_{\underline{3SL}}$ and $\Lambda_{\underline{2SL}}$. (C_b)

exponentially asymptotically stable. However, this case and the others examining the example with negative resistors reveal the stability of the completions' reduced-order systems is also dependent on the output equation.

Unlike the other cases investigated in Chapter 6, this case focuses on the reduced-order observer first. Due to time considerations, the results of the observers with output matrix C_b were computed on VCL computers; for example, it took a VCL computer 208.805 seconds to run code it took 857.854 seconds for the personal computer to run (the VCL computer tic toc time is listed again with the observer that was being constructed). Recall, transformation matrix $R = [\vec{e}_3 \quad \vec{e}_6 \quad \vec{e}_1 \quad \vec{e}_4 \quad \vec{e}_5 \quad \vec{e}_2]$, where \vec{e}_i is a unit vector with 1 in the i th row, is returned by the VCL computer's MATLAB for output matrix C_b .

The code ends prematurely at $t = 15.346035$ when attempting to construct the reduced-order observer using the stabilized least squares completion with stabilization parameter matrix $\Lambda_{\underline{3SL}}$ (SLSC3) and gain matrix parameters $\delta = 0.1$, $\eta = 1$, and $\xi = 0.001$. Using these same gain matrix parameters, the solver's integration tolerance error does not occur until $t = 21.929462$ when attempting to construct the reduced-order observer using the stabilized least squares completion with stabilization parameter matrix $\Lambda_{\underline{2SL}}$ (SLSC2). The observability Gramian's rank for pair $(A_{R_{22}}(t), A_{R_{12}}(t))_{\underline{3SL}}$ is 4 from 0.01 to 48.27 but then fluctuates between 4 and 3 on the interval shown in Figure 6.85a. The rank remains at 3 from 49.01 to 54.92 before dropping to 2 at $t = 54.93$ and to 1 at $t = 73.73$. For one value of time, rank($W_o(74.28)$) increases from 1 to 2. A rank drop also occurs when implementing the observability check for pair $(A_{R_{22}}(t), A_{R_{12}}(t))_{\underline{2SL}}$. Although rank($W_o(t)$) = 4 from 0.01 to 55.71, it drops to 3 at $t = 55.72$, to 2 at $t = 62.68$, and to 1 at $t = 94.06$. The rank fluctuation between 2 and 1 on the $t \in [93.00, 94.50]$ interval is pictured in Figure 6.85b.

Figure 6.86 plots $\|\Phi_R(t, 0)\|$ on the extended time interval $t \in [0, 135]$ for the reduced-order systems of these two stabilized least squares completions. Noticing these figures' vertical scales, the growth is greater than what was seen earlier in either linear time-varying example's

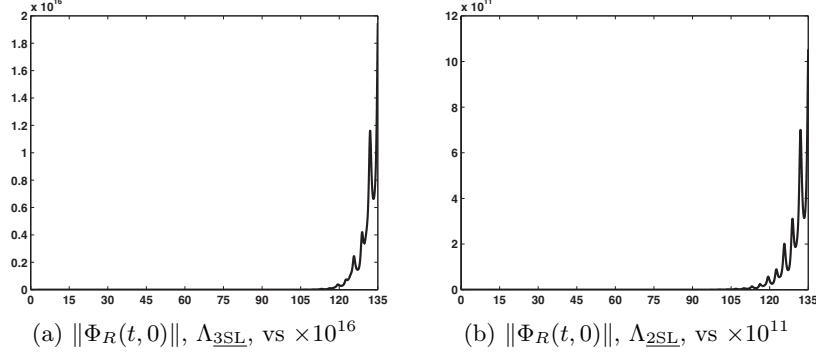


Figure 6.86: SLSC reduced-order systems' $\|\Phi_R(t, 0)\|$ for $\Lambda_{\underline{3SL}}$ and $\Lambda_{\underline{2SL}}$. $\|\Phi_R(t, 0)\|$ fails to go to zero by $t = 135$ for both Λ_{SL} options. (C_b)

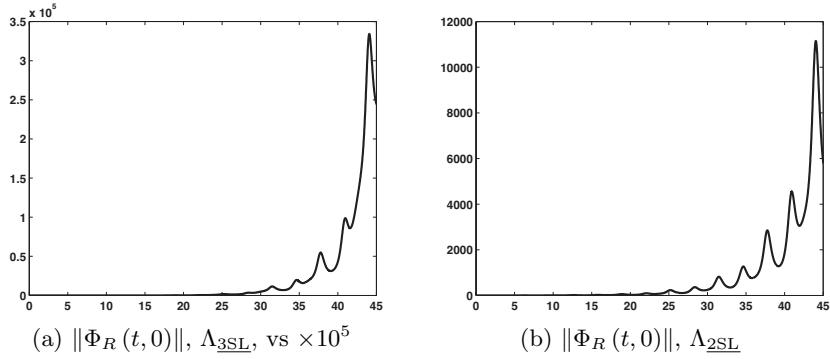


Figure 6.87: SLSC reduced-order systems' $\|\Phi_R(t, 0)\|$ for $\Lambda_{\underline{3SL}}$ and $\Lambda_{\underline{2SL}}$. On $t \in [0, 45]$, $\|\Phi_R(t, 0)\|$ does not grow as quickly for $\Lambda_{\underline{2SL}}$. (C_b)

Criterion 1 results, which suggests a greater challenge of selecting an appropriate combination of gain matrix parameters. Even by desired final time $t_f = 45$, Figure 6.87 shows the norm's maximum for SLSC₃ is more than ten times greater than the norm's maximum for SLSC₂. Thus, one way to slow the growth of $S_L(t)$ for this case's SLSC/RO observer is to stabilize with $\Lambda_{\underline{2SL}}$ rather than $\Lambda_{\underline{3SL}}$.

Computational efficiency is also taken into consideration when deciding on which gain matrix parameters to test. If $\delta = 0.1$, $\eta = 1$, and $\xi = 0.001$, ending the code at $t = 20$ before $t = 21.929462$ produces the estimation error in Figure 6.88a for this SLSC₂/RO observer (identified by $\xi = 0.001$). The observer construction code terminates prematurely at $t = 21.929462$ as well if $\delta = 0.001$, $\eta = 100$, and $\xi = 0.1$ are chosen to construct the SLSC₂/RO observer (identified by $\delta = 0.001$). The estimation error for this observer on the $t \in [0, 20]$ interval is in Figure 6.88b. Except for some numerical error, the difference between these SLSC₂/RO observers

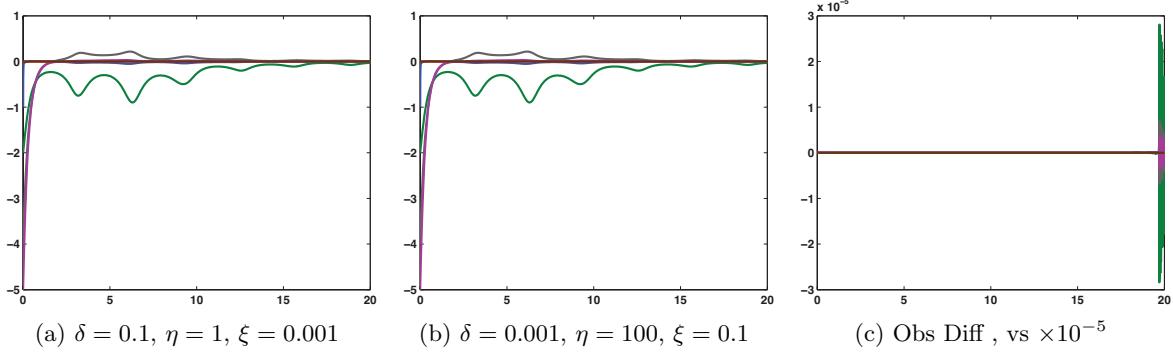


Figure 6.88: SLSC₂/RO observers' $\tilde{x}(t) - \hat{x}(t)$ as δ , η , and ξ are varied; the SLSC₂/RO observer difference is $(\xi = 0.001) - (\delta = 0.001)$. Integration tolerance error occurs after $t = 20$. (C_b)

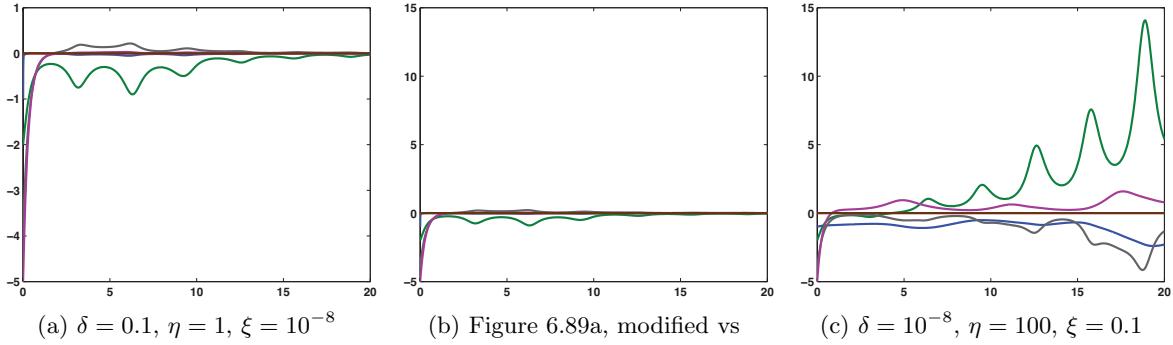


Figure 6.89: SLSC₂/RO observers' $\tilde{x}(t) - \hat{x}(t)$ as δ , η , and ξ are varied. With the modifications to the gain matrix parameters, the SLSC₂/RO observers are no longer the same. (C_b)

$((\xi = 0.001) - (\delta = 0.001))$ is zero. MATLAB's run time for constructing these reduced-order observers is approximately 8.5 minutes for both sets of gain matrix parameters (tic toc = 514.302 seconds, $\xi = 0.001$; tic toc = 516.110 seconds, $\delta = 0.001$).

Based on the LSC/FO observer analysis from Case 1 in the example with positive resistors, decreasing ξ or δ delays the effects from $S_L(t)$'s growth on the observer construction code with varying consequences. On the $t \in [0, 20]$ interval, Figures 6.88a and 6.89a show the SLSC₂/RO observer's estimate of $\tilde{x}(t)$ does not change when ξ is decreased from 0.001 to 10^{-8} (keeping $\delta = 0.1$ and $\eta = 1$; $\xi = 10^{-8}$ identifies observer). However, decreasing δ without modifying η affects the observer's convergence properties. Figure 6.89c displays $e(t)$ on $t \in [0, 20]$ for the SLSC₂/RO observer when δ is decreased from 0.001 to 10^{-8} (keeping $\eta = 100$ and $\xi = 0.1$; $\delta = 10^{-8}$ identifies observer). Figure 6.89b includes the results from Figure 6.89a on a modified vertical scale for comparison with Figure 6.89c. The benefit of decreasing δ instead of ξ , though, is a shorter run time. The observer construction code for $t \in [0, 20]$ finishes executing in about

6 minutes (tic toc = 368.640 seconds) for the $\xi = 10^{-8}$ SLSC₂/RO observer and in only 14.485 seconds for the $\delta = 10^{-8}$ SLSC₂/RO observer. The largest eigenvalues of $S_L(20)$ are 82.551560 and 0.825516×10^{-5} for these reduced-order observers with $\xi = 10^{-8}$ and $\delta = 10^{-8}$, respectively. The solver has a finer time step as $S_L(t)$ grows, reducing computational efficiency and forcing a longer run time. If our primary interest is in whether or not the observer converges by $t_f = 45$ rather than in how well the observer estimates the state, our choice for this case would be to decrease δ in order to delay the solver's integration tolerance error. (Note, a user-initiated end stopped the code from constructing the $\xi = 10^{-8}$ SLSC₂/RO observer on the $t \in [0, 45]$ interval after it had been running for almost 9.5 out of the 10 hours available for a VCL session. It took just 3.5 minutes (tic toc = 208.805 seconds) for the code to construct the $\delta = 10^{-8}$ SLSC₂/RO observer on the same interval of time.)

With $\eta = 100$ and $\xi = 0.1$, decreasing δ to 10^{-8} delays the solver's integration tolerance error until $t = 49.910501$. This SLSC₂/RO observer appears to converge to $\tilde{x}(t)$ by $t_f = 45$ in Figure 6.90a, but $e(45) = (1.0 \times 10^{-2}) * \begin{bmatrix} -0.005593 & -0.133569 & 0.0 & 0.022445 & 0.001962 & 0.0 \end{bmatrix}^T$. Making η equal 1000 and then 10000 improves the observer's estimate of the state but does not decrease the difference $\tilde{x}(45) - \hat{x}(45)$ enough to be within $\pm 10^{-3}$ of zero:

$$\begin{aligned}\eta = 1000 : \quad & (1.0 \times 10^{-2}) * \begin{bmatrix} -0.005513 & -0.129832 & 0.0 & 0.021803 & 0.001934 & 0.0 \end{bmatrix}^T, \\ \eta = 10000 : \quad & (1.0 \times 10^{-2}) * \begin{bmatrix} -0.005438 & -0.103403 & 0.0 & 0.017182 & 0.001907 & 0.0 \end{bmatrix}^T.\end{aligned}$$

MATLAB's run time with the user defined inputs as the only outputs goes from nearly 20 minutes (tic toc = 1068.234 seconds) to almost 2 hours (tic toc = 6671.046 seconds) when η is increased from 1000 to 10000. Rather than continuing to trade computational efficiency for the desired $e(45) \in (-10^{-3}, 10^{-3})$, an alternative solution is to implement a moving horizon. Using $\delta = 10^{-8}$, $\eta = 100$, and $\xi = 0.1$, the value of $S_L(10)$ is reset to $\delta = 10^{-8}$, which delays the solver's integration tolerance error until $t = 58.766716$. The resulting estimation error for this SLSC₂/RO observer appears in Figure 6.91b. At $t_f = 45$, $e(t) = (1.0 \times 10^{-3}) * \begin{bmatrix} 0.011409 & 0.281319 & 0.0 & -0.047338 & -0.004002 & 0.0 \end{bmatrix}^T$ is within the desired interval and the code's run time is only 36.557 seconds. Figure 6.91a includes the results from Figure 6.90a on a modified vertical scale for comparison with Figure 6.91b, and Figure 6.91c plots the estimation error for this moving horizon observer on the $t \in [0, 58]$ interval. Selecting this particular SLSC₂/RO observer with output matrix C_b sacrifices how well the observer estimates the state on the entire $t \in [0, 45]$ interval for computational efficiency.

The last suggestion for constructing a reduced-order observer with output matrix C_b is to use the alternative stabilized completion. Figure 6.92a plots $\|\Phi_R(t, 0)\|$ on the extended time interval for the reduced-order system represented by pair $(A_{R_{22}}(t), A_{R_{12}}(t))_A$. This figure's

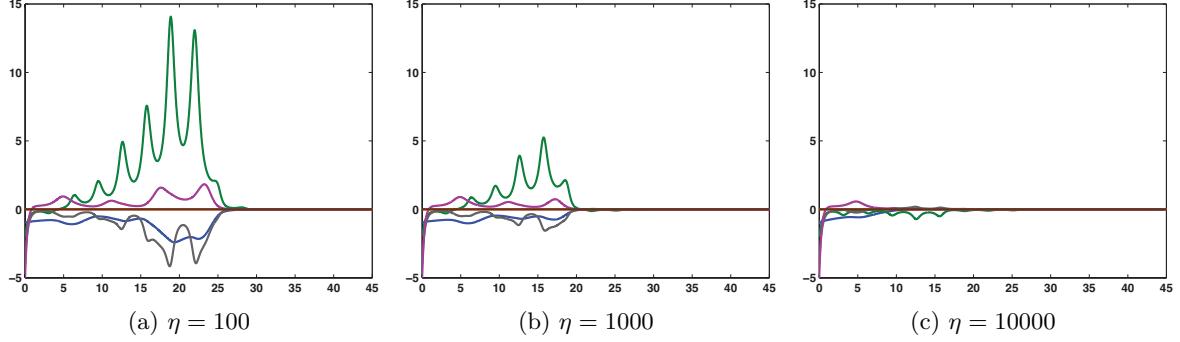


Figure 6.90: SLSC2/RO observers' $\tilde{x}(t) - \hat{x}(t)$ as η is varied. Increasing η improves observer convergence, but $e(45)$ is not within $(-10^{-3}, 10^{-3})$. ($C_b, \delta = 10^{-8}, \xi = 0.1$)

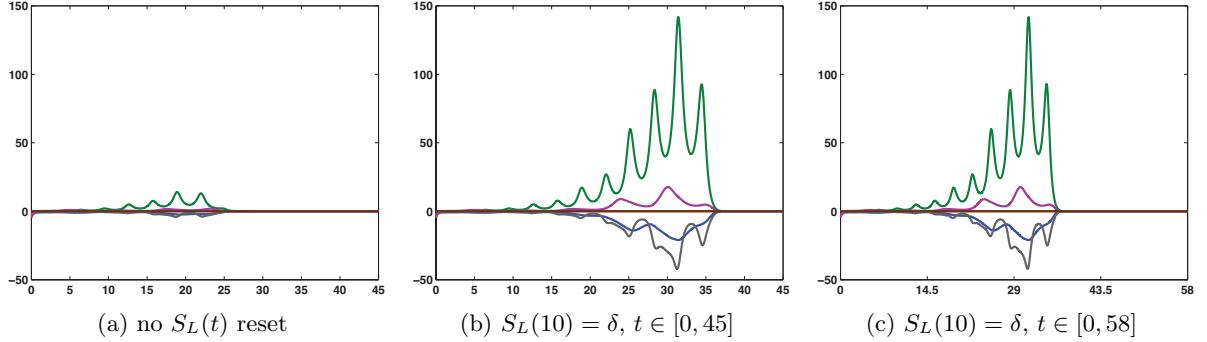


Figure 6.91: SLSC2/RO observers' $\tilde{x}(t) - \hat{x}(t)$ without and with moving horizon. $e(45)$ is within desired interval for moving horizon observer. ($C_b, \delta = 10^{-8}, \eta = 100, \xi = 0.1$)

vertical scale has a maximum less than those seen in Figures 6.86a and 6.86b ($\times 10^9$ as opposed to $\times 10^{16}$ or $\times 10^{11}$), indicating slower growth. The rank of this pair's observability Gramian is 4 and then drops to 3 at 79.34, to 2 at 85.85, and to 1 at 106.80. One fluctuation occurs when rank($W_o(79.64)$) equals 4 instead of 3. If either δ or ξ is decreased to 10^{-6} (rather than to 10^{-8} for the SLSC2/RO observer), then the solver's integration tolerance error is delayed until after $t_f = 45$. Considering $\xi = 10^{-6}$ with $\delta = 0.1$ and $\eta = 1$, the observer construction code finishes running in just over 9.25 minutes (tic toc = 558.926 seconds) and produces the ASC/RO observer with the estimation error presented in Figure 6.92b. At $t_f = 45$, the difference $\tilde{x}(t) - \hat{x}(t)$ is $(1.0 \times 10^{-5}) * [0.0 \ 0.645772 \ 0.0 \ -0.113264 \ 0.0 \ 0.0]^T$, so numerical and visual estimation error results support selecting this reduced-order observer for its convergence properties and rate and for the code's computational efficiency.

The example mentioned in the introduction of Section 6.2 that compelled the relative error tolerance (RelTol) of MATLAB's `ode45` to be set at `1e-9` instead of its default `1e-3` is the

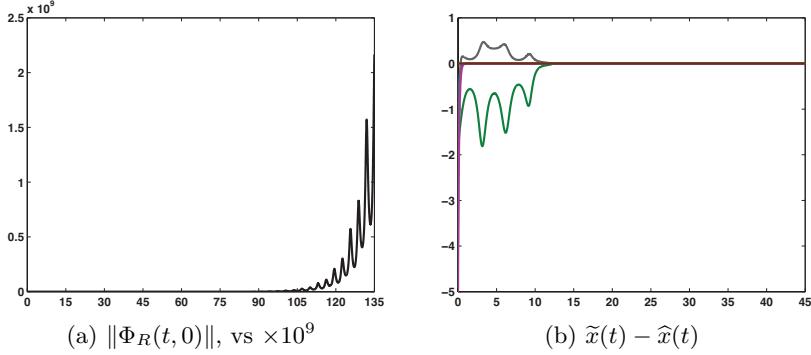


Figure 6.92: ASC reduced-order system's $\|\Phi_R(t, 0)\|$ does not grow as quickly as results in Figure 6.86; ASC/RO observer's $\tilde{x}(t) - \hat{x}(t)$. This observer's convergence properties and rate are preferable to those for the SLSC2/RO observer's. ($C_b, \Lambda_{4A}, \delta = 0.1, \eta = 1, \xi = 10^{-6}$)

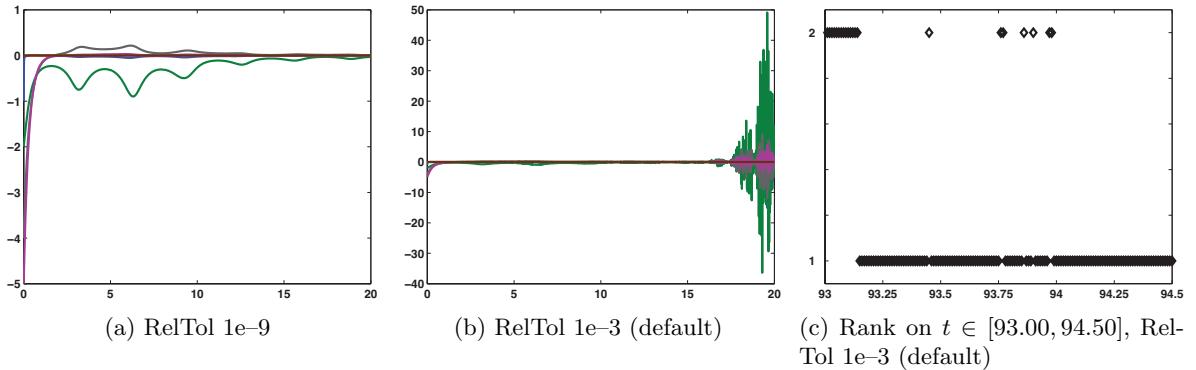


Figure 6.93: SLSC2/RO observers' $\tilde{x}(t) - \hat{x}(t)$ as RelTol is varied. The rank of $W_o(t)$ is affected by RelTol value. ($C_b, \delta = 0.001, \eta = 100, \xi = 0.1$)

construction of the $\delta = 0.001$ SLSC2/RO observer. Although Figure 6.93a shows the estimation error tends toward zero by $t = 20$ when RelTol equals $1e-9$, Figure 6.93b reveals the estimation error grows when RelTol equals its default. If RelTol is left at its default value, the solver's solution does not need to be as exact, which for this research, affects the observer's convergence. The value of RelTol also influences the observability Gramian's rank calculation. The rank discrepancies occur on the $t \in [93.00, 94.50]$ interval for pair $(A_{R22}(t), A_{R12}(t))_{2SL}$, and Figure 6.93c displays rank($W_o(t)$) when RelTol equals its default. Although the fluctuation between ranks 2 and 1 is reduced compared with the results in Figure 6.85b for the modified RelTol, a less sensitive RelTol could hide rank($W_o(t)$) concerns.

The construction of the full-order observers using the stabilized completions is straightforward compared with the process described for constructing the reduced-order observers. The

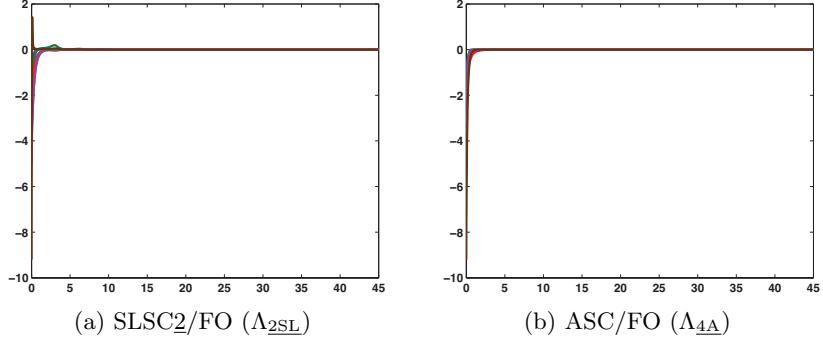


Figure 6.94: Full-order observers' $\tilde{x}(t) - \hat{x}(t)$. ($C_b, \delta = 0.1, \eta = 100, \xi = 0.001$)

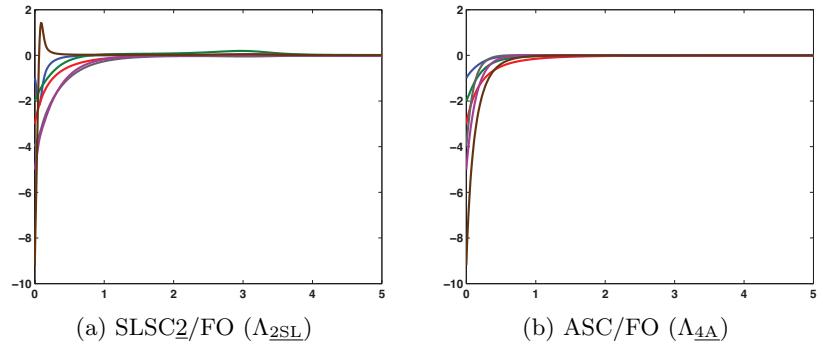


Figure 6.95: Full-order observers' $\tilde{x}(t) - \hat{x}(t)$ (results from Figure 6.94 on $t \in [0, 5]$). Shortened interval shows these completions' full-order observers have better convergence properties and rates than their respective reduced-order observers. ($C_b, \delta = 0.1, \eta = 100, \xi = 0.001$)

observer construction code terminates at $t = 47.543955$ for the SLSC2/FO observer and at $t = 47.536731$ for the ASC/FO observer when gain matrix parameters $\delta = 0.1, \eta = 100$, and $\xi = 0.001$ are selected, but these observers' estimation errors are within the desired interval by $t_f = 45$. Figure 6.94 displays the estimation errors, and although the difference $\tilde{x}(45) - \hat{x}(45)$ is less for the SLSC2/FO observer,

$$\begin{aligned} \text{SLSC2/FO : } & (1.0 \times 10^{-7}) * \begin{bmatrix} 0.0 & -0.040959 & 0.379079 & 0.007180 & 0.0 & -0.007180 \end{bmatrix}^T, \\ \text{ASC/FO : } & (1.0 \times 10^{-6}) * \begin{bmatrix} 0.0 & 0.072989 & -0.641353 & -0.012801 & 0.0 & 0.012801 \end{bmatrix}^T, \end{aligned}$$

the ASC/FO observer provides a better estimate of $\tilde{x}(t)$ on the $t \in [0, 5]$ interval (see Figure 6.95). The code took less than a minute to construct each full-order observer, so for this case or any example where $\|\Phi_R(t, 0)\|$ grows faster than $\|\Phi(t, 0)\|$, reducing the order of the observer is not currently promoted.

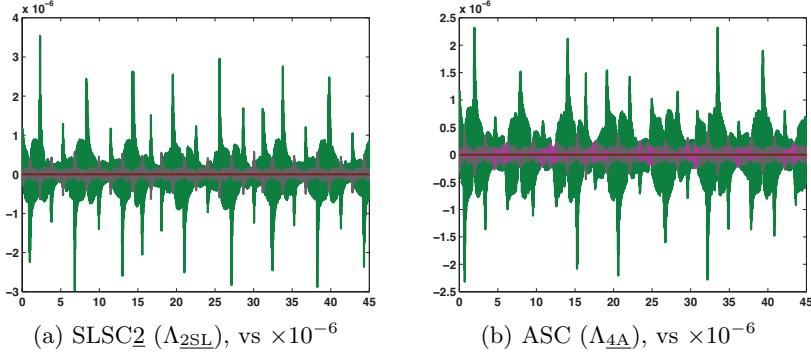


Figure 6.96: $\tilde{x}(t) - x(t)$; equation (6.13) computes $x(t)$ without observer. (C_b)

One desirable characteristic about output matrix C_b is that its rows are linearly independent from the rows of (6.14) for all time. Thus, the constraints characterizing the solution manifold and output equation $\tilde{y}(t) = C_b \tilde{x}(t)$ provide enough information about the state to make constructing an observer unnecessary. Figure 6.96 compares the stabilized completion solutions with the solutions of equation (6.13) by plotting their differences ($\tilde{x}(t) - x(t)$).

Case 3, C_d

Although neither the stabilized least squares completion nor the alternative stabilized completion with output matrix C_d is observable, full-order observers can be constructed to estimate the circuit example's state. After receiving the solver's integration tolerance error before $t_f = 45$ from trying both $\xi = 0.1$ and $\xi = 0.01$ with $\delta = 0.1$ and $\eta = 100$, these values of δ and η successfully combine with $\xi = 0.001$ to delay the error until $t = 47.437914$ for the SLSC/FO observer and $t = 47.536731$ for the ASC/FO observer. The estimation errors for both observers in Figure 6.97 appear to go to zero by the desired final time. The numerical results at $t_f = 45$ confirm this observation: $e(45) = (1.0 \times 10^{-7}) * [0.0 \quad -0.066530 \quad 0.673990 \quad 0.011663 \quad 0.0 \quad -0.011663]^T$ for the SLSC/FO observer and $e(45) = [0.0 \quad 0.0 \quad 0.0 \quad 0.0 \quad 0.0 \quad 0.0]^T$ for the ASC/FO observer. The closer views in Figure 6.98 again expose the polynomial influence from the stabilized least squares completion's Jordan blocks on the SLSC/FO observer's estimate of the state.

The observability Gramians of pairs $(\tilde{A}(t), C_d)_L$ and $(\tilde{A}(t), C_d)_A$ have the same rank: 4 on $t \in (0, 135]$. However, the gain matrix parameters successfully implemented in the construction of the SLSC/FO and ASC/FO observers trigger the solver's integration tolerance error during the construction of the LSC/FO observer on the $t \in [0, 45]$ interval. Once ξ is decreased to 0.0001, the observer construction code finishes executing and produces an LSC/FO observer with the estimation error presented in Figure 6.99a. The behavior of the estimation error implies the least squares completion with output matrix C_d is not detectable, and in Figure 6.99b, the

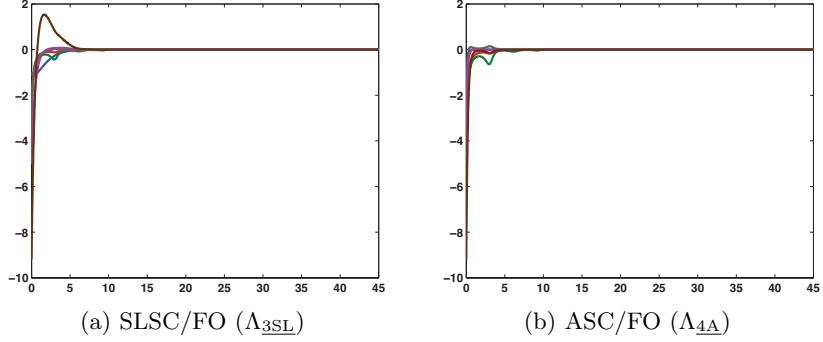


Figure 6.97: Full-order observers' $\tilde{x}(t) - \hat{x}(t)$. $e(t) \rightarrow 0$ by $t_f = 45$. (C_d , $\delta = 0.1$, $\eta = 100$, $\xi = 0.001$)

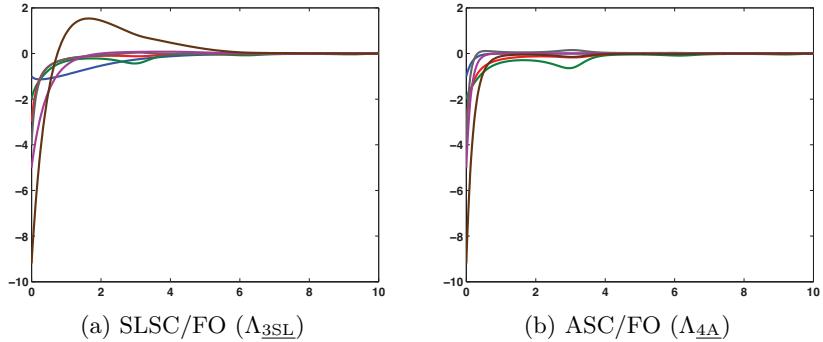


Figure 6.98: Full-order observers' $\tilde{x}(t) - \hat{x}(t)$ (results from Figure 6.97 on $t \in [0, 10]$). Shortened interval provides closer view for estimation error comparison. (C_d , $\delta = 0.1$, $\eta = 100$, $\xi = 0.001$)

graph of $\|\Phi_3(t, 0)\|$, where $\Phi_3(t, 0)$ is the state transition matrix of the unobservable subsystem, does not go to zero on the interval $t \in [0, 45]$. Figure 6.100 includes the graphs of $\|\Phi_3(t, 0)\|$ for the stabilized least squares completion and the alternative stabilized completion, supporting the detectability suspicions for these stabilized completions with output matrix C_d .

Case 3 from [23] also compared the SLSC/FO and ASC/FO observers when considering output matrix C_d (designated as output matrix C_c in the article). In the article, the ASC/FO observer was constructed using the smooth decomposition method (SDM) and stabilization parameter matrix Λ_{2A} , while the stabilized least squares completion was stabilized with Λ_{1SL} . The observability check revealed the stabilized least squares completion was not observable but the alternative stabilized completion was. The rank results summarized in Table 6.11 for the alternative stabilized completion found in terms of symbolically defined matrices (SYM) and stabilized with Λ_{2SL} indicate this completion is not observable. The same discrepancy occurs when stabilizing with Λ_{4SL} .

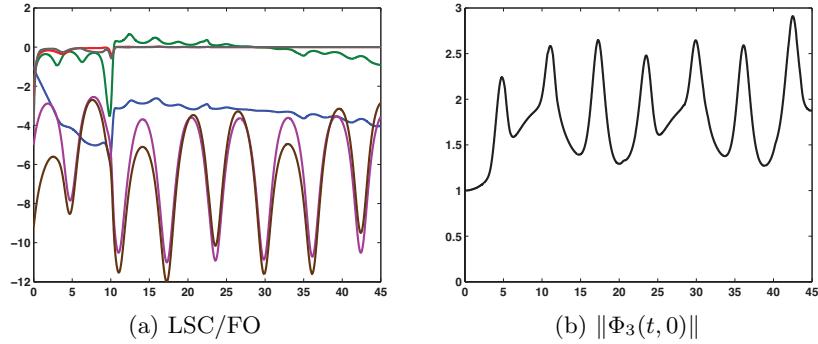


Figure 6.99: LSC/FO observer's $\tilde{x}(t) - \hat{x}(t)$. $\|\Phi_3(t, 0)\|$ fails to converge to zero for the LSC. ($C_d, \Lambda_L, \delta = 0.1, \eta = 100, \xi = 0.001$)

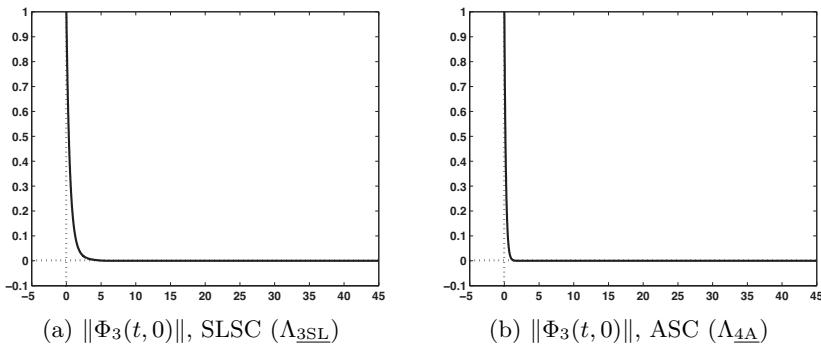


Figure 6.100: $\|\Phi_3(t, 0)\| \rightarrow 0$, so the stabilized completions are detectable. (C_d)

As seen in Table 6.11, the rank of the observability Gramian for SYM pair $(\tilde{A}(t), C_d)_{\underline{A}}$ is 4 from 0.01 to 135.00. On the article's $t \in [0, 45]$ interval, the rank of $W_o(t)$ for the SDM pair reaches 6 at $t = 0.26$ and remains there for the rest of the interval after equaling 4 from 0.01 to 0.08 and 5 from 0.09 to 0.25. However, with this chapter's implementation of the observability check on the extended time interval, a rank drop is uncovered. The rank of the observability Gramian decreases to 5 at $t = 91.64$ and to 4 at $t = 105.47$. The plot of the difference between the two alternative stabilized completions (SDM – SYM) in Figure 6.101a shows the completions are equivalent except for some numerical error. Additionally, their full-order observers with gain matrix parameters $\delta = 0.1$, $\eta = 100$, and $\xi = 0.001$ are comparable. Figure 6.101b and, for a closer view, Figure 6.101c display the ASC/FO difference (SDM – SYM). The greatest dissimilarity between the observers occurs in their estimates of state variable $i_v(t)$, but by $t = 5$ no visible distinction can be made. Although the numerical smooth decomposition method remains an acceptable alternative to the exact symbolic method in estimating the state of this example with negative resistors, the observability check should not return such a rank and therefore requires further investigation.

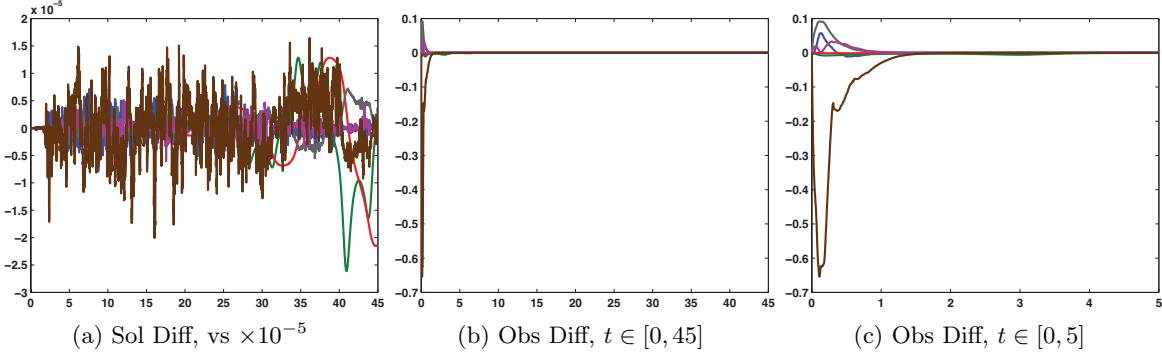


Figure 6.101: Solution and observer comparisons. The ASC solution and ASC/FO observer differences are $SDM - SYM$. ($C_d, \Lambda_{4A}, \delta = 0.1, \eta = 100, \xi = 0.001$)

Transformation matrix $R = \begin{bmatrix} C_d \\ C_R \end{bmatrix}$ is made nonsingular with the following rows listed in order from third to sixth: $\begin{bmatrix} -1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$, $\begin{bmatrix} 0 & -1 & 0 & 0 & 0 & 0 \end{bmatrix}$, $\begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$, and $\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$. The observability check produces $\text{rank}(W_o(t)) = 3$ on $t \in (0, 135]$ for the stabilized least squares completion's reduced-order system and $\text{rank}(W_o(t)) = 2$ on $t \in (0, 135]$ for the alternative stabilized completion's reduced-order system. Gain matrix parameters $\delta = 0.1, \eta = 1$, and $\xi = 0.1$ fail to generate either an SLSC/RO or an ASC/RO observer that converges to the state by the desired final time. Figures 6.102a and 6.102b display the estimation errors, and the ASC/RO observer seems to converge to zero by $t_f = 45$, but at this final time, the difference $\tilde{x}(t) - \hat{x}(t)$ equals $\begin{bmatrix} 0.0 & -0.004813 & 0.0 & 0.0 & 0.0 & -0.001565 \end{bmatrix}^T$. At $t_f = 45$, the estimation error is $\begin{bmatrix} -0.099979 & -0.482158 & 0.0 & 0.0 & 0.035069 & 0.118128 \end{bmatrix}^T$ for the SLSC/RO observer.

After η is increased to 10, the visual and numerical convergence checks, Figure 6.103c and $e(45) = (1.0 \times 10^{-8}) * \begin{bmatrix} 0.0 & 0.318354 & 0.0 & 0.0 & 0.0 & 0.231201 \end{bmatrix}^T$, agree for the ASC/RO observer's estimation error. This larger η also improves the SLSC/RO observer's convergence rate. Figure 6.103a for the SLSC/RO observer when $\delta = 0.1, \eta = 10$, and $\xi = 0.1$ is comparable to Figure 6.102b for the ASC/RO observer when $\delta = 0.1, \eta = 1$, and $\xi = 0.1$, but again, the estimation error $\begin{bmatrix} -0.000875 & -0.004219 & 0.0 & 0.0 & 0.000307 & 0.001034 \end{bmatrix}^T$ is not within the desired interval at $t_f = 45$ for this SLSC/RO observer. One more increase of η by a factor of 10 produces an estimation error of $(1.0 \times 10^{-6}) * \begin{bmatrix} -0.010956 & -0.102967 & 0.0 & 0.0 & 0.004354 & 0.001149 \end{bmatrix}^T$ at $t_f = 45$ and the difference $\tilde{x}(t) - \hat{x}(t)$ in Figure 6.103b for the SLSC/RO observer.

The solver's integration tolerance error does not occur when running the observer construction code on the extended time interval $t \in [0, 135]$ for either the SLSC/RO observer with

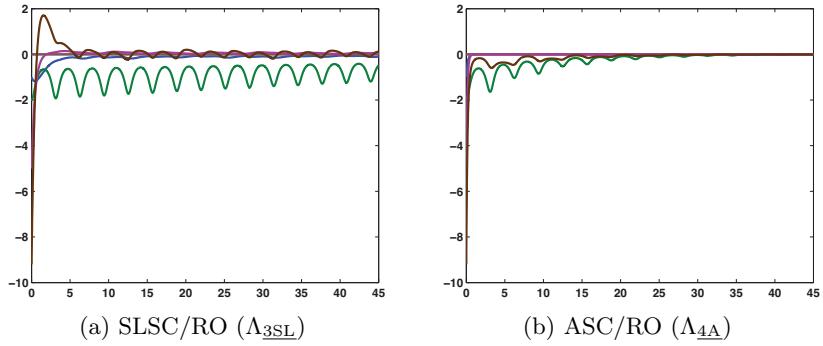


Figure 6.102: Reduced-order observers' $\tilde{x}(t) - \hat{x}(t)$. The observers' estimates of $e_2(t)$ are visibly different. ($C_d, \delta = 0.1, \eta = 1, \xi = 0.1$)

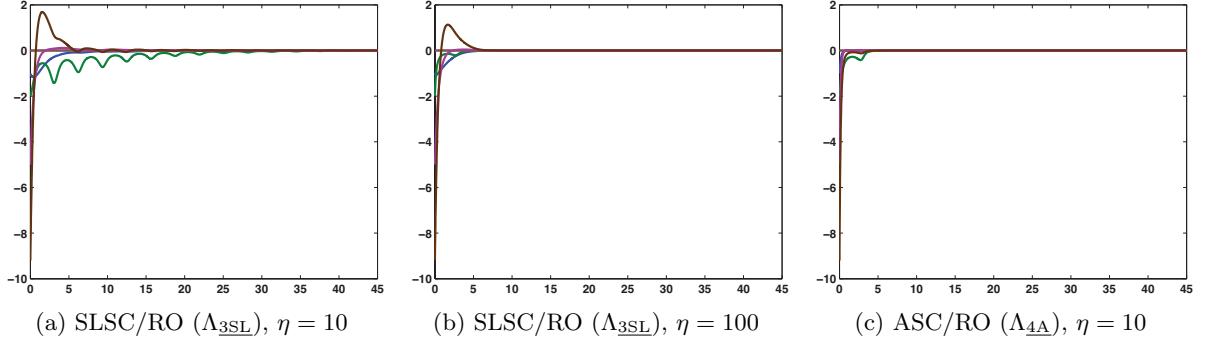


Figure 6.103: Reduced-order observers' $\tilde{x}(t) - \hat{x}(t)$. Observers with results in Figures 6.103b and 6.103c have similar convergence rates. ($C_d, \delta = 0.1, \xi = 0.1$)

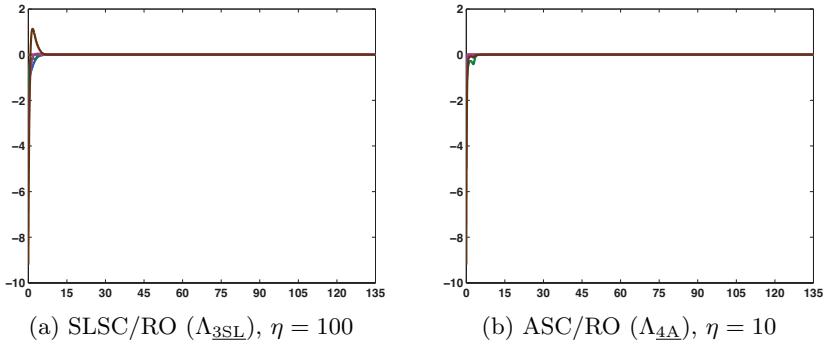


Figure 6.104: Reduced-order observers' $\tilde{x}(t) - \hat{x}(t)$. Results in Figures 6.103b and 6.103c extended to $t = 135$. ($C_d, \delta = 0.1, \xi = 0.1$)

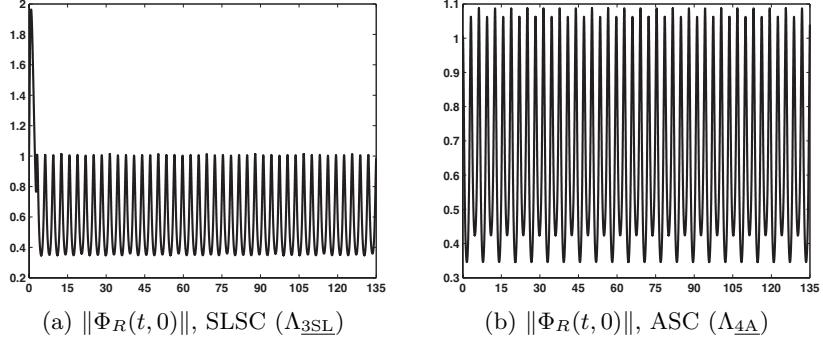


Figure 6.105: $\|\Phi_R(t, 0)\|$ is bounded on $t \in [0, 135]$ for the stabilized completions. (C_d)

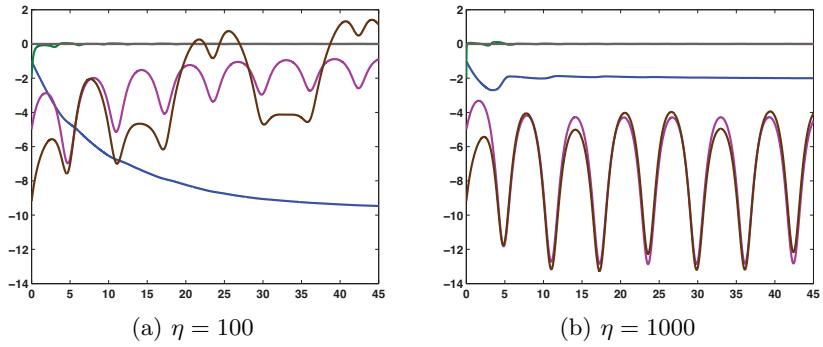


Figure 6.106: LSC/RO observers' $\tilde{x}(t) - \hat{x}(t)$ as η is varied. The reduced-order system is unobservable. (C_d , Λ_L , $\delta = 0.1$, $\xi = 0.1$)

$\delta = 0.1$, $\eta = 100$, and $\xi = 0.1$ or the ASC/RO observer with $\delta = 0.1$, $\eta = 10$, and $\xi = 0.1$. These observers' estimation errors appear in Figure 6.104. The largest eigenvalue of $S_L(t)$ is 0.022534 at $t_f = 45$ and 0.209431 at $t = 135$ for the SLSC/RO observer, while the largest eigenvalue of $S_L(t)$ is 0.023315 at $t_f = 45$ and 0.285158 at $t = 135$ for the ASC/RO observer. The plots of $\|\Phi_R(t, 0)\|$ in Figure 6.105 for the reduced-order systems do not go to zero but are bounded above. This stability is keeping the Riccati equations' $S_L(t)$ from growing large enough to cause computational issues.

Although the least squares completion with output matrix C_d is not detectable, two estimation error plots for LSC/RO observers (Figure 6.106a when $\delta = 0.1$, $\eta = 100$, $\xi = 0.001$; Figure 6.106b when $\delta = 0.1$, $\eta = 1000$, $\xi = 0.001$) are included to reiterate why the stabilized completions are preferred in our observer construction approach.

The rows of output matrix C_d and (6.14) are linearly independent for all time. Having enough information about the state available from the output and the characterization of the solution manifold means $x(t)$ can be solved for explicitly using equation (6.13). Figure 6.107 presents the difference $\tilde{x}(t) - x(t)$ for each of the three completions.

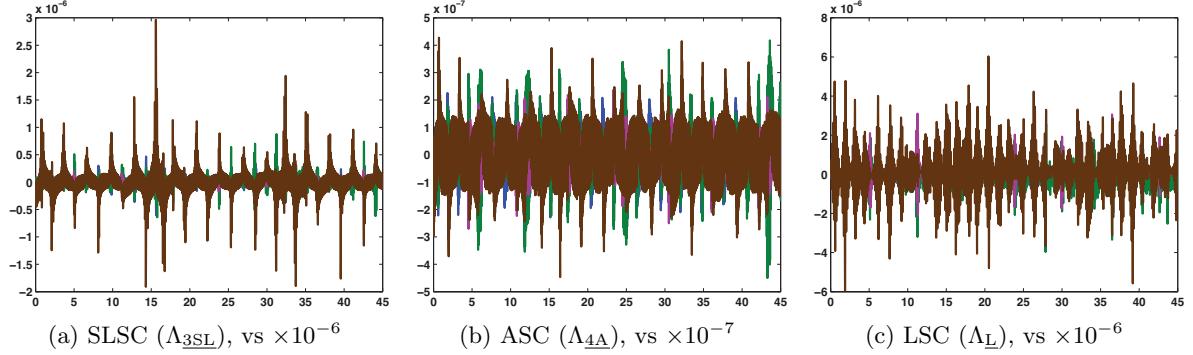


Figure 6.107: $\tilde{x}(t) - x(t)$; equation (6.13) computes $x(t)$ without observer. (C_d)

Case 4, C_f

The results listed for Criterion 4's observability checks reveal neither the stabilized least squares completion (Table 6.9), the alternative stabilized completion (Table 6.12), nor the least squares completion is observable when paired with output matrix C_f . Figure 6.108 displays the norm of the unobservable system's state transition matrix $\Phi_3(t, 0)$ for each of the full-order systems. The norms' convergence to zero in Figures 6.108a and 6.108b indicates the stabilized completions are detectable, so with a suitable combination of gain matrix parameters, the SLSC/FO and ASC/FO observers can be designed to estimate the state. Figure 6.108c shows the least squares completion fails the detectability check; thus, pair $(\tilde{A}(t), C_f)_L$ is unobservable.

The selection of the gain matrix parameters for the SLSC/FO and ASC/FO observers involved more trial and error than many of the previous cases. The standard guesses of $\eta = 10$, $\eta = 100$, and $\eta = 1000$ with $\delta = 0.1$ and $\xi = 0.001$ for the SLSC/FO observer return the estimation errors in Figure 6.109 and the following $e(45)$ values:

$$\begin{aligned}\eta = 10 : & \left[\begin{matrix} 0.0 & 0.020920 & -0.161556 & -0.003669 & 0.0 & 0.003669 \end{matrix} \right]^T, \\ \eta = 100 : & \left[\begin{matrix} 0.0 & 0.000410 & -0.021326 & -0.000072 & 0.0 & 0.000072 \end{matrix} \right]^T, \\ \eta = 1000 : & \left[\begin{matrix} 0.0 & 0.030622 & -15.708626 & -0.005371 & 0.0 & 0.005371 \end{matrix} \right]^T.\end{aligned}$$

Increasing η from 10 to 100 reduces the estimation error's oscillatory behavior and decreases the amount of time it takes the SLSC/FO observer to converge to the completion. Although the estimation error appears to converge to zero by $t_f = 45$, the numerical measurement $e(45)$ is not within the $(-10^{-3}, 10^{-3})$ interval. If $\eta = 1000$ instead, the SLSC/FO observer's estimate of state variable $i_l(t)$ worsens, with the estimation error's behavior on the $t \in [0, 45]$ interval in Figure 6.109c implying the magnitude of the difference in the

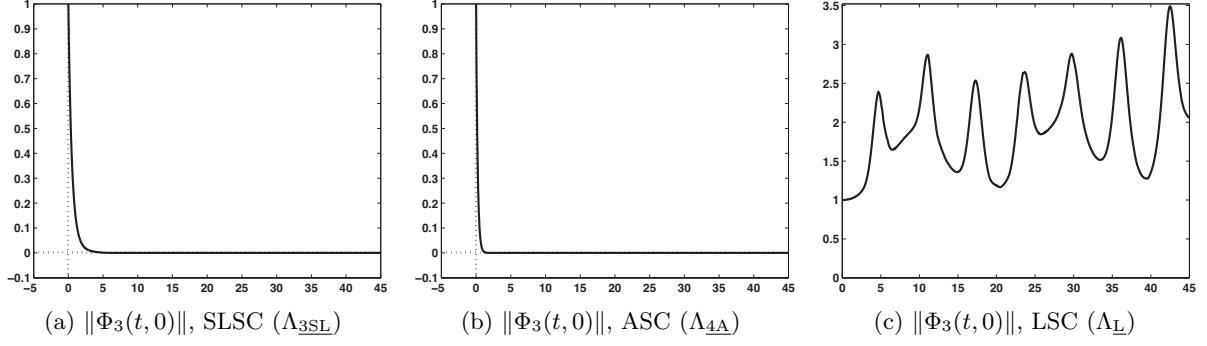


Figure 6.108: $\|\Phi_3(t, 0)\| \rightarrow 0$ for the stabilized completions but not for the LSC. The LSC is not detectable. (C_f)

$i_l(t)$ component will continue to grow for this set of gain matrix parameters. The difference $\tilde{x}(45) - \hat{x}(45) = [0.0 \ -0.001112 \ 0.086147 \ 0.000195 \ 0.0 \ -0.000195]^T$ when $\eta = 150$ is not an improvement over $e(45)$ when $\eta = 100$. Rather, reducing η to 50 produces the desired result: $\tilde{x}(45) - \hat{x}(45) = (1.0 \times 10^{-3}) * [0.0 \ 0.021733 \ -0.587231 \ -0.003812 \ 0.0 \ 0.003812]^T$. The graph of the SLSC/FO observer's estimation error when $\eta = 100$ is included again in Figure 6.110a with a modified vertical scale for comparison with the SLSC/FO observers' estimation errors in Figures 6.110b and 6.110c when $\eta = 150$ and $\eta = 50$, respectively.

A similar guess and check method occurred for selecting η when constructing the ASC/FO observer with output matrix C_f . After trying η as 100, 50, 150, 75, 70, and 65, an ASC/FO observer that converges within $\pm 10^{-3}$ of the completion by $t_f = 45$ is successfully constructed when $\delta = 0.1$, $\eta = 71$, and $\xi = 0.001$. Figure 6.111a presents this ASC/FO observer's estimation error, and its $e(45)$ is $(1.0 \times 10^{-3}) * [0.0 \ -0.025378 \ 0.965575 \ 0.004451 \ 0.0 \ -0.004451]^T$. The estimation error for the ASC/FO observer when $\eta = 75$ is included in Figure 6.111b. Although its $e(45)$ is $(1.0 \times 10^{-2}) * [0.0 \ 0.003779 \ -0.151548 \ -0.000663 \ 0.0 \ 0.000663]^T$, the plot of the two estimation errors together on the $t \in [5, 15]$ interval in Figure 6.111c reveals the ASC/FO observer with $\eta = 71$ does not provide a better estimate of the state for the entire $t \in [0, 45]$ interval.

The rank of $W_o(t)$ for pair $(A_{R_{22}}(t), A_{R_{12}}(t))_{\text{SL}}$ is 2 at $t = 0.01$, 3 from 0.01 to 0.22, and 4 from 0.23 to 135.00. The reduced-order system for the alternative stabilized completion has $\text{rank}(W_o(t)) = 3$ after equaling 2 at $t = 0.01$. In order for the output equation to have an output matrix with the identity/zero block structure $[I \ 0]$, the completions are transformed

by transformation matrix $R = \begin{bmatrix} C_f \\ C_R \end{bmatrix} = \begin{bmatrix} -\vec{e}_4 & \vec{e}_2 & \vec{e}_3 & \vec{e}_1 & \vec{e}_5 & \vec{e}_6 \end{bmatrix}$.

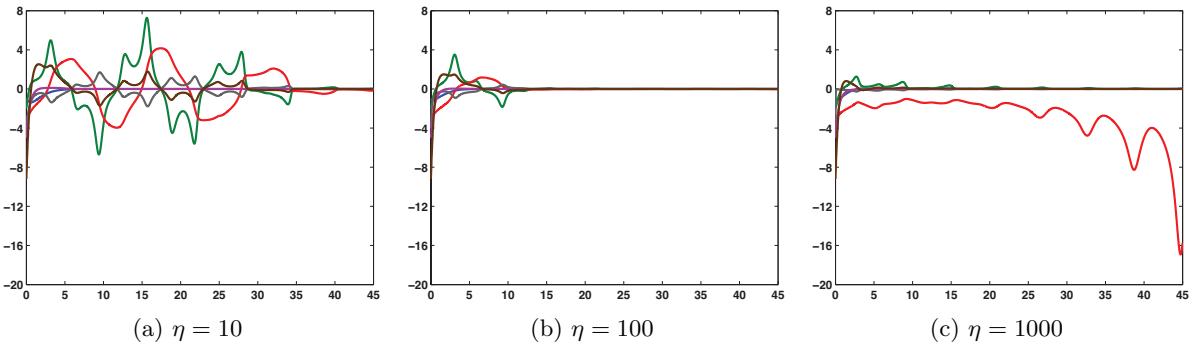


Figure 6.109: SLSC/FO observers' $\tilde{x}(t) - \hat{x}(t)$ as η is varied. For these values of η , $e(45)$ is not within $(-10^{-3}, 10^{-3})$. (C_f , Λ_{3SL} , $\delta = 0.1$, $\xi = 0.001$)

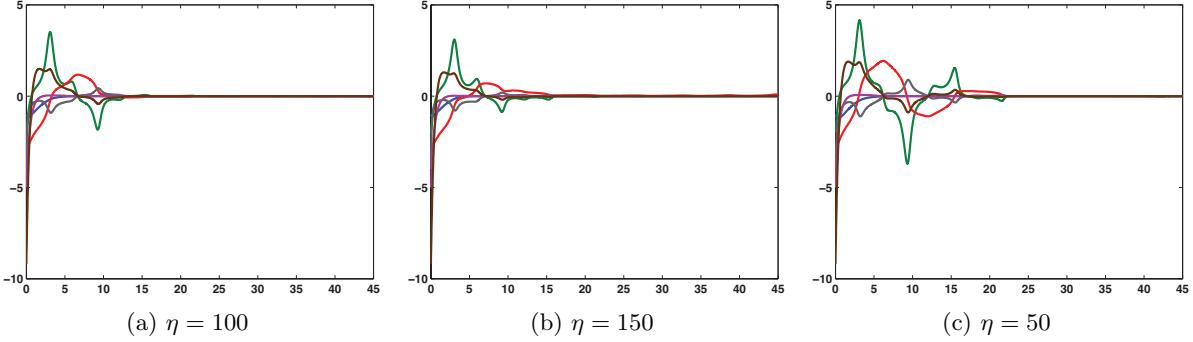


Figure 6.110: SLSC/FO observers' $\tilde{x}(t) - \hat{x}(t)$ as η is varied. When $\eta = 50$, $e(45) \in (-10^{-3}, 10^{-3})$. (C_f, Λ_{3SL} , $\delta = 0.1$, $\xi = 0.001$)

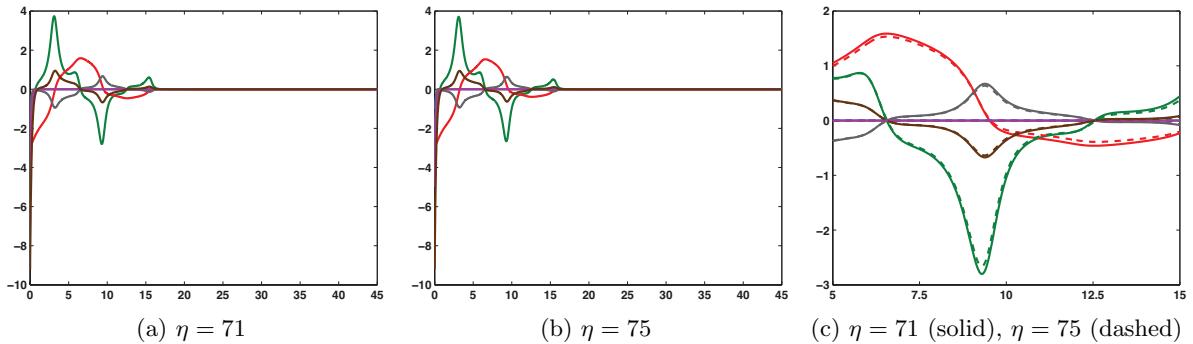


Figure 6.111: ASC/FO observers' $\tilde{x}(t) - \hat{x}(t)$ as η is varied plotted separately on $t \in [0, 45]$ and together on $t \in [5, 15]$. (C_f , Λ_{4A} , $\delta = 0.1$, $\xi = 0.001$)

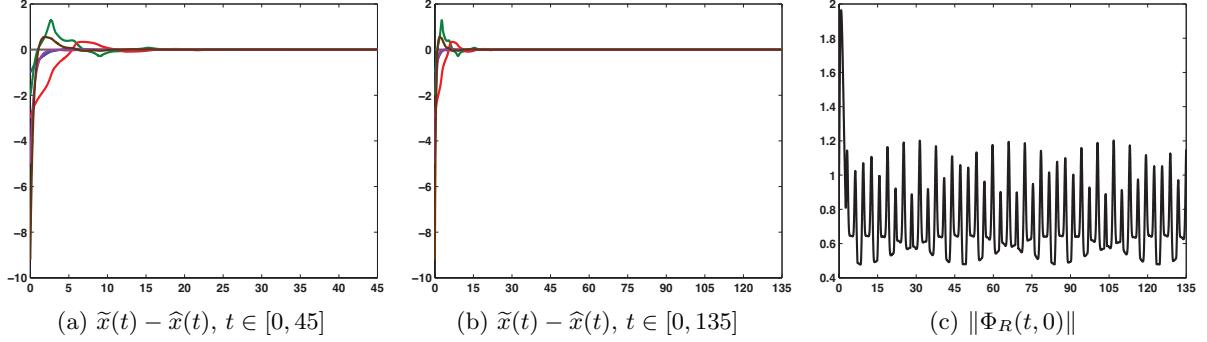


Figure 6.112: SLSC/RO observer's $\tilde{x}(t) - \hat{x}(t)$ on two intervals of time; $\|\Phi_R(t, 0)\|$ is bounded. ($C_f, \Lambda_{3SL}, \delta = 0.1, \eta = 100, \xi = 0.1$)

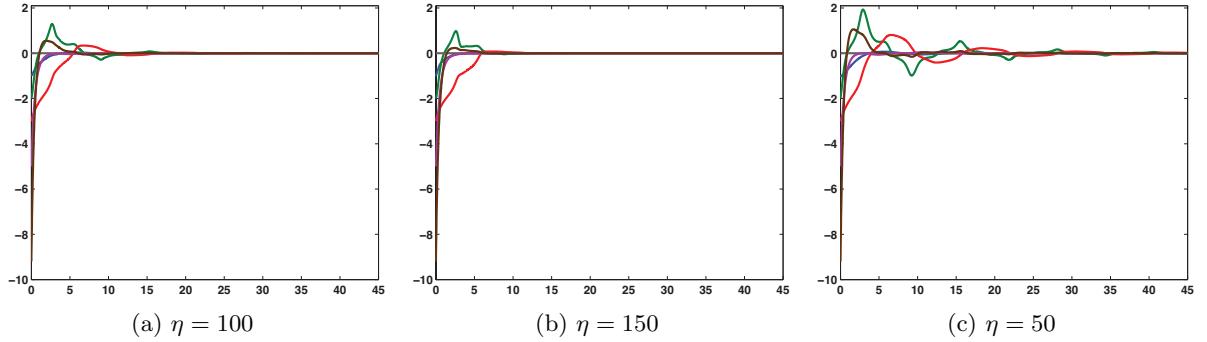


Figure 6.113: SLSC/RO observers' $\tilde{x}(t) - \hat{x}(t)$ as η is varied. When $\eta = 100$, $e(t) \in (-10^{-3}, 10^{-3})$. ($C_f, \Lambda_{3SL}, \delta = 0.1, \xi = 0.1$)

Values of η as specific as those for the construction of the full-order observers are unnecessary when constructing this case's SLSC/RO and ASC/RO observers. For gain matrix parameters $\delta = 0.1$, $\eta = 100$, and $\xi = 0.1$, the SLSC/RO observer appears to converge to the completion by $t_f = 45$ in Figure 6.112a, and the difference $\tilde{x}(45) - \hat{x}(45) = (1.0 \times 10^{-3}) * \begin{bmatrix} 0.008768 & -0.128872 & 0.360912 & 0.0 & -0.003076 & -0.049484 \end{bmatrix}^T$ upholds the visual check for convergence. This combination of reduced-order system and gain matrix parameters does not lead to the observer construction code ending prematurely on the extended time interval. Although $\|\Phi_R(t, 0)\|$ in Figure 6.112c does not converge to zero, its stability indicates any growth of $S_L(t)$ is slow. Figures 6.113b and 6.113c are included to show how the estimation error of the SLSC/RO observer responds when $\eta = 150$ and $\eta = 50$ while keeping $\delta = \xi = 0.1$.

For the same gain matrix parameters that produce a desirable SLSC/RO observer, Figure 6.114c reveals the ASC/RO observer is unable to estimate components $e_2(t)$ and $i_l(t)$ by $t_f = 45$. Instead, decreasing η from 100 to 10 when δ and ξ equal 0.1 reduces the amount of time it takes

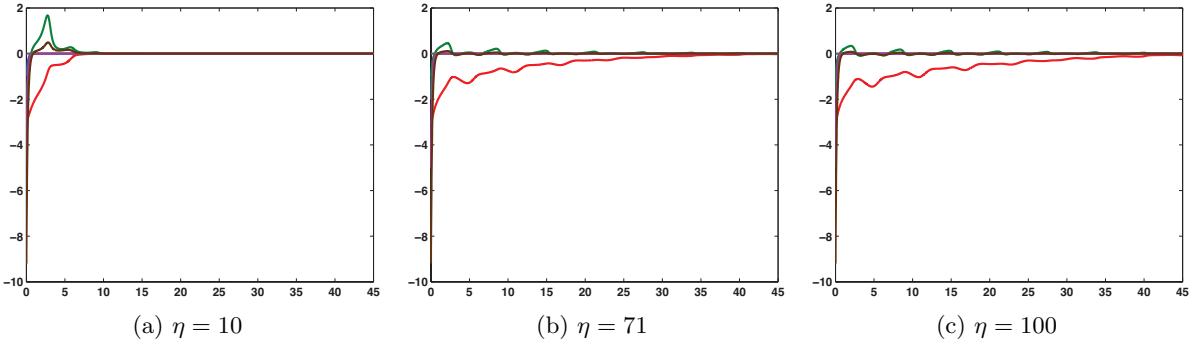


Figure 6.114: ASC/RO observers' $\tilde{x}(t) - \hat{x}(t)$ as η is varied. When $\eta = 10$, $e(45) \in (-10^{-3}, 10^{-3})$. (C_f , $\Lambda_{\underline{4A}}$, $\delta = 0.1$, $\xi = 0.1$)

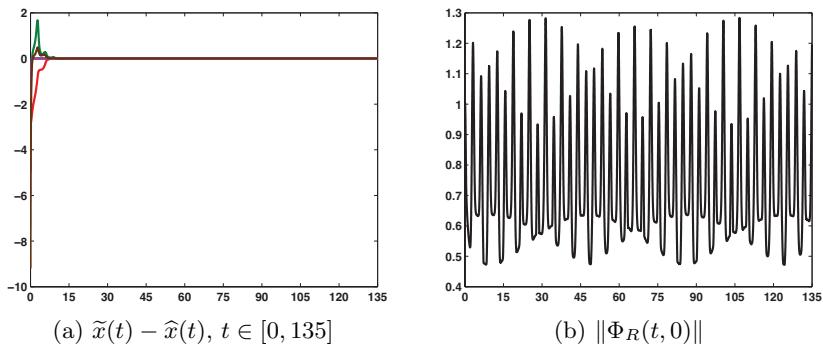


Figure 6.115: ASC/RO observer's $\tilde{x}(t) - \hat{x}(t)$ on the extended time interval; $\|\Phi_R(t, 0)\|$ is bounded. (C_f , Λ_{4A} , $\delta = 0.1$, $\eta = 10$, $\xi = 0.1$)

the ASC/RO observer to converge to $\tilde{x}(t)$ (see Figure 6.114a). Recall, this reduced-order system is detectable, so there appears to exist an upper bound for η if convergence is to occur by a specific time. Notice in Figure 6.115b, $\|\Phi_R(t, 0)\|$ is again bounded, which improves the chances of avoiding the solver's integration tolerance error.

At $t_f = 45$, $e(t) = (1.0 \times 10^{-7}) * \begin{bmatrix} 0.0 & 0.223294 & -0.557975 & 0.0 & 0.0 & 0.082477 \end{bmatrix}^T$ and at $t = 135$, $e(t) = (1.0 \times 10^{-6}) * \begin{bmatrix} 0.0 & -0.133698 & 0.087493 & 0.0 & 0.0 & -0.001581 \end{bmatrix}^T$ for the AS-C/RO observer when $\delta = 0.1$, $\eta = 10$, and $\xi = 0.1$. This increase occurs because the estimation errors continue to oscillate, which can be seen in Figures 6.116a and 6.116b. Throughout this chapter, the numerical measure for convergence has been the interval $(-10^{-3}, 10^{-3})$, but the user may decide on a finer interval if oscillation of a certain magnitude at the desired final time for convergence is a concern.

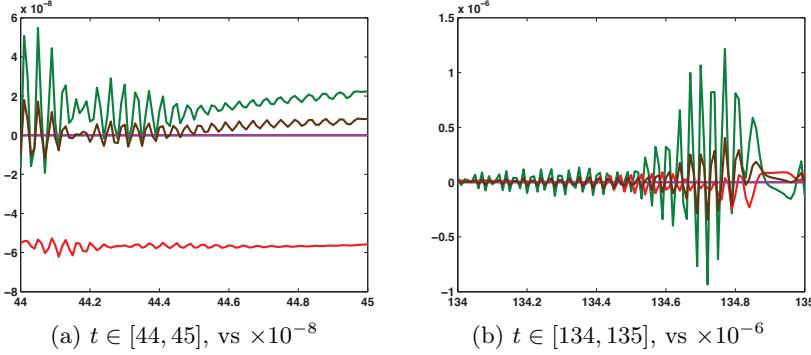


Figure 6.116: ASC/RO observer's $\tilde{x}(t) - \hat{x}(t)$ at two end times. The estimation error may appear to grow numerically as time progresses due to oscillation. (C_f , $\Lambda_{\underline{A}}$, $\delta = 0.1$, $\eta = 10$, $\xi = 0.1$)

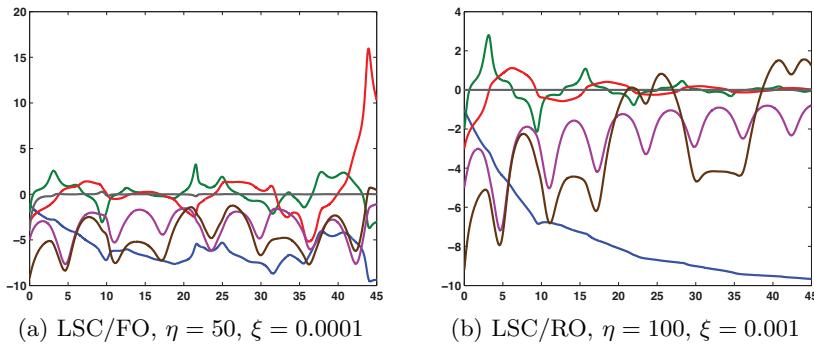


Figure 6.117: $\tilde{x}(t) - \hat{x}(t)$ for LSC/FO and LSC/RO observers. FO and RO observers constructed using the LSC with output matrix C_f cannot be designed to estimate $\tilde{x}(t)$. ($\Lambda_{\underline{L}}$, $\delta = 0.1$)

At the start of this case, Figure 6.108c revealed the least squares completion with output matrix C_f is unobservable, so neither the LSC/FO nor the LSC/RO observer can be constructed to estimate $\tilde{x}(t)$. As examples, consider the estimation errors in Figure 6.117a for the LSC/FO observer with gain matrix parameters $\delta = 0.1$, $\eta = 100$, and $\xi = 0.0001$ and in Figure 6.117b for the LSC/RO observer with gain matrix parameters $\delta = 0.1$, $\eta = 100$, and $\xi = 0.001$.

While the full-order observer has order 6 and the reduced-order observer has order 5, the maximally reduced observer has order 1. Output matrix C_f is linearly independent from the rows of (6.14), so with $C_x = C_f$, extended output equation $\tilde{y}(t) = C_{\Xi} \tilde{x}(t)$ provides information on five state variables. After transforming each completion using transformation matrix $\bar{R}(t) = \begin{bmatrix} C_{\Xi} \\ C_{\bar{R}} \end{bmatrix}$ with $C_{\bar{R}} = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}$, pairs $(A_{\bar{R}_{22}}(t), A_{\bar{R}_{12}}(t))_{\underline{SL}}$, $(A_{\bar{R}_{22}}(t), A_{\bar{R}_{12}}(t))_{\underline{A}}$, and $(A_{\bar{R}_{22}}(t), A_{\bar{R}_{12}}(t))_{\underline{L}}$ are observable. Each of the three observability Gramians has rank 1 for

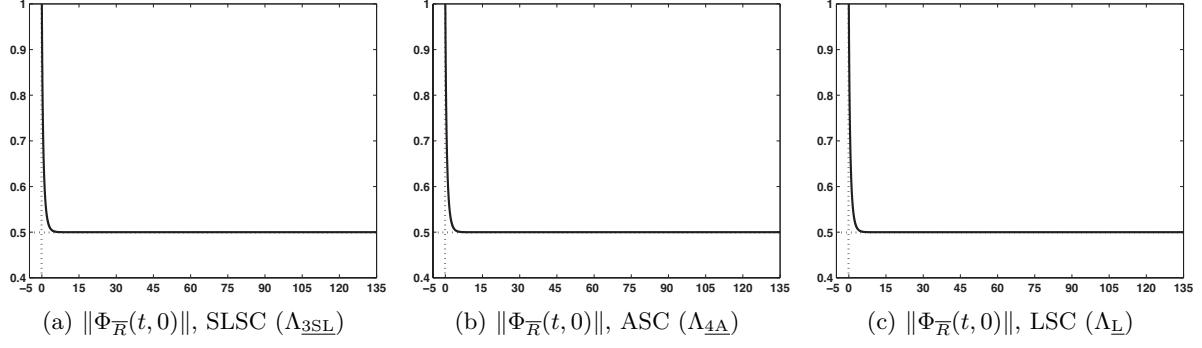


Figure 6.118: $\|\Phi_{\bar{R}}(t, 0)\| \rightarrow 0.5$ for each completion. (C_f)

$t \in (0, 135]$. The norms of these pairs' state transition matrices ($\|\Phi_{\bar{R}}(t, 0)\|$) shown in Figure 6.118 suggest the computations of $S_L(t)$ will not cause any integration tolerance errors.

The differences $\tilde{x}(t) - \hat{x}(t)$ in Figure 6.119 are for the maximally reduced observers constructed with gain matrix parameters $\delta = 0.1$, $\eta = 1000$, and $\xi = 0.1$. At $t_f = 45$, $e(t)$ equals

$$\begin{aligned} \text{SLSC : } & (1.0 \times 10^{-6}) * \begin{bmatrix} 0.025231 & 0.039879 & -0.008204 & 0.0 & -0.011594 & -0.120978 \end{bmatrix}^T, \\ \text{ASC : } & (1.0 \times 10^{-6}) * \begin{bmatrix} -0.032528 & -0.115542 & -0.007346 & 0.0 & 0.021985 & -0.097468 \end{bmatrix}^T, \\ \text{LSC : } & (1.0 \times 10^{-6}) * \begin{bmatrix} 0.236417 & 0.140137 & 0.173040 & 0.0 & -0.054131 & -0.039729 \end{bmatrix}^T. \end{aligned}$$

Figures 6.120a and 6.120b reiterate the maximally reduced observer's estimate is independent of the completion or stabilization parameter matrix but Figure 6.120c shows the observer's convergence is dependent on η using an estimation error progression. Also, the ease at which the maximally reduced observer was constructed for this case and its resulting state estimate are excellent indicators that this observer is a promising alternative to the full-order and reduced-order observers.

If the estimation error for this case's maximally reduced observer seems familiar, it is because of the maximally reduced observer results from Case 4 of the example with positive resistors. Figures 6.121a and 6.121b plot the completion solution (solid) with the maximally reduced observer's estimate (dashed) for component $i_l(t)$ from the examples with negative and positive resistors. Neither the completion solutions nor the maximally reduced observers are identical for the examples' Case 4 results but their estimation errors are (see Figures 6.62b and 6.119). The difference in Figure 6.121c ($(e(t), \text{Neg}) - (e(t), \text{Pos})$) is zero except for some numerical error. The modification to the characterization of the solution manifold from the resistors' sign change, seen in the first two rows of (6.14) (compared with (6.11)), affects the maximally reduced observer but not its performance.

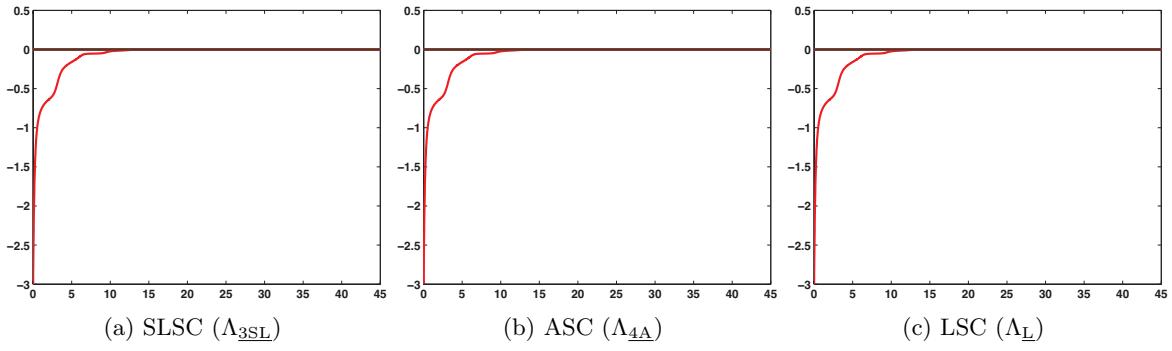


Figure 6.119: $\tilde{x}(t) - \hat{x}(t)$ for SLSC/MR, ASC/MR, and LSC/MR observers. ($C_f, \delta = 0.1, \eta = 1000, \xi = 0.1$)

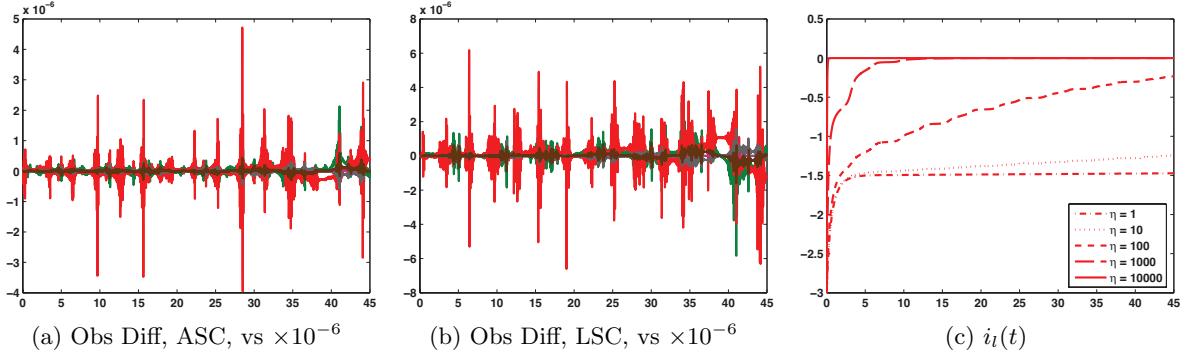


Figure 6.120: The observer differences, SLSC – ASC and SLSC – LSC, confirm the construction of the MR observer distinguishes between neither completion nor Λ . $\tilde{x}(t) - \hat{x}(t)$ progression in $i_l(t)$; varying η affects the MR observer's convergence. ($C_f, \Lambda_{3\text{SL}}, \Lambda_{4\text{A}}, \Lambda_{\text{L}}, \delta = 0.1, \xi = 0.1$)

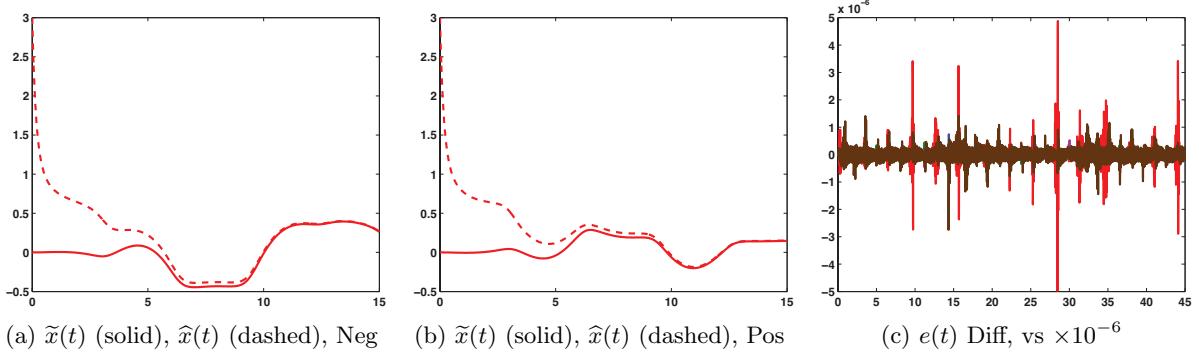


Figure 6.121: Completion solution and MR observer estimate in $i_l(t)$ for two examples (negative, positive resistors). The estimation error difference is $(e(t), \text{Neg}) - (e(t), \text{Pos})$. The completions' solutions and observers are not the same but the observers' estimation errors are. ($C_f, \delta = 0.1, \eta = 1000, \xi = 0.1$)

6.3 Observer Construction Code

RunLTV.m: Example specific decisions are made in this script. The user chooses the output matrix, selects the observer and completion, defines Λ , picks initial and final times, assigns the gain matrix parameters, and turns on or off seven switches in order to identify which sections of code should run. This script calls LTVcircuit.m if constructing either the full-order or reduced-order observer and ExtendOutput.m if constructing the maximally reduced observer. (pp. 252–255)

LTVcircuit.m: Main calling program. This function checks Criteria 2 and 3 from the selection of Λ , calculates the rank of the observability Gramian, computes the norms of the state transition matrices, and presents the observer construction results. (pp. 255–267)

SLSCcoeff.m, ASCcoeff.m: These functions are called by LTVcircuit.m, WoRank.m, LTVDAE_Solveplus.m, and LTVComp_Solve.m to calculate the completions' coefficients \tilde{A} and \tilde{B} and the first derivative of \tilde{A} at specific values of time. (pp. 267–270, pp. 271–279)

WoRank.m: This function is called by LTVcircuit.m to calculate the rank of the observability Gramian and to return state transition matrices (including ones for unobservable systems and for reduced-order systems). (pp. 280–283)

LTVDAE_Solveplus.m: This function is called by LTVcircuit.m to find solutions for the completion, the transformed completion, and the observer. (pp. 283–287)

ExtendOutput.m: This function extends the output matrix for constructing the maximally reduced observer. If the extended output matrix is nonsingular, then LTVComp_Solve.m is called to explicitly solve for the state. If the extended output matrix is not full column rank, then LTVcircuit.m is called and the process for constructing the maximally reduced observer continues. Note, if $C_{\bar{R}}$ is returned as a time-varying matrix, then it needs to be hard coded where appropriate. (pp. 287–291, pp. 291–292)

CompUnobs.m: This script compares the full-order and reduced-order observers' unobservable subspaces based on theory in Section 4.3. (pp. 292–294)

Functions Rounding.m and DAEobservable.m listed in Section 5.3 were also called during the analysis of the linear time-varying example system. Note, tic toc outputs listed throughout this chapter describing run times were for observer construction only; that is, the switches for Criteria 2 and 3 calculations and for observability checks in function LTVcircuit.m were off.

At times, symbolic matrices were defined using both MATLAB and Maple due to command availability for symbolic computations and to different result structures returned by related

commands. As an example, the following steps outline the process for symbolically defining the alternative stabilized completion's coefficient matrix $\tilde{A}(t)$.

1. In Maple using the LinearAlgebra package, define matrices $E(t)$, $F(t)$ from the linear time-varying system of DAEs and matrices $\mathcal{E}(t)$, $\mathcal{F}(t)$ from the derivative array equations found without stabilized differentiation.
2. With MATLAB and symbolic variable t , use the null command to find matrix $Z_2(t)^T$, where the columns of $Z_2(t)$ form a basis for $N(\mathcal{E}(t)^T)$, and enter the result in Maple.
3. In Maple, Multiply $Z_2(t)^T \mathcal{F}(t)$.
4. With MATLAB, use the null command to find matrix $T_2(t)$, where the columns of $T_2(t)$ form a basis for $N(Z_2(t)^T \mathcal{F}(t))$, and enter the result in Maple.
5. In Maple, Multiply $E(t)T_2(t)$. Use Maple's ColumnSpace command to find matrix $Z_{1,0}(t)^T$. The ColumnSpace command returns basis vectors rather than a matrix with columns equaling the basis vectors for $R(E(t)T_2(t))$, so matrix $Z_{1,0}(t)$ must be built from the vectors and then transposed.
6. In Maple, Multiply $Z_{1,0}(t)^T E(t)$ and then define
$$\begin{bmatrix} Z_{1,0}(t)^T E(t) \\ Z_2(t)^T \mathcal{F}(t) \end{bmatrix}.$$
7. With MATLAB, use the inv command to find the inverse of the matrix defined in Step 6 and enter the result in Maple.
8. In Maple, Multiply $Z_{1,0}(t)^T F(t)$; define a general stabilization parameter matrix Λ_A (variable rather than numerical entries); and use the map and diff commands with Multiply to compute $(Z_2(t)^T \mathcal{F}(t))' + \Lambda_A Z_2(t)^T \mathcal{F}(t)$. Then the coefficient matrix of $x(t)$ in equation (2.11) can be defined.

Chapter 7

Conclusions, Contributions, and Future Research Topics

7.1 Conclusions

The linear time-invariant example in Chapter 5 was the first test of our proposal to construct observers for systems of DAEs using completions. The cases for which observers could be constructed analyzed how the user's choices for λ (> 0 , stabilized differentiation; $= 0$, least squares completion) and for ρ (from the construction of the gain matrix) affected the convergence and estimates of the full-order, reduced-order, and maximally reduced observers. Our maximally reduced observer, incorporating information from the constraints characterizing the solution manifold with the output equation, consistently had comparable but usually better convergence properties and rates than the previously developed full-order and reduced-order observers. It was shown experimentally that the maximally reduced observer's estimate is the same no matter the completion or λ . Thus, figuring which completion with what λ will produce the best estimate is unnecessary, reducing variability in the observer's construction.

The theory we developed connecting observers and completions was applied to the analysis of Case 2 when it was discovered the matrix pencil eigenvalues were unobservable. Established theory connecting the construction of the full-order and reduced-order observers did not carry over to the construction of the maximally reduced observer. Rather, the maximally reduced observer constructed using the least squares completion successfully estimated the state even though the least squares completion and its reduced-order system were unobservable due to the additional dynamics eigenvalues.

Our DAE manifold observer also produced promising results. With an estimation error comparable to the alternative stabilized completion's results for the same ρ , the DAE manifold observer's use of the canonical form of a matrix pencil and the characterization of the solution

manifold allows the observer to be constructed without finding a completion. This observer's development in the time-varying case is discussed as a future research topic in Section 7.3.

In Chapter 6, we applied our observer construction approach to a linear time-varying system of DAEs describing a circuit. Constructing an observer using the least squares completion or to estimate the example with negative resistors proved challenging when the norms of the systems' state transition matrices were unbounded. Manipulating δ , η , and ξ , what we identified as the gain matrix parameters, delayed the solver's integration tolerance error so that our observer construction approach could estimate the state by a desired final time even though the exponentially asymptotically stable $\tilde{A}(t)$ condition included in [2], [3] was not met. Additionally, reducing the order of the observer improved the stability of some systems so that the calculation of the Riccati equation's $S_L(t)$ from the definition of the gain matrix did not cause the observer construction code to end prematurely.

One of the reasons the authors of [63] decided to develop their own observer for linear time-varying systems of ODEs was to avoid the Lyapunov theory, Riccati equation method for computing the gain matrix. They wanted to know how their choices affected the observer's convergence rate and felt the responses from the user's inputs for the Riccati-derived gain matrix could not be generalized. Our choices for δ , η , and ξ and our decision to test other values in consistent increments allowed us to develop a sense of how the convergence rate would respond to our gain matrix parameters. For example, the value of ξ does not affect convergence, but the values of δ and η do. By making $M = \eta I$ constant, we identified that increasing η improves the convergence rate and affects the observer's estimate but that there may be a maximum η if convergence is to occur by a specific time.

Unlike the linear time-invariant case, a connection between stabilized differentiation and the additional dynamics of linear time-varying completions is not clear. We introduced a four-criteria check for selecting Λ in order to analyze the stabilization parameter matrices' potential effects on observability and on growth impacting the solver's calculation of $S_L(t)$.

For many of the cases, enough information was known about the state vector through the constraints characterizing the solution manifold and the output equation that $x(t)$ could be solved for explicitly using the extended output equation. Thus, no observers, as well as no completions when data is available, were required for determining the states of the linear time-varying systems of DAEs. When the extended output matrix was not full column rank, the resulting maximally reduced observers' estimates were affected by neither the completion, the stabilization parameter matrix, nor the unobservability of the LSC or its reduced-order system (as long as the stabilized completions were either observable or detectable). The maximally reduced observer and its construction process provide new ways for determining the state vector, taking advantage of linear systems of DAEs' characteristics that have previously made their solution and observation daunting.

Our observer construction approach - constructing observers for systems of DAEs using completions - was successfully implemented for linear time-invariant and linear time-varying systems of DAEs. Our only assumptions on the linear system of DAEs were that they be solvable and that their coefficient matrices be smooth matrix functions in the time-varying case. Otherwise, structural assumptions such as having a specific index or being in Hessenberg form were unnecessary. Our observer construction approach has the potential to be extended to nonlinear systems of DAEs. This possibility of developing a general approach applicable to observing linear and nonlinear systems of DAEs is further discussed in Section 7.3 following Section 7.2's list of our research's contributions. Note, article [58] is indirectly related to this dissertation's research since it focuses on solving rather than observing systems of DAEs.

7.2 Contributions

- [23]: K. Bobinyec, S. L. Campbell, and P. Kunkel, “Constructing Observers for Linear Time Varying DAEs”, *Proc. 51st IEEE Conference on Decision and Control*, Maui, HI, 2012, pp. 5749–5754. Presentation by K. Bobinyec.
- [24]: K. Bobinyec, S. L. Campbell, and P. Kunkel, “Full order observers for linear DAEs”, *Proc. 50th IEEE Conference on Decision and Control and European Control Conference*, Orlando, FL, 2011, pp. 4011–4016. Presentation by K. Bobinyec.
- [25]: K. Bobinyec, S. L. Campbell, and P. Kunkel, “Maximally reduced observers for linear time varying DAEs”, *Proc. IEEE Multi-Conference on Systems and Control*, Denver, CO, 2011, pp. 1373–1378. Presentation by K. Bobinyec.
- [26]: K. Bobinyec, S. L. Campbell, and P. Kunkel, “Stabilized Completions of Differential Algebraic Equations and the Design of Observers”, *Proc. Neural, Parallel, and Scientific Computations*, Atlanta, GA, 2010. Presentation by S. L. Campbell.
- [58]: S. L. Campbell, P. Kunkel, and K. Bobinyec, “A minimal norm corrected underdetermined Gauss-Newton procedure”, *Appl. Numer. Math.*, 62 (2012), pp. 592–605. Paper only.
- “Observer Construction for Systems of DAEs Using Completions”, International Council for Industrial and Applied Mathematics (ICIAM), Vancouver, BC, Canada, 2011. Presentation only, K. Bobinyec.

7.3 Future Research Topics

1) LINEAR TIME-VARYING EXAMPLE: We made some assumptions in Chapter 6 designed to simplify observer construction. The output matrices were constant as were $\bar{Q} = \xi I$ (from $Q_L(t) = \bar{Q}(t) + C(t)^T M(t)C(t)$, for example) and $M = \eta I$ from the computation of the gain matrix. One modification is to vary these matrices with time, but having any necessary derivatives available also needs to be kept in mind. Another suggestion is to keep \bar{Q} and M constant but to change their and $S_L(t_0)$'s diagonal entries so they are not all ξ , η , or δ . For example, investigate if and how the results when $M = \begin{bmatrix} 100 & 0 \\ 0 & 1000 \end{bmatrix}$ are different from the ones when $M = 100I$ or $M = 1000I$. An observation was made that a possible upper bound may exist for M if the observer is to converge by a desired final time. A second look at the development of the gain matrix and piecing together more estimation error progressions are good starting places for determining if this upper bound observation can be turned into theory. For some of the pairs, the observability check returned a rank fluctuation, but for others, the rank of the observability Gramian dropped for longer periods of time. This drop hints that incorrect ranks were being calculated, so formulating a more robust observability check should be considered. Recently, a closer look at the numerical results for the detectability check revealed the algorithm is not producing a matrix of the structure $\mathcal{A}_u(t) = \begin{bmatrix} A_1(t) & 0 \\ A_2(t) & A_3(t) \end{bmatrix}$ by $t = 135$ for pair $(\tilde{A}(t), C_f)_L$ or by $t_f = 45$ for pairs $(\tilde{A}(t), C_d)_L$ and $(\tilde{A}(t), C_f)_L$. Further analysis is required to discover if the transformation matrix constructed from the nullspace of the observability Gramian or some other explanation is causing this miscalculation. Since the numerical results support the conclusion that pair $(\tilde{A}(t), C_e)_L$ is unobservable, the conclusions about the other pairs' unobservability are not expected to change. Finally, the derivation of the reduced-order observer revealed matrix P_2 could be constant in addition to being nonsingular, so we chose $P_2 = I$. Trying constant matrices other than the identity may reveal this matrix is another parameter that can be affected to influence the reduced-order and maximally reduced observers.

2) COMPLETIONS AND OBSERVABILITY THEORY: The proof of Theorem 4.1 showed that $N(M) \cap N(\mathcal{O}_\Delta) = N(M) \cap N(\mathcal{O}_{\tilde{A}})$ for a linear time-invariant system of DAEs. The focus is on the observability of the matrix pencil eigenvalues, but considering the nullspaces of observability Gramians rather than of observability matrices may lead to theoretically extending this completion and observer connection to linear time-varying systems of DAEs.

3) DAE MANIFOLD OBSERVER: A standard canonical form described in [35] and [53] is the first step in extending the DAE manifold observer to solvable linear time-varying systems of

DAEs. The second step is to figure out how to determine matrices $\mathcal{P}(t)$ and $\mathcal{Q}(t)$ for revealing the canonical form. In the linear time-invariant case, a command in Scilab provided the desired matrices that were then hard coded into MATLAB. A better understanding of Scilab would reveal if pencan or another related command could return symbolic definitions for $\mathcal{P}(t)$ and $\mathcal{Q}(t)$ or if there was some other way to have Scilab work in conjunction with MATLAB [59]. If not, a similar algorithm could be designed for MATLAB if one does not already exist in a special toolbox.

4) OTHER OBSERVER APPLICATIONS: Possible directions for future research topics include developing the observer construction approach for discrete time systems of DAEs; modifying the observers to accommodate unknown inputs or systems with delays as in [43], [71], [92], and [104]; and constructing robust observers for easily perturbed systems [118].

5) NONLINEAR SYSTEMS OF DAEs: Research on observing nonlinear systems of ODEs is included in articles [75], [90], [99], [161], and [166], which span over twenty years with the most recent of this list being published in 2009. Observer theory for nonlinear systems of ODEs is not well-established like it is for linear systems of ODEs. The consideration of systems of DAEs has introduced another challenge to observing nonlinear systems. Research published in [27] constructs an observer for a nonlinear time-invariant system of DAEs but assumes the coefficient matrix of the first derivative of the state vector has a specific structure. The authors of [69] and [172] also consider observing nonlinear time-invariant systems of DAEs. Another proposal designs the observer for a nonlinear system of DAEs as an index 1 system of DAEs rather than as a system of ODEs [10]. In [176], a transformation produces an explicit nonlinear system that can then be observed using methods for systems of ODEs. The extension of our observer construction approach will build on the foundation laid in [38], [39], [40], [48], [49], [52], and [108] for completions of nonlinear systems of DAEs and in [65], [79], and [175] for nonlinear Luenberger-like observers. When the approach is ready to be tested, examples of nonlinear systems of DAEs are available in [113] and [145].

6) COMPUTATIONAL EFFICIENCY: More of a goal for future algorithms rather than a topic for future research, we would like our numerical algorithms to be independent from any example specific content. There would be a calling script in which the user would define the system of DAEs to be observed, but the main set of programs would be general and not require adjustment by the user. In order to improve computational efficiency, though, we removed symbolic differentiations by hard coding derivatives that had been found outside of the numerical algorithms. Computational efficiency was addressed in Chapter 6, but further evaluation is required and can begin by considering [156].

REFERENCES

- [1] B. D. O. Anderson and J. B. Moore, “Detectability and Stabilizability of Time-Varying Discrete-Time Linear Systems”, *SIAM J. Control and Optimization*, 19 (1981), pp. 20–32.
- [2] B. D. O. Anderson and J. B. Moore, “New Results in Linear System Stability”, *SIAM J. Control*, 7 (1969), pp. 398–414.
- [3] B. D. Anderson and J. B. Moore, “Time-Varying Version of the Lemma of Lyapunov”, *Electronics Letters*, 3 (1967), pp. 293–294.
- [4] P. J. Antsaklis and A. N. Michel, *Linear Systems*, Birkhäuser, Boston, 2006.
- [5] J. D. Aplevich, *Implicit Linear Systems*, Lecture Notes in Control and Information Sciences Vol. 152, Springer-Verlag, Berlin, 1991.
- [6] U. M. Ascher and L. R. Petzold, *Computer Methods for Ordinary Differential Equations and Differential-Algebraic Equations*, SIAM, Philadelphia, 1998.
- [7] U. M. Ascher and L. R. Petzold, “Stability of Computational Methods for Constrained Dynamics Systems”, *SIAM J. Sci. Comput.*, 14 (1993), pp. 95–120.
- [8] U. M. Ascher, H. Chin, and S. Reich, “Stabilization of DAEs and invariant manifolds”, *Numer. Math.*, 67 (1994), pp. 131–149.
- [9] U. M. Ascher, H. Chin, L. R. Petzold, and S. Reich, “Stabilization of Constrained Mechanical Systems with DAEs and Invariant Manifolds”, *Mech. Struct. & Mach.*, 23 (1995), pp. 135–157.
- [10] J. Åslund and E. Frisk, “An observer for non-linear differential-algebraic systems”, *Automatica*, 42 (2006), pp. 959–965.
- [11] A. Barrlund, “Constrained least squares methods for linear timevarying DAE systems”, *Numer. Math.*, 60 (1991), pp. 145–161.
- [12] A. Barrlund and B. Kågström, “Analytical and numerical solutions to higher index linear variable coefficient DAE systems”, *J. Comput. Appl. Math.*, 31 (1990), pp. 305–330.
- [13] M. Baumann and U. Helmke, “Singular value decomposition of time-varying matrices”, *Future Generation Computer Systems*, 19 (2003), pp. 353–361.
- [14] J. Baumgarte, “Stabilization of constraints and integrals of motion in dynamical systems”, *Computer Methods in Applied Mechanics and Engineering*, 1 (1972), pp. 1–16.
- [15] J. S. Bay, *Fundamentals of Linear State Space Systems*, WCB/McGraw-Hill, New York, 1999.
- [16] S. J. G. Bell and N. K. Nichols, *A Numerical Method for the Computation of the Analytic Singular Value Decomposition*, Numerical Analysis Report 2, University of Reading, Reading, United Kingdom, 1994.

- [17] T. Berger and A. Ilchmann, “Zero dynamics of linear time-varying linear systems”, *Ilmenau University of Technology, Tech. Rep.*, 2010.
- [18] N. Biehn, S. L. Campbell, F. Delebecque, and R. Nikoukhah, “Observer Design for Linear Time Varying Descriptor Systems: Numerical Algorithms”, *Proc. 37th IEEE Conference on Decision and Control*, Tampa, FL, 1998, pp. 3801–3806.
- [19] N. Biehn, S. L. Campbell, R. Nikoukhah, and F. Delebecque, “Numerically constructible observers for linear time-varying descriptor systems”, *Automatica*, 37 (2001), pp. 445–452.
- [20] S. Bittanti, P. Bolzern, and P. Colaneri, “The Extended Periodic Lyapunov Lemma”, *Automatica*, 21 (1985), pp. 603–605.
- [21] W. Blajer, “Index of differential-algebraic equations governing the dynamics of constrained mechanical systems”, *Appl. Math. Modelling*, 16 (1992), pp. 70–77.
- [22] F. Blanchini, “Eigenvalue assignment via state observer for descriptor systems”, *Kybernetika*, 27 (1991), pp. 384–392.
- [23] K. Bobinyec, S. L. Campbell, and P. Kunkel, “Constructing Observers for Linear Time Varying DAEs”, *Proc. 51st IEEE Conference on Decision and Control*, Maui, HI, 2012, pp. 5749–5754.
- [24] K. Bobinyec, S. L. Campbell, and P. Kunkel, “Full Order Observers for Linear DAEs”, *Proc. 50th IEEE Conference on Decision and Control and European Control Conference*, Orlando, FL, 2011, pp. 4011–4016.
- [25] K. Bobinyec, S. L. Campbell, and P. Kunkel, “Maximally reduced observers for linear time varying DAEs”, *Proc. 2011 IEEE Multi-Conference on Systems and Control*, Denver, CO, 2011, pp. 1373–1378.
- [26] K. Bobinyec, S. L. Campbell, and P. Kunkel, “Stabilized Completions of Differential Algebraic Equations and the Design of Observers”, *Proc. Neural, Parallel, and Scientific Computations*, Atlanta, GA, 2010.
- [27] D. Boutat, G. Zheng, L. Boutat-Baddas, and M. Darouach, “Observers design for a class of nonlinear singular systems”, *Proc. 51st IEEE Conference on Decision and Control*, Maui, HI, 2012, pp. 7407–7412.
- [28] K. E. Brenan, S. L. Campbell, and L. R. Petzold, *Numerical Solution of Initial-Value Problems in Differential-Algebraic Equations*, SIAM, Philadelphia, 1996.
- [29] J. J. Brophy, *Basic Electronics for Scientists*, McGraw-Hill, New York, 1966.
- [30] P. N. Brown, A. C. Hindmarsh, and L. R. Petzold, “Consistent Initial Condition Calculation for Differential-Algebraic Systems”, *SIAM J. Sci. Comput.*, 19 (1998), pp. 1495–1512.
- [31] A. Bunse-Gerstner, R. Byers, V. Mehrmann, and N. K. Nichols, “Numerical computation of an analytic singular value decomposition of a matrix valued function”, *Numer. Math.*, 60 (1991), pp. 1–39.

- [32] K. Burrage and L. Petzold, “On Order Reduction for Runge-Kutta Methods Applied to Differential/Algebraic Systems and to Stiff Systems of ODEs”, *SIAM J. Numer. Anal.*, 27 (1990), pp. 447–456.
- [33] G. D. Byrne and P. R. Ponzi, “Differential-Algebraic Systems, Their Applications and Solutions”, *Comput. Chem. Engng.*, 12 (1988), pp. 377–382.
- [34] S. L. Campbell, “A Computational Method for General Higher Index Nonlinear Singular Systems of Differential Equations”, *Proc. 12th IMACS World Congress on Scientific Computation*, Paris, France, 1988, pp. 178–180.
- [35] S. L. Campbell, “A General Form for Solvable Linear Time Varying Singular Systems of Differential Equations”, *SIAM J. Math. Anal.*, 18 (1987), pp. 1101–1115.
- [36] S. L. Campbell, “High-Index Differential Algebraic Equations”, *Mech. Struct. & Mach.*, 23 (1995), pp. 199–222.
- [37] S. L. Campbell, “Index Two Linear Time Varying Singular Systems of Differential Equations”, *Circuits Systems Signal Process*, 5 (1986), pp. 97–107.
- [38] S. L. Campbell, “Least squares completions for nonlinear differential algebraic equations”, *Numer. Math.*, 65 (1993), pp. 77–94.
- [39] S. L. Campbell, “Least Squares Completions of Nonlinear Higher Index Hessenberg DAEs”, *Computational and Applied Mathematics, II: Differential Equations: Selected and Revised Papers from the IMACS 13th World Congress, Dublin, Ireland*, North-Holland, New York, 1992, pp. 117–126.
- [40] S. L. Campbell, “Least Squares Completions of Nonlinear Index Three Hessenberg DAEs”, *Proc. 13th IMACS World Congress on Computation and Applied Mathematics*, Dublin, Ireland, 1991, pp. 1145–1148.
- [41] S. L. Campbell, “Local Realizations of Time Varying Descriptor Systems”, *Proc. 26th IEEE Conference on Decision and Control*, Los Angeles, CA, 1987, pp. 1129–1130.
- [42] S. L. Campbell, “The Numerical Solution of Higher Index Linear Time Varying Singular Systems of Differential Equations”, *SIAM J. Sci. Stat. Comput.*, 6 (1985), pp. 334–348.
- [43] S. L. Campbell, “Singular Linear Systems of Differential Equations with Delays”, *Applicable Analysis*, 11 (1980), pp. 129–136.
- [44] S. L. Campbell, *Singular Systems of Differential Equations*, Pitman, Boston, 1980.
- [45] S. L. Campbell, *Singular Systems of Differential Equations II*, Pitman, Boston, 1982.
- [46] S. L. Campbell, “Uniqueness of Completions for Linear Time Varying Differential Algebraic Equations”, *Lin. Alg. Appl.*, 161 (1992), pp. 55–67.
- [47] S. L. Campbell and C. W. Gear, “The index of general nonlinear DAEs”, *Numer. Math.*, 72 (1995), pp. 173–196.

- [48] S. L. Campbell and E. Griepentrog, “Solvability of General Differential Algebraic Equations”, *SIAM J. Sci. Comput.*, 16 (1995), pp. 257–270.
- [49] S. L. Campbell and P. Kunkel, “Completions of nonlinear DAE flows based on index reduction techniques and their stabilization”, *J. Comput. Appl. Math.*, 233 (2009), pp. 1021–1034.
- [50] S. L. Campbell and B. Leimkuhler, “Differentiation of Constraints in Differential-Algebraic Equations”, *Mech. Struct. & Mach.*, 19 (1991), pp. 19–39.
- [51] S. L. Campbell and C. D. Meyer, *Generalized Inverses of Linear Transformations*, SIAM, Philadelphia, 2009.
- [52] S. L. Campbell and E. Moore, “Progress on a General Numerical Method for Nonlinear Higher Index DAEs II”, *Circuits Systems Signal Process*, 13 (1994), pp. 123–138.
- [53] S. L. Campbell and L. R. Petzold, “Canonical Forms and Solvable Singular Systems of Differential Equations”, *SIAM J. Alg. Disc. Meth.*, 4 (1983), pp. 517–521.
- [54] S. L. Campbell and W. J. Terrell, “Observability of Linear Time-Varying Descriptor Systems”, *SIAM J. Matrix Anal. Appl.*, 12 (1991), pp. 484–496.
- [55] S. L. Campbell and K. D. Yeomans, “Behavior of the nonunique terms in general DAE integrators”, *Appl. Numer. Math.*, 28 (1998), pp. 209–226.
- [56] S. L. Campbell, F. Delebecque, and R. Nikoukhah, “Observer design for linear time varying descriptor systems”, *Proc. Control Industrial Systems (CIS97)*, Belfort, France, 1997, pp. 507–512.
- [57] S. L. Campbell, C. T. Kelley, and K. D. Yeomans, “Consistent Initial Conditions for Unstructured Higher Index DAEs: A Computational Study”, *Proc. Computational Engineering in Systems Applications*, Lille, France, 1996, pp. 416–421.
- [58] S. L. Campbell, P. Kunkel, and K. Bobinyec, “A minimal norm corrected underdetermined Gauss-Newton procedure”, *Appl. Numer. Math.*, 62 (2012), pp. 592–605.
- [59] S. L. Campbell, R. Hollenbeck, K. Yeomans, and Y. Zhong, “Mixed symbolic-numerical computations with general DAEs I: System properties”, *Numerical Algorithms*, 19 (1998), pp. 73–83.
- [60] Y. Chahlaoui and P. Van Dooren, “Estimating Gramians of Large-Scale Time-Varying Systems”, *Proc. 15th IFAC World Congress*, Barcelona, Spain, Paper 2440, 2002.
- [61] W. Chai and N. K. Loh, “Design of minimal-order state observers for time-varying multi-variable systems”, *Int. J. Systems Sci.*, 23 (1992), pp. 581–592.
- [62] W. Chai, N. K. Loh, and H. Hu, “Observer design for time-varying systems”, *Int. J. Systems Sci.*, 22 (1991), pp. 1177–1196.

- [63] M.-S. Chen and J.-Y. Yen, “Application of the Least Squares Algorithm to the Observer Design for Linear Time-Varying Systems”, *IEEE Trans. Automat. Contr.*, 44 (1999), pp. 1742–1745.
- [64] J.-L. Chern and L. Dieci, “Smoothness and Periodicity of Some Matrix Decompositions”, *SIAM J. Matrix Anal. Appl.*, 22 (2000), pp. 772–792.
- [65] G. Ciccarella, M. Dalla Mora, and A. Germani, “A Luenberger-like observer for nonlinear systems”, *Int. J. Control*, 57 (1993), pp. 537–556.
- [66] K. D. Clark, “A Structural Form for Higher-Index Semistate Equations I: Theory and Applications to Circuit and Control Theory”, *Lin. Alg. Appl.*, 98 (1988), pp. 169–197.
- [67] C. J. Dafis and C. O. Nwankpa, “Addressing Nonlinear Observability Issues in Power Systems”, *Proc. 14th Power Systems Computation Conference*, Seville, Spain, 2002.
- [68] M. Darouach, “Functional Observers For Linear Descriptor Systems”, *Proc. 17th Mediterranean Conference on Control and Automation*, Thessaloniki, Greece, 2009, pp. 1535–1539.
- [69] M. Darouach and L. Boutat-Baddas, “Observers for a Class of Nonlinear Singular Systems”, *IEEE Trans. Automat. Contr.*, 53 (2008), pp. 2627–2633.
- [70] M. Darouach and M. Boutayeb, “Design of Observers for Descriptor Systems”, *IEEE Trans. Automat. Contr.*, 40 (1995), pp. 1323–1327.
- [71] M. Darouach, M. Zasadzinski, and M. Hayar, “Reduced-Order Observer Design for Descriptor Systems with Unknown Inputs”, *IEEE Trans. Automat. Contr.*, 41 (1996), pp. 1068–1072.
- [72] P. Deuflhard, E. Hairer, and J. Zugck, “One-step and Extrapolation Methods for Differential-Algebraic Systems”, *Numer. Math.*, 51 (1987), pp. 501–516.
- [73] L. Dieci and T. Eirola, “On Smooth Decompositions of Matrices”, *SIAM J. Matrix Anal. Appl.*, 20 (1999), pp. 800–819.
- [74] L. Dieci, M. G. Gasparo, and A. Papini, “Continuation of Singular Value Decompositions”, *Mediterr. J. Math.*, 2 (2005), pp. 179–203.
- [75] Z. Ding, “Differential Stability and Design of Reduced Order Observers for Nonlinear Systems”, *Proc. 2009 IEEE International Conference on Control and Automation*, Christchurch, New Zealand, 2009, pp. 1104–1109.
- [76] M. El-Tohami, V. Lovass-Nagy, and R. Mukundan, “On the design of observers for generalized state space systems using singular value decomposition”, *Int. J. Control*, 38 (1983), pp. 673–683.
- [77] J.-C. Evard, “On the Existence of Bases of Class C^p of the Kernel and the Image of a Matrix Function”, *Lin. Alg. Appl.*, 135 (1990), pp. 33–67.

- [78] M. M. Fahmy and J. O'Reilly, "Observers for descriptor systems", *Int. J. Control.*, 49 (1989), pp. 2013–2028.
- [79] B. Friedland, *Advanced Control System Design*, Prentice Hall, Englewood Cliffs, NJ, 1996.
- [80] B. Friedland, "A Nonlinear Observer for Estimating Parameters in Dynamic Systems", *Automatica*, 33 (1997), pp. 1525–1530.
- [81] C. Führer and B. J. Leimkuhler, "Numerical solution of differential-algebraic equations for constrained mechanical motion", *Numer. Math.*, 59 (1991), pp. 55–69.
- [82] F. R. Gantmacher, *The Theory of Matrices*, Vol. 2, AMS Chelsea Publishing, Providence, RI, 2000.
- [83] G. Garcia, P. L. D. Peres, and S. Tarbouriech, "Necessary and sufficient numerical conditions for asymptotic stability of linear time-varying systems", *Proc. 47th IEEE Conference on Decision and Control*, Cancun, Mexico, 2008, pp. 5146–5151.
- [84] C. W. Gear, "Simultaneous Numerical Solution of Differential-Algebraic Equations", *IEEE Trans. Circ. Theor.*, CT-18 (1971), pp. 89–95.
- [85] C. W. Gear and L. R. Petzold, "Differential/Algebraic Systems and Matrix Pencils", *Matrix Pencils*, Springer-Verlag, Berlin, 1983, pp. 75–89.
- [86] C. W. Gear and L. R. Petzold, "ODE Methods for the Solution of Differential/Algebraic Systems", *SIAM J. Numer. Anal.*, 21 (1984), pp. 716–728.
- [87] G. H. Golub and C. F. Van Loan, *Matrix Computations*, The Johns Hopkins University Press, Baltimore, 1983.
- [88] B. Gopinath, "On the Control of Linear Multiple Input-Output Systems", *The Bell System Technical Journal*, 50 (1971), pp. 1063–1081.
- [89] W. Govaerts and B. Werner, "Continuous bordering of matrices and continuous matrix decompositions", *Numer. Math.*, 70 (1995), pp. 303–310.
- [90] D. A. Haessig and B. Friedland, "Nonlinear Reduced-Order State and Parameter Observer", *Proc. American Control Conference*, Arlington, VA, 2001, pp. 1978–1980.
- [91] M. Hanke, "On a Least-Squares Collocation Method for Linear Differential-Algebraic Equations", *Numer. Math.*, 54 (1988), pp. 79–90.
- [92] F. Hao, "Full-order Observer Design for Descriptor Systems with Delayed State and Unknown Inputs", *Proc. 25th Chinese Control Conference*, Harbin, P. R. China, 2006, pp. 765–770.
- [93] D. J. Hill and I. M. Y. Mareels, "Stability Theory for Differential/Algebraic Systems with Application to Power Systems", *IEEE Trans. Circ. Syst.*, 37 (1990), pp. 1416–1423.
- [94] M. Hou and P. C. Müller, "Observer Design for Descriptor Systems", *IEEE Trans. Automat. Contr.*, 44 (1999), pp. 164–168.

- [95] M. Hou, Th. Schmidt, R. Schüpphaus, and P. C. Müller, “Normal Form and Luenberger Observer for Linear Mechanical Descriptor Systems”, *J. Dyn. Sys., Meas., Contr.*, 115 (1993), pp. 611–620.
- [96] A. Ilchmann and M. Mueller, “Time-Varying Linear Systems: Relative Degree and Normal Form”, *IEEE Trans. Automat. Contr.*, 52 (2007), pp. 840–851.
- [97] D. Janovská, V. Janovský, and K. Tanabe, “An algorithm for computing the Analytic Singular Value Decomposition”, *World Academy of Science, Engineering and Technology*, 47 (2008), pp. 135–140.
- [98] G. W. Johnson, “A Deterministic Theory of Estimation and Control”, *IEEE Trans. Automat. Contr.*, 14 (1969), pp. 380–384.
- [99] D. Karagiannis, D. Carnevale, and A. Astolfi, “Invariant Manifold Based Reduced-Order Observer Design for Nonlinear Systems”, *IEEE Trans. Automat. Contr.*, 53 (2008), pp. 2602–2614.
- [100] B. Kisačanin and G. C. Agarwal, *Linear Control Systems: with solved problems and MATLAB examples*, Kluwer Academic/Plenum Publishers, New York, 2001.
- [101] D. L. Kleinman, *Suboptimal Design of Linear Regulator Systems Subject to Computer Storage Limitations*, PhD Dissertation, Massachusetts Institute of Technology, 1967.
- [102] M. Knorrenschild, “Differential/Algebraic Equations as Stiff Ordinary Differential Equations”, *SIAM J. Numer. Anal.*, 29 (1992), pp. 1694–1715.
- [103] D. Koenig, “Unknown Input Proportional Multiple-Integral Observer Design for Linear Descriptor Systems: Application to State and Fault Estimation”, *IEEE Trans. Automat. Contr.*, 50 (2005), pp. 212–217.
- [104] D. Koenig and S. Mammar, “Design of Proportional-Integral Observer for Unknown Input Descriptor Systems”, *IEEE Trans. Automat. Contr.*, 47 (2002), pp. 2057–2062.
- [105] P. Kunkel and V. Mehrmann, “Canonical forms for linear differential-algebraic equations with variable coefficients”, *J. Comput. Appl. Math.*, 56 (1994), pp. 225–251.
- [106] P. Kunkel and V. Mehrmann, *Differential-Algebraic Equations: Analysis and Numerical Solution*, European Mathematical Society, Zürich, 2006.
- [107] P. Kunkel and V. Mehrmann, “A New Class of Discretization Methods for the Solution of Linear Differential-Algebraic Equations with Variable Coefficients”, *SIAM J. Numer. Anal.*, 33 (1996), pp. 1941–1961.
- [108] P. Kunkel and V. Mehrmann, “Regular solutions of nonlinear differential-algebraic equations and their numerical determination”, *Numer. Math.*, 79 (1998), pp. 581–600.
- [109] P. Kunkel and V. Mehrmann, “Smooth factorizations of matrix valued functions and their derivatives”, *Numer. Math.*, 60 (1991), pp. 115–131.

- [110] P. Kunkel and V. Mehrmann, “Stability Properties of Differential-Algebraic Equations and Spin-Stabilized Discretizations”, *Electronic Transactions on Numerical Analysis*, 26 (2007), pp. 385–420.
- [111] P. Kunkel, V. Mehrmann, W. Rath, and J. Weickert, “A new software package for linear differential-algebraic equations”, *SIAM J. Sci. Comput.*, 18 (1997), pp. 115–138.
- [112] R. Lamour, “Index Determination and Calculation of Consistent Initial Values for DAEs”, *Comput. Math. Appl.*, 50 (2005), pp. 1125–1140.
- [113] R. A. Layton, *Principles of Analytical System Dynamics*, Springer, New York, 1998.
- [114] V. C. LeFevre, “The Design and Implementation of a Time-Varying Observer and an EMA Controller for Linear Systems”, *Proc. Twenty-Ninth Southeastern Symposium on System Theory*, IEEE Conference Publications, 1997, pp. 297–301.
- [115] B. Leimkuhler, L. R. Petzold, and C. W. Gear, “Approximation Methods for the Consistent Initialization of Differential-Algebraic Equations”, *SIAM J. Numer. Anal.*, 28 (1991), pp. 205–226.
- [116] F. L. Lewis, “A Survey of Linear Singular Systems”, *Circuits Systems Signal Process*, 5 (1986), pp. 3–36.
- [117] X. Li and K. Zhou, “A time domain approach to robust fault detection of linear time-varying systems”, *Automatica*, 45 (2009), pp. 94–102.
- [118] C.-H. Lien, “An efficient method to design robust observer-based control of uncertain linear systems”, *Appl. Math. Comput.*, 158 (2004), pp. 29–44.
- [119] C.-S. Liu and H. Peng, “Inverse-Dynamics Based State and Disturbance Observers for Linear Time-Invariant Systems”, *J. Dyn. Sys., Meas., Contr.*, 124 (2002), pp. 375–381.
- [120] V. Lovass-Nagy, R. J. Miller, and R. Mukundan, “On the Application of Matrix Generalized Inverses to the Design of Observers for Time-Varying and Time-Invariant Linear Systems”, *IEEE Trans. Automat. Contr.*, AC-25 (1980), pp. 1213–1218.
- [121] D. G. Luenberger, “Observers for Multivariable Systems”, *IEEE Trans. Automat. Contr.*, AC-11 (1966), pp. 190–197.
- [122] D. G. Luenberger, “Observing the State of a Linear System”, *IEEE Trans. on Military Electronics*, 8 (1964), pp. 74–80.
- [123] D. G. Luenberger, “Time-Invariant Descriptor Systems”, *Automatica*, 14 (1978), pp. 473–480.
- [124] J. Luo, N. Pongratananukul, J. A. Abu-Qahouq, and I. Batarseh, “Time-varying Current Observer with Parameter Estimation for Multiphase Low-Voltage High-Current Voltage Regulator Modules”, *Proc. Eighteenth Annual IEEE Applied Power Electronics Conference and Exposition*, Miami Beach, FL, 2003, pp. 444–450.

- [125] R. März and R. Riaza, “Linear differential-algebraic equations with properly stated leading term: Regular points”, *J. Math. Anal. Appl.*, 323 (2006), pp. 1279–1299.
- [126] V. Mehrmann and W. Rath, “Numerical Methods for the Computation of Analytic Singular Value Decompositions”, *Electronic Transactions on Numerical Analysis*, 1 (1993), pp. 72–88.
- [127] P. H. Menold, R. Findeisen, and F. Allgöwer, “Finite time convergent observers for linear time-varying systems”, *Proc. 11th IEEE Mediterranean Conference on Control and Automation*, Rhodes, Greece, 2003.
- [128] C. Meyer, *Matrix and Analysis and Applied Linear Algebra*, SIAM, Philadelphia, 2000.
- [129] N. Minamide, N. Arii, and Y. Uetake, “Design of observers for descriptor systems using a descriptor standard form”, *Int. J. Control.*, 50 (1989), pp. 2141–2149.
- [130] R. W. Newcomb, “The Semistate Description of Nonlinear Time-Variable Circuits”, *IEEE Trans. Circ. Syst.*, CAS-28 (1981), pp. 62–71.
- [131] R. W. Newcomb and B. Dziurla, “Some Circuits and Systems Applications of Semistate Theory”, *Circuits Systems Signal Process*, 8 (1989), pp. 235–260.
- [132] C. C. Nguyen, “Design of reduced-order state estimators for linear time-varying multi-variable systems”, *Int. J. Control.*, 46 (1987), pp. 2113–2126.
- [133] C. Nguyen and T. N. Lee, “Design of a State Estimator for a Class of Time-Varying Multivariable Systems”, *IEEE Trans. Automat. Contr.*, AC-30 (1985), pp. 179–182.
- [134] R. Nikoukhah, S. L. Campbell, and F. Delebecque, “Observer Design for General Linear Time-invariant Systems”, *Automatica*, 34 (1998), pp. 575–583.
- [135] J. O'Reilly, *Observers for Linear Systems*, Academic Press, London, 1983.
- [136] J. O'Reilly and M. M. Newmann, “Minimal-order observer-estimators for continuous-time linear systems”, *Int. J. Control.*, 22 (1975), pp. 573–590.
- [137] I. Okay, *The Additional Dynamics of the Least Squares Completions of Linear Differential Algebraic Equations*, PhD Dissertation, Department of Mathematics, North Carolina State University, Raleigh, NC, 2008.
- [138] I. Okay, S. L. Campbell, and P. Kunkel, “The additional dynamics of least squares completions for linear differential algebraic equations”, *Lin. Alg. Appl.*, 425 (2007), pp. 471–485.
- [139] I. Okay, S. L. Campbell, and P. Kunkel, “Completions of implicitly defined linear time varying vector fields”, *Lin. Alg. Appl.*, 431 (2009), pp. 1422–1438.
- [140] M. Otter, “A Constructive Definition of the Perturbation Index”, *COSY Workshop on Mathematical Modeling of Complex Systems*, Lund, Sweden, 1996.

- [141] D. W. Pearson, M. J. Chapman, and D. N. Shields, “Partial Singular-Value Assignment in the Design of Robust Observers for Discrete-Time Descriptor Systems”, *IMA J. Math. Contr. Inform.*, 5 (1988), pp. 203–213.
- [142] L. R. Petzold, “A Description of DASSL: A Differential/Algebraic System Solver”, *Sandia Report*, Sandia National Laboratories, Albuquerque, NM, 1982.
- [143] L. Petzold, “Differential/Algebraic Equations are not ODEs”, *SIAM J. Sci. Stat. Comput.*, 3 (1982), pp. 367–384.
- [144] P. J. Rabier and W. C. Rheinboldt, “Classical and Generalized Solutions of Time-Dependent Linear Differential-Algebraic Equations”, *Lin. Alg. Appl.*, 245 (1996), pp. 259–293.
- [145] P. J. Rabier and W. C. Rheinboldt, *Nonholonomic Motion of Rigid Mechanical Systems from a DAE Viewpoint*, SIAM, Philadelphia, 2000.
- [146] G. Reissig, “The Index of the Standard Circuit Equations of Passive RLCTG-Networks Does Not Exceed 2”, *Proc. 1998 IEEE International Symposium on Circuits and Systems*, IEEE Conference Publications, 1998, pp. 419–422.
- [147] W. C. Rheinboldt, “On the Computation of Multi-Dimensional Solution Manifolds of Parametrized Equations”, *Numer. Math.*, 53 (1988), pp. 165–181.
- [148] R. Riaza, *Differential-Algebraic Systems. Analytical Aspects and Circuit Applications*, World Scientific, Hackensack, NJ, 2008.
- [149] C. E. Seal and A. R. Stubberud, “Canonical Forms for Multiple-Input Time-Variable Systems”, *IEEE Trans. Automat. Contr.*, 14 (1969), pp. 704–707.
- [150] B. Shafai and R. L. Carroll, “Design of a minimal-order observer for singular systems”, *Int. J. Control.*, 45 (1987), pp. 1075–1081.
- [151] B. Shafai and R. L. Carroll, “Design of Proportional-Integral Observer for Linear Time-Varying Multivariable Systems”, *Proc. 24th Conference on Decision and Control*, Ft. Lauderdale, FL, 1985, pp. 597–599.
- [152] B. Shafai and R. L. Carroll, “Minimal-Order Observer Designs for Linear Time-Varying Multivariable Systems”, *IEEE Trans. Automat. Contr.*, AC-31 (1986), pp. 757–761.
- [153] K.-C. Shin and P. T. Kabamba, “Observation and Estimation in Linear Descriptor Systems With Application to Constrained Dynamical Systems”, *J. Dyn. Sys., Meas., Contr.*, 110 (1988), pp. 255–265.
- [154] L. M. Silverman and H. E. Meadows, “Controllability and Observability in Time-Variable Linear Systems”, *J. SIAM Control*, 5 (1967), pp. 64–73.
- [155] R. F. Sincovec, A. M. Erisman, E. L. Yip, and M. A. Epton, “Analysis of Descriptor Systems Using Numerical Algorithms”, *IEEE Trans. Automat. Contr.*, AC-26 (1981), pp. 139–147.

- [156] G. Söderlind and L. Wang, “Evaluating numerical ODE/DAE methods, algorithms and software”, *J. Comput. Appl. Math.*, 185 (2006), pp. 244–260.
- [157] R. T. Stefani, B. Shahian, C. J. Savant, Jr., and G. H. Hostetter, *Design of Feedback Control Systems*, Oxford University Press, New York, 2002.
- [158] V. Sundarapandian, “Reduced order observer design for discrete-time nonlinear systems”, *Applied Mathematics Letters*, 19 (2006), pp. 1013–1018.
- [159] H.-M. Tai, “Equivalent characterizations of detectability and stabilizability for a class of linear time-varying systems”, *Systems & Control Letters*, 8 (1987), pp. 425–428.
- [160] H.-M. Tai, *Linear Time-Varying Systems: Algebraic Structure, System Properties and Control*, PhD Dissertation, Texas Tech University, 1987.
- [161] A. Tornambè, “Asymptotic observers for non-linear systems”, *Int. J. Systems Sci.*, 23 (1992), pp. 435–442.
- [162] J. Trumpf, “Observers for linear time-varying systems”, *Lin. Alg. Appl.*, 425 (2007), pp. 303–312.
- [163] P. Van Dooren, “Reduced order observers: A new algorithm and proof”, *Systems & Control Letters*, 4 (1984), pp. 243–251.
- [164] A. T. Vemuri, M. M. Polycarpou, and A. R. Cirić, “Fault Diagnosis of Differential-Algebraic Systems”, *IEEE Trans. Syst., Man, Cybern. - Part A: Systems and Humans*, 31 (2001), pp. 143–152.
- [165] M. H. Verhaegen and P. Van Dooren, “A reduced order observer for descriptor systems”, *Systems & Control Letters*, 8 (1986), pp. 29–37.
- [166] B. L. Walcott, M. J. Corless, and S. H. Źak, “Comparative study of non-linear state-observation techniques”, *Int. J. Control*, 45 (1987), pp. 2109–2132.
- [167] K. Wright, “Differential equations for the analytic singular value decomposition of a matrix”, *Numer. Math.*, 63 (1992), pp. 283–295.
- [168] K. Wright, “Numerical Solution of Differential Equations for the Analytic Singular Value Decomposition”, *Proc. First International Colloquium on Numerical Analysis*, Plovdiv, Bulgaria, 1992, pp. 131–140.
- [169] A.-G. Wu and G.-R. Duan, “Design of Generalized PI Observers for Descriptor Linear Systems”, *IEEE Trans. Circ. Syst. - I: Regular Papers*, 53 (2006), pp. 2828–2837.
- [170] M. Y. Wu, “A Note on Stability of Linear Time-Varying Systems”, *IEEE Trans. Automat. Contr.*, 19 (1974), p. 162.
- [171] A. Xu and Q. Zhang, “Fault Detection and Isolation Based on Adaptive Observers for Linear Time Varying Systems”, *Proc. 15th IFAC World Congress*, Barcelona, Spain, 2002.

- [172] C. Yang, Q. Zhang, and T. Chai, “Observer design for a class of nonlinear descriptor systems”, *Proc. Joint 48th IEEE Conference on Decision and Control and 28th Chinese Control Conference*, Shanghai, P. R. China, 2009, pp. 8232–8237.
- [173] K. D. Yeomans, *Initialization Issues in General Differential Algebraic Equation Integrators*, PhD Dissertation, Department of Mathematics, North Carolina State University, Raleigh, NC, 1997.
- [174] Y. Ö. Yüksel and J. J. Bongiorno, Jr., “Observers for Linear Multivariable Systems with Applications”, *IEEE Trans. Automat. Contr.*, AC-16 (1971), pp. 603–613.
- [175] A. Zemouche and M. Boutayeb, “Observers for Continuous-Time Lipschitz Nonlinear Systems. Analysis and Comparisons”, *Proc. 51st IEEE Conference on Decision and Control*, Maui, HI, 2012, pp. 4774–4779.
- [176] G. Zimmer and J. Meier, “On observing nonlinear descriptor systems”, *Systems & Control Letters*, 32 (1997), pp. 43–48.

APPENDICES

Appendix A

Smooth Decomposition

In this appendix

1. Section A.1 provides a detailed description of the method from [109] and [147] for finding a smooth decomposition of a linear time-varying matrix;
2. Section A.2 explains the process from [109] for determining first derivatives of the smooth orthonormal bases without differentiating a numerically computed quantity; and
3. Section A.3 mentions alternate decompositions of linear time-varying matrices.

A smooth decomposition of a smooth, constant rank matrix $S(t)$ produces smooth orthonormal bases for its four fundamental subspaces $R(S(t))$, $N(S(t)^T)$, $R(S(t)^T)$, and $N(S(t))$, which are shown to exist in [64], [77]. Please be aware the notation in this appendix is based on the notation in [109] but differs at times due to deviations in some derivations. Also, the method described in [147] is not listed as a smooth decomposition but as the result of a moving frame algorithm. Articles [109] and [147] include the smooth decomposition development for one fundamental subspace after which the others can be modeled, but Sections A.1 and A.2 develop all four.

A.1 Smooth Decomposition

For an $m \times n$ smooth matrix $S(t)$ with constant rank r , Section A.1 describes five steps for finding a smooth decomposition

$$S(t) = U(t) \begin{bmatrix} \Sigma(t) & 0 \\ 0 & 0 \end{bmatrix} V(t)^T = \begin{bmatrix} U_1(t) & U_2(t) \end{bmatrix} \begin{bmatrix} \Sigma(t) & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} V_1(t)^T \\ V_2(t)^T \end{bmatrix} \quad (\text{A.1})$$

such that $\Sigma(t)$ is nonsingular and $U(t)$, $V(t)$ are orthogonal matrices. Note, the smooth decomposition is found in a neighborhood of t^* , which allows for local rather than global assumptions.

STEP 1: Determine a decomposition

$$S(t) = \tilde{U}(t) \begin{bmatrix} \tilde{\Sigma}(t) & 0 \\ 0 & 0 \end{bmatrix} \tilde{V}(t)^T = \begin{bmatrix} \tilde{U}_1(t) & \tilde{U}_2(t) \end{bmatrix} \begin{bmatrix} \tilde{\Sigma}(t) & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \tilde{V}_1(t)^T \\ \tilde{V}_2(t)^T \end{bmatrix} \quad (\text{A.2})$$

such that $\tilde{U}(t)$, $\tilde{V}(t)$ are orthogonal matrices.

Decomposition (A.2) is not assumed smooth. In terms of the notation from [109], $\tilde{U}(t) = Q_1(t)$ and $\tilde{V}(t) = \Pi(t)Q_2(t)$, where $Q_1(t)$ and $Q_2(t)$ are orthogonal and $\Pi(t)$ is a permutation matrix. The columns of the $m \times r$ matrix $\tilde{U}_1(t)$ are an orthonormal basis for $R(S(t))$; the columns of the $m \times (m - 2)$ matrix $\tilde{U}_2(t)$ are an orthonormal basis for $N(S(t)^T)$; the columns of the $n \times r$ matrix $\tilde{V}_1(t)$ are an orthonormal basis for $R(S(t)^T)$; and the columns of the $n \times (n - 2)$ matrix $\tilde{V}_2(t)$ are an orthonormal basis for $N(S(t))$ [128]. Additionally, U^* , V^* represent $\tilde{U}(t)$, $\tilde{V}(t)$ evaluated at t^* :

$$\begin{aligned} \tilde{U}(t^*) &= \begin{bmatrix} \tilde{U}_1(t^*) & \tilde{U}_2(t^*) \end{bmatrix} = \begin{bmatrix} U_1^* & U_2^* \end{bmatrix} = U^*, \\ \tilde{V}(t^*) &= \begin{bmatrix} \tilde{V}_1(t^*) & \tilde{V}_2(t^*) \end{bmatrix} = \begin{bmatrix} V_1^* & V_2^* \end{bmatrix} = V^*. \end{aligned}$$

STEP 2: Construct

- $P_U(t) = \tilde{U}_1(t)\tilde{U}_1(t)^T$, an orthogonal projector onto $R(S(t))$;
- $Q_U(t) = \tilde{U}_2(t)\tilde{U}_2(t)^T = I - P_U(t)$, the complementary orthogonal projector of $P_U(t)$;
- $P_V(t) = \tilde{V}_1(t)\tilde{V}_1(t)^T$, an orthogonal projector onto $R(S(t)^T)$; and
- $Q_V(t) = \tilde{V}_2(t)\tilde{V}_2(t)^T = I - P_V(t)$, the complementary orthogonal projector of $P_V(t)$.

Orthogonal projectors $P_U(t)$ and $P_V(t)$ can also be defined in terms of $S(t)$ and its Moore-Penrose pseudoinverse $S(t)^\dagger$. For example,

$$\begin{aligned} P_U(t) &= \tilde{U}_1(t)\tilde{U}_1(t)^T \\ &= \tilde{U}_1(t)\tilde{\Sigma}(t)\tilde{\Sigma}(t)^{-1}\tilde{U}_1(t)^T \\ &= \tilde{U}_1(t)\tilde{\Sigma}(t)\tilde{V}_1(t)^T\tilde{V}_1(t)\tilde{\Sigma}(t)^{-1}\tilde{U}_1(t)^T \\ &= \begin{bmatrix} \tilde{U}_1(t) & \tilde{U}_2(t) \end{bmatrix} \begin{bmatrix} \tilde{\Sigma}(t) & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \tilde{V}_1(t)^T \\ \tilde{V}_2(t)^T \end{bmatrix} \begin{bmatrix} \tilde{V}_1(t) & \tilde{V}_2(t) \end{bmatrix} \begin{bmatrix} \tilde{\Sigma}(t)^{-1} & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \tilde{U}_1(t)^T \\ \tilde{U}_2(t)^T \end{bmatrix} \\ &= \left(\tilde{U}(t) \begin{bmatrix} \tilde{\Sigma}(t) & 0 \\ 0 & 0 \end{bmatrix} \tilde{V}(t)^T \right) \left(\tilde{V}(t) \begin{bmatrix} \tilde{\Sigma}(t)^{-1} & 0 \\ 0 & 0 \end{bmatrix} \tilde{U}(t)^T \right) \\ &= S(t)S(t)^\dagger. \end{aligned}$$

Similarly,

$$\begin{aligned}
P_V(t) &= \tilde{V}_1(t)\tilde{V}_1(t)^T \\
&= \tilde{V}_1(t)\tilde{\Sigma}(t)^{-1}\tilde{\Sigma}(t)\tilde{V}_1(t)^T \\
&= \tilde{V}_1(t)\tilde{\Sigma}(t)^{-1}\tilde{U}_1(t)^T\tilde{U}_1(t)\tilde{\Sigma}(t)\tilde{V}_1(t)^T \\
&= \begin{bmatrix} \tilde{V}_1(t) & \tilde{V}_2(t) \end{bmatrix} \begin{bmatrix} \tilde{\Sigma}(t)^{-1} & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \tilde{U}_1(t)^T \\ \tilde{U}_2(t)^T \end{bmatrix} \begin{bmatrix} \tilde{U}_1(t) & \tilde{U}_2(t) \end{bmatrix} \begin{bmatrix} \tilde{\Sigma}(t) & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \tilde{V}_1(t)^T \\ \tilde{V}_2(t)^T \end{bmatrix} \\
&= \left(\tilde{V}(t) \begin{bmatrix} \tilde{\Sigma}(t)^{-1} & 0 \\ 0 & 0 \end{bmatrix} \tilde{U}(t)^T \right) \left(\tilde{U}(t) \begin{bmatrix} \tilde{\Sigma}(t) & 0 \\ 0 & 0 \end{bmatrix} \tilde{V}(t)^T \right) \\
&= S(t)^\dagger S(t).
\end{aligned}$$

Due to the assumptions that $S(t)$ is smooth and has constant rank, $S(t)^\dagger$ is also smooth so projectors $P_U(t)$, $Q_U(t)$, $P_V(t)$, and $Q_V(t)$ are as smooth as $S(t)$ and $S(t)^\dagger$ [77].

STEP 3: Solve the orthogonal Procrustes problem for U_1^* and $\tilde{U}_1(t)$; for U_2^* and $\tilde{U}_2(t)$; for V_1^* and $\tilde{V}_1(t)$; and for V_2^* and $\tilde{V}_2(t)$. In the following discussion: U_i^* and $\tilde{U}_i(t)$, V_i^* and $\tilde{V}_i(t)$ for $i = 1, 2$.

In general notation, the orthogonal Procrustes problem determines an orthogonal matrix Q that minimizes $\|A - BQ\|_F$. An algorithm for finding the matrix that minimizes the norm is presented in [87]. $\|U_i^* - \tilde{U}_i(t)Z_{Ui}(t)\|_F$ is minimized if $Z_{Ui}(t) = \hat{U}_i(t)\hat{V}_i(t)^T$ for $\hat{U}_i(t)$, $\hat{V}_i(t)$ from the singular value decomposition $\tilde{U}_i(t)^T U_i^* = \hat{U}_i(t)\hat{\Sigma}_i(t)\hat{V}_i(t)^T$. Likewise, $Z_{Vi}(t) = \bar{U}_i(t)\bar{V}_i(t)^T$ minimizes $\|V_i^* - \tilde{V}_i(t)Z_{Vi}(t)\|_F$ for $\bar{U}_i(t)$, $\bar{V}_i(t)$ from the singular value decomposition $\tilde{V}_i(t)^T V_i^* = \bar{U}_i(t)\bar{\Sigma}_i(t)\bar{V}_i(t)^T$.

STEP 4: Determine the polar decompositions of $\tilde{U}_1(t)^T U_1^*$, of $\tilde{U}_2(t)^T U_2^*$, of $\tilde{V}_1(t)^T V_1^*$, and of $\tilde{V}_2(t)^T V_2^*$. In the following discussion: $\tilde{U}_i(t)^T U_i^*$, $\tilde{V}_i(t)^T V_i^*$ for $i = 1, 2$.

As described in [87], the polar decomposition of a real matrix A is $A = \Phi\Psi$ such that $\Phi^T\Phi = I$ and Ψ is symmetric and positive semi-definite. A polar decomposition of $\tilde{U}_i(t)^T U_i^*$ can be derived from its singular value decomposition:

$$\begin{aligned}
\tilde{U}_i(t)^T U_i^* &= \hat{U}_i(t)\hat{\Sigma}_i(t)\hat{V}_i(t)^T \\
&= \hat{U}_i(t)\hat{V}_i(t)^T \hat{V}_i(t)\hat{\Sigma}_i(t)\hat{V}_i(t)^T \\
&= Z_{Ui}(t)H_{Ui}(t).
\end{aligned}$$

The Φ and Ψ matrices of this polar decomposition are $Z_{Ui}(t)$ and $H_{Ui}(t)$, respectively. Defined in Step 3, matrix $Z_{Ui}(t)$ is orthogonal so $Z_{Ui}(t)^T Z_{Ui}(t) = I$. Matrix $H_{Ui}(t) = \hat{V}_i(t)\hat{\Sigma}_i(t)\hat{V}_i(t)^T$

is symmetric because

$$H_{Ui}(t)^T = \left(\widehat{V}_i(t) \widehat{\Sigma}_i(t) \widehat{V}_i(t)^T \right)^T = \widehat{V}_i(t) \widehat{\Sigma}_i(t)^T \widehat{V}_i(t)^T = \widehat{V}_i(t) \widehat{\Sigma}_i(t) \widehat{V}_i(t)^T = H_{Ui}(t).$$

Furthermore, matrix $H_{Ui}(t)$ can be factored

$$H_{Ui}(t) = \widehat{V}_i(t) \widehat{\Sigma}_i(t) \widehat{V}_i(t)^T = \widehat{V}_i(t) \widehat{\Sigma}_i(t)^{1/2} \widehat{\Sigma}_i(t)^{1/2} \widehat{V}_i(t)^T = B_{Ui}(t)^T B_{Ui}(t)$$

such that orthogonal $\widehat{V}_i(t)$ and the positive entries of the diagonal matrix $\widehat{\Sigma}_i(t)$ make $B_{Ui}(t) = \widehat{\Sigma}_i(t)^{1/2} \widehat{V}_i(t)^T$ nonsingular. Thus, matrix $H_{Ui}(t)$ is positive definite (positive semi-definite with no zero eigenvalues) [128]. The manipulation from the singular value decomposition $\widetilde{V}_i(t)^T V_i^* = \overline{U}_i(t) \overline{\Sigma}_i(t) \overline{V}_i(t)^T$ to its polar decomposition produces $\widetilde{V}_i(t)^T V_i^* = Z_{Vi}(t) H_{Vi}(t)$ for $Z_{Vi}(t)$ defined in Step 3 and $H_{Vi}(t) = \overline{V}_i(t) \overline{\Sigma}_i(t) \overline{V}_i(t)^T$.

In addition to being symmetric and positive definite, $H_{Ui}(t)$ and $H_{Vi}(t)$ are smooth. For example,

$$\begin{aligned} H_{U1}(t)^2 &= H_{U1}(t)^T H_{U1}(t) \\ &= H_{U1}(t)^T Z_{U1}(t)^T Z_{U1}(t) H_{U1}(t) \\ &= (Z_{U1}(t) H_{U1}(t))^T (Z_{U1}(t) H_{U1}(t)) \\ &= (\widetilde{U}_1(t)^T U_1^*)^T (\widetilde{U}_1(t)^T U_1^*) \\ &= (U_1^*)^T \widetilde{U}_1(t) \widetilde{U}_1(t)^T U_1^* \\ &= (U_1^*)^T P_U(t) U_1^* \end{aligned}$$

reveals $H_{U1}(t)$ can also be defined in terms of constant matrix U_1^* and orthogonal projector $P_U(t)$. Therefore, $H_{U1}(t) = ((U_1^*)^T P_U(t) U_1^*)^{1/2}$ is as smooth as $P_U(t)$. Similarly, $H_{U2}(t)$ is as smooth as $Q_U(t)$, $H_{V1}(t)$ is as smooth as $P_V(t)$, and $H_{V2}(t)$ is as smooth as $Q_V(t)$.

STEP 5: Define $U_1(t) = \widetilde{U}_1(t) Z_{U1}(t)$, $U_2(t) = \widetilde{U}_2(t) Z_{U2}(t)$, $V_1(t) = \widetilde{V}_1(t) Z_{V1}(t)$, and $V_2(t) = \widetilde{V}_2(t) Z_{V2}(t)$ as the matrices with columns forming smooth orthonormal bases for the desired smooth decomposition. In the following discussion: $U_i(t) = \widetilde{U}_i(t) Z_{Ui}(t)$, $V_i(t) = \widetilde{V}_i(t) Z_{Vi}(t)$ for $i = 1, 2$.

These definitions of $U_i(t)$ and $V_i(t)$ come from the polar decompositions of $\widetilde{U}_i(t)^T U_i^*$ and $\widetilde{V}_i(t)^T V_i^*$, respectively. $H_{Ui}(t)$ and $H_{Vi}(t)$ are invertible from being positive definite, but [109] specifies $H_{Ui}(t)$ and $H_{Vi}(t)$ are nonsingular in a neighborhood of t^* because

$$H_{Ui}(t^*) = ((U_i^*)^T U_i^* (U_i^*)^T U_i^*)^{1/2} = I \quad \text{and} \quad H_{Vi}(t^*) = ((V_i^*)^T V_i^* (V_i^*)^T V_i^*)^{1/2} = I.$$

Looking at $\tilde{U}_i(t)^T U_i^* = Z_{Ui}(t) H_{Ui}(t)$ in a neighborhood of t^* ,

$$Z_{Ui}(t) = \tilde{U}_i(t)^T U_i^* H_{Ui}(t)^{-1}. \quad (\text{A.3})$$

Multiplying equation (A.3) on the left by $\tilde{U}_i(t)$ results in

$$\tilde{U}_1(t) Z_{U1}(t) = P_U(t) U_1^* H_{U1}(t)^{-1} \quad \text{and} \quad \tilde{U}_2(t) Z_{U2}(t) = Q_U(t) U_2^* H_{U2}(t)^{-1}$$

after substituting in the orthogonal projectors for $\tilde{U}_i(t) \tilde{U}_i(t)^T$. Since the columns of $\tilde{U}_1(t)$ form an orthonormal basis for $R(S(t))$ and $Z_{U1}(t)$ is behaving as an orthogonal transformation matrix, the columns of matrix $\tilde{U}_1(t) Z_{U1}(t)$ also form an orthonormal basis for $R(S(t))$. Like the columns of matrix $\tilde{U}_2(t)$, the columns of matrix $\tilde{U}_2(t) Z_{U2}(t)$ form an orthonormal basis for $N(S(t)^T)$. Matrix $\tilde{U}_i(t) Z_{Ui}(t)$ is smooth because U_i^* is constant and both the projector and matrix $H_{Ui}(t)$ are smooth.

For the polar decomposition $\tilde{V}_i(t)^T V_i^* = Z_{Vi}(t) H_{Vi}(t)$ in a neighborhood of t^* ,

$$\tilde{V}_1(t) Z_{V1}(t) = P_V(t) V_1^* H_{V1}(t)^{-1} \quad \text{and} \quad \tilde{V}_2(t) Z_{V2}(t) = Q_V(t) V_2^* H_{V2}(t)^{-1}.$$

The columns of $V_1(t) = \tilde{V}_1(t) Z_{V1}(t)$ form an orthonormal basis for $R(S(t)^T)$, while the columns of $V_2(t) = \tilde{V}_2(t) Z_{V2}(t)$ form an orthonormal basis for $N(S(t))$. Matrices $V_1(t)$ and $V_2(t)$ are smooth because V_i^* is constant, the orthogonal projectors $P_V(t)$ and $Q_V(t)$ are smooth, and $H_{Vi}(t)$ is smooth.

A.2 First Derivatives

The following discussion outlines the steps for finding the first derivatives of smooth decomposition matrices $U_i(t)$ and $V_i(t)$ for $i = 1, 2$ in terms of matrices with known derivatives. From Step 5 of Section A.1, $U_1(t) = \tilde{U}_1(t) Z_{U1}(t) = P_U(t) U_1^* H_{U1}(t)^{-1}$. In a neighborhood of t^* ,

$$U_1(t) H_{U1}(t) = P_U(t) U_1^*. \quad (\text{A.4})$$

The first derivative of equation (A.4) reveals an expression for $\dot{U}_1(t)$:

$$\begin{aligned} \dot{U}_1(t) H_{U1}(t) + U_1(t) \dot{H}_{U1}(t) &= \dot{P}_U(t) U_1^* \\ \implies \dot{U}_1(t) &= \left(\dot{P}_U(t) U_1^* - U_1(t) \dot{H}_{U1}(t) \right) H_{U1}(t)^{-1}. \end{aligned}$$

Repeating this process for smooth decomposition matrices $U_2(t)$, $V_1(t)$, and $V_2(t)$ further reveals

$$\begin{aligned}\dot{U}_2(t) &= -\left(\dot{P}_U(t)U_2^* + U_2(t)\dot{H}_{U2}(t)\right)H_{U2}(t)^{-1}, \\ \dot{V}_1(t) &= \left(\dot{P}_V(t)V_1^* - V_1(t)\dot{H}_{V1}(t)\right)H_{V1}(t)^{-1}, \\ \dot{V}_2(t) &= -\left(\dot{P}_V(t)V_2^* + V_2(t)\dot{H}_{V2}(t)\right)H_{V2}(t)^{-1},\end{aligned}$$

since $Q_U(t) = I - P_U(t)$ and $Q_V(t) = I - P_V(t)$. The first derivatives of projectors $P_U(t)$ and $P_V(t)$ and of matrices $H_{Ui}(t)$ and $H_{Vi}(t)$ are known without differentiating a numerically computed quantity.

STEP 1: Determine the first derivatives of orthogonal projectors $P_U(t)$ and $P_V(t)$.

From Step 2 in Section A.1, $P_U(t) = S(t)S(t)^\dagger$ and $P_V(t) = S(t)^\dagger S(t)$; from the properties of the Moore-Penrose pseudoinverse, $P_U(t)S(t) = S(t)S(t)^\dagger S(t) = S(t)$ and $S(t)P_V(t) = S(t)S(t)^\dagger S(t) = S(t)$ [128]; and from the properties of orthogonal projectors, $P_U(t)^2 = P_U(t) = P_U(t)^T$ and $P_V(t)^2 = P_V(t) = P_V(t)^T$ [128].

The first derivatives of $P_U(t)S(t) = S(t)$ and $S(t)P_V(t) = S(t)$ are

$$\dot{P}_U(t)S(t) + P_U(t)\dot{S}(t) = \dot{S}(t), \quad (\text{A.5a})$$

$$\dot{S}(t)P_V(t) + S(t)\dot{P}_V(t) = \dot{S}(t). \quad (\text{A.5b})$$

Multiplying equations (A.5a) and (A.5b) by $S(t)^\dagger$ on the right and left, respectively, and then substituting in the appropriate projectors for $S(t)S(t)^\dagger$ and $S(t)^\dagger S(t)$ gives

$$\begin{aligned}\dot{P}_U(t)P_U(t) + P_U(t)\dot{S}(t)S(t)^\dagger &= \dot{S}(t)S(t)^\dagger \\ \implies \dot{P}_U(t)P_U(t) &= (I - P_U(t))\dot{S}(t)S(t)^\dagger,\end{aligned} \quad (\text{A.6a})$$

$$\begin{aligned}S(t)^\dagger\dot{S}(t)P_V(t) + P_V(t)\dot{P}_V(t) &= S(t)^\dagger\dot{S}(t) \\ \implies P_V(t)\dot{P}_V(t) &= S(t)^\dagger\dot{S}(t)(I - P_V(t)).\end{aligned} \quad (\text{A.6b})$$

Matrices $\dot{P}_U(t)$ and $\dot{P}_V(t)$ are symmetric since $P_U(t) = P_U(t)^T$ and $P_V(t) = P_V(t)^T$ from the properties of orthogonal projectors, so the transposes of equations (A.6a) and (A.6b) are

$$P_U(t)^T\dot{P}_U(t)^T = P_U(t)\dot{P}_U(t) = \left((I - P_U(t))\dot{S}(t)S(t)^\dagger\right)^T,$$

$$\dot{P}_V(t)^T P_V(t)^T = \dot{P}_V(t)P_V(t) = \left(S(t)^\dagger\dot{S}(t)(I - P_V(t))\right)^T.$$

Therefore, the first derivatives of $P_U(t) = P_U(t)P_U(t)$ and $P_V(t) = P_V(t)P_V(t)$ are

$$\begin{aligned}\dot{P}_U(t) &= \dot{P}_U(t)P_U(t) + P_U(t)\dot{P}_U(t) \\ &= (I - P_U(t))\dot{S}(t)S(t)^\dagger + ((I - P_U(t))\dot{S}(t)S(t)^\dagger)^T,\end{aligned}\quad (\text{A.7a})$$

$$\begin{aligned}\dot{P}_V(t) &= \dot{P}_V(t)P_V(t) + P_V(t)\dot{P}_V(t) \\ &= (S(t)^\dagger\dot{S}(t)(I - P_V(t)))^T + S(t)^\dagger\dot{S}(t)(I - P_V(t)).\end{aligned}\quad (\text{A.7b})$$

From [51], since $S(t)$ is assumed smooth with a constant rank, the first derivative of the Moore-Penrose pseudoinverse of $S(t)$ is

$$\begin{aligned}(S(t)^\dagger)' &= -S(t)^\dagger\dot{S}(t)S(t)^\dagger + (I - S(t)^\dagger S(t))\dot{S}(t)^T(S(t)^\dagger)^T S(t)^\dagger \\ &\quad + S(t)^\dagger(S(t)^\dagger)^T\dot{S}(t)^T(I - S(t)S(t)^\dagger).\end{aligned}$$

Knowing this derivative, an alternate method for determining the first derivatives of projectors $P_U(t)$ and $P_V(t)$ is to consider

$$\begin{aligned}\dot{P}_U(t) &= \dot{S}(t)S(t)^\dagger + S(t)(S(t)^\dagger)', \\ \dot{P}_V(t) &= (S(t)^\dagger)'S(t) + S(t)^\dagger\dot{S}(t),\end{aligned}$$

and then substitute in for $(S(t)^\dagger)'$. The resulting expression can be simplified by employing the properties of the Moore-Penrose pseudoinverse and of orthogonal projectors. For example,

$$\begin{aligned}\dot{P}_U(t) &= \dot{S}(t)S(t)^\dagger + S(t)(S(t)^\dagger)' \\ &= \dot{S}(t)S(t)^\dagger - S(t)S(t)^\dagger\dot{S}(t)S(t)^\dagger + S(t)\dot{S}(t)^T(S(t)^\dagger)^TS(t)^\dagger \\ &\quad - S(t)S(t)^\dagger S(t)\dot{S}(t)^T(S(t)^\dagger)^TS(t)^\dagger + S(t)S(t)^\dagger(S(t)^\dagger)^T\dot{S}(t)^T \\ &\quad - S(t)S(t)^\dagger(S(t)^\dagger)^T\dot{S}(t)^TS(t)S(t)^\dagger \\ &= \dot{S}(t)S(t)^\dagger - P_U(t)\dot{S}(t)S(t)^\dagger + S(t)\dot{S}(t)^T(S(t)^\dagger)^TS(t)^\dagger - S(t)\dot{S}(t)^T(S(t)^\dagger)^TS(t)^\dagger \\ &\quad + P_U(t)(S(t)^\dagger)^T\dot{S}(t)^T - P_U(t)(S(t)^\dagger)^T\dot{S}(t)^TP_U(t) \\ &= (I - P_U(t))\dot{S}(t)S(t)^\dagger + P_U(t)(S(t)^\dagger)^T\dot{S}(t)^T(I - P_U(t)) \\ &= (I - P_U(t))\dot{S}(t)S(t)^\dagger + ((I - P_U(t))\dot{S}(t)S(t)^\dagger S(t)S(t)^\dagger)^T \\ &= (I - P_U(t))\dot{S}(t)S(t)^\dagger + ((I - P_U(t))\dot{S}(t)S(t)^\dagger)^T.\end{aligned}$$

STEP 2: Determine the first derivatives of $H_{U1}(t)$, of $H_{U2}(t)$, of $H_{V1}(t)$, and of $H_{V2}(t)$. In the following discussion: $H_{Ui}(t)$, $H_{Vi}(t)$ for $i = 1, 2$.

From Step 4 in Section A.1, $H_{Ui}(t) = \widehat{V}_i(t)\widehat{\Sigma}_i(t)\widehat{V}_i(t)^T$, $H_{Vi}(t) = \overline{V}_i(t)\overline{\Sigma}_i(t)\overline{V}_i(t)^T$, and

$$\begin{aligned} H_{U1}(t)^2 &= (U_1^*)^T P_U(t) U_1^*, \\ H_{U2}(t)^2 &= (U_2^*)^T Q_U(t) U_2^* = (U_2^*)^T (I - P_U(t)) U_2^*, \\ H_{V1}(t)^2 &= (V_1^*)^T P_V(t) V_1^*, \\ H_{V2}(t)^2 &= (V_2^*)^T Q_V(t) V_2^* = (V_2^*)^T (I - P_V(t)) V_2^*. \end{aligned}$$

The first derivatives of $H_{Ui}(t)^2$ and $H_{Vi}(t)^2$ are

$$\begin{aligned} \dot{H}_{U1}(t)H_{U1}(t) + H_{U1}(t)\dot{H}_{U1}(t) &= (U_1^*)^T \dot{P}_U(t) U_1^*, \\ \dot{H}_{U2}(t)H_{U2}(t) + H_{U2}(t)\dot{H}_{U2}(t) &= -(U_2^*)^T \dot{P}_U(t) U_2^*, \\ \dot{H}_{V1}(t)H_{V1}(t) + H_{V1}(t)\dot{H}_{V1}(t) &= (V_1^*)^T \dot{P}_V(t) V_1^*, \\ \dot{H}_{V2}(t)H_{V2}(t) + H_{V2}(t)\dot{H}_{V2}(t) &= -(V_2^*)^T \dot{P}_V(t) V_2^*, \end{aligned}$$

which can be generalized as

$$\dot{H}_{Ui}(t)H_{Ui}(t) + H_{Ui}(t)\dot{H}_{Ui}(t) = (-1)^{i+1} (U_i^*)^T \dot{P}_U(t) U_i^*, \quad (\text{A.8a})$$

$$\dot{H}_{Vi}(t)H_{Vi}(t) + H_{Vi}(t)\dot{H}_{Vi}(t) = (-1)^{i+1} (V_i^*)^T \dot{P}_V(t) V_i^*. \quad (\text{A.8b})$$

After substituting $\widehat{V}_i(t)\widehat{\Sigma}_i(t)\widehat{V}_i(t)^T$ into equation (A.8a) and $\overline{V}_i(t)\overline{\Sigma}_i(t)\overline{V}_i(t)^T$ into equation (A.8b) for $H_{Ui}(t)$ and $H_{Vi}(t)$, respectively,

$$\dot{H}_{Ui}(t)\widehat{V}_i(t)\widehat{\Sigma}_i(t)\widehat{V}_i(t)^T + \widehat{V}_i(t)\widehat{\Sigma}_i(t)\widehat{V}_i(t)^T \dot{H}_{Ui}(t) = (-1)^{i+1} (U_i^*)^T \dot{P}_U(t) U_i^*, \quad (\text{A.9a})$$

$$\dot{H}_{Vi}(t)\overline{V}_i(t)\overline{\Sigma}_i(t)\overline{V}_i(t)^T + \overline{V}_i(t)\overline{\Sigma}_i(t)\overline{V}_i(t)^T \dot{H}_{Vi}(t) = (-1)^{i+1} (V_i^*)^T \dot{P}_V(t) V_i^*. \quad (\text{A.9b})$$

Multiplying equation (A.9a) on the left by $\widehat{V}_i(t)^T$ and on the right by $\widehat{V}_i(t)$ gives

$$\widehat{V}_i(t)^T \dot{H}_{Ui}(t)\widehat{V}_i(t)\widehat{\Sigma}_i(t) + \widehat{\Sigma}_i(t)\widehat{V}_i(t)^T \dot{H}_{Ui}(t)\widehat{V}_i(t) = (-1)^{i+1} \widehat{V}_i(t)^T (U_i^*)^T \dot{P}_U(t) U_i^* \widehat{V}_i(t),$$

while multiplying equation (A.9b) on the left by $\overline{V}_i(t)^T$ and on the right by $\overline{V}_i(t)$ produces

$$\overline{V}_i(t)^T \dot{H}_{Vi}(t)\overline{V}_i(t)\overline{\Sigma}_i(t) + \overline{\Sigma}_i(t)\overline{V}_i(t)^T \dot{H}_{Vi}(t)\overline{V}_i(t) = (-1)^{i+1} \overline{V}_i(t)^T (V_i^*)^T \dot{P}_V(t) V_i^* \overline{V}_i(t),$$

since $\widehat{V}_i(t)^T \widehat{V}_i(t) = I$ and $\overline{V}_i(t)^T \overline{V}_i(t) = I$. Therefore, the first derivatives of $H_{Ui}(t)$ and $H_{Vi}(t)$

should be defined as

$$\begin{aligned}\dot{H}_{Ui}(t) &= \widehat{V}_i(t)G_{Ui}(t)\widehat{V}_i(t)^T, \\ \dot{H}_{Vi}(t) &= \overline{V}_i(t)G_{Vi}(t)\overline{V}_i(t)^T,\end{aligned}$$

where $G_{Ui}(t)$ and $G_{Vi}(t)$ satisfy

$$\begin{aligned}G_{Ui}(t)\widehat{\Sigma}_i(t) + \widehat{\Sigma}_i(t)G_{Ui}(t) &= (-1)^{i+1}\widehat{V}_i(t)^T(U_i^*)^T\dot{P}_U(t)U_i^*\widehat{V}_i(t), \\ G_{Vi}(t)\overline{\Sigma}_i(t) + \overline{\Sigma}_i(t)G_{Vi}(t) &= (-1)^{i+1}\overline{V}_i(t)^T(V_i^*)^T\dot{P}_V(t)V_i^*\overline{V}_i(t).\end{aligned}$$

To see how $G_{U1}(t)$ should be defined, for example, consider the following 2×2 case (the time dependence of $g_{ij}(t)$ and $\sigma_i(t)$ is removed due to space considerations):

$$\begin{aligned}\widehat{V}_1(t)^T(U_1^*)^T\dot{P}_U(t)U_1^*\widehat{V}_1(t) &= G_{U1}(t)\widehat{\Sigma}_1(t) + \widehat{\Sigma}_1(t)G_{U1}(t) \\ &= \begin{bmatrix} g_{11} & g_{12} \\ g_{21} & g_{22} \end{bmatrix} \begin{bmatrix} \sigma_1 & 0 \\ 0 & \sigma_2 \end{bmatrix} + \begin{bmatrix} \sigma_1 & 0 \\ 0 & \sigma_2 \end{bmatrix} \begin{bmatrix} g_{11} & g_{12} \\ g_{21} & g_{22} \end{bmatrix} \\ &= \begin{bmatrix} g_{11}\sigma_1 & g_{12}\sigma_2 \\ g_{21}\sigma_1 & g_{22}\sigma_2 \end{bmatrix} + \begin{bmatrix} \sigma_1 g_{11} & \sigma_1 g_{12} \\ \sigma_2 g_{21} & \sigma_2 g_{22} \end{bmatrix} \\ &= \begin{bmatrix} g_{11}(\sigma_1 + \sigma_1) & g_{12}(\sigma_2 + \sigma_1) \\ g_{21}(\sigma_1 + \sigma_2) & g_{22}(\sigma_2 + \sigma_2) \end{bmatrix}.\end{aligned}$$

Confirmed by Lemma 2.22 (incorrectly listed as 2.2) in [109], the components of $G_{U1}(t) = [g_{ij}(t)]$ for $i, j = 1, \dots, r$ are calculated by

$$g_{ij}(t) = \frac{1}{\sigma_i(t) + \sigma_j(t)}\gamma_{ij}(t) \quad (\text{A.11})$$

with $\widehat{\Sigma}_1(t) = [\sigma_i(t)] = \text{diag}(\sigma_1(t), \dots, \sigma_r(t))$ and $\Gamma_{U1}(t) = [\gamma_{ij}(t)] = \widehat{V}_1(t)^T(U_1^*)^T\dot{P}_U(t)U_1^*\widehat{V}_1(t)$. Similarly, the i, j indices, $[\sigma_i(t)]$ elements, and $[\gamma_{ij}(t)]$ elements

$[g_{ij}(t)]$	i, j	$[\sigma_i(t)]$	$[\gamma_{ij}(t)]$
$G_{U2}(t)$	$1, \dots, m-r$	$\widehat{\Sigma}_2(t)$	$\Gamma_{U2}(t) = -\widehat{V}_2(t)^T(U_2^*)^T\dot{P}_U(t)U_2^*\widehat{V}_2(t)$
$G_{V1}(t)$	$1, \dots, r$	$\overline{\Sigma}_1(t)$	$\Gamma_{V1}(t) = \overline{V}_1(t)^T(V_1^*)^T\dot{P}_V(t)V_1^*\overline{V}_1(t)$
$G_{V2}(t)$	$1, \dots, n-r$	$\overline{\Sigma}_2(t)$	$\Gamma_{V2}(t) = -\overline{V}_2(t)^T(V_2^*)^T\dot{P}_V(t)V_2^*\overline{V}_2(t)$

are used with equation (A.11) to calculate the components of $G_{U2}(t)$, $G_{V1}(t)$, and $G_{V2}(t)$.

A.3 Alternatives to the Smooth Decomposition

The smooth decomposition is an example of an algebraic factorization technique. Its development uses algebraic equations and identities rather than ordinary differential equations to determine smooth orthogonal factors of a linear time-varying matrix. A similar decomposition can be found in [64].

An alternative to the smooth decomposition is the analytic singular value decomposition (ASVD) first developed as an algebraic factorization technique in [31]. One concern in [31] about the smooth decomposition is that it may not be smooth beyond a neighborhood of t^* . Instead, the ASVD produces smooth bases free from local assumptions. However, rather than requiring a smooth, constant rank matrix, the ASVD assumes the matrix-to-be-factored is a real analytic function, defined as a “[function] with [a] Taylor series that, in a neighborhood of each point, [converges] to the original function” ([31], pg. 4). Additionally, this decomposition relaxes the ordering and nonnegative conditions typically associated with the singular value matrix ($\Sigma(t)$ in (A.1), for example). The algebraic ASVD from [31] is modified in [126] to include an analytic polar decomposition during its construction.

Analytic singular value decompositions in [13] and [167], [168] use ordinary differential equations (ODEs) to determine orthogonal factors. The method in [167], [168] cannot be generalized due to dependence on the singular values and has the potential to be computationally inefficient because of the interpolations solving the ODEs. This efficiency concern is addressed a second time in [126] during a comparison of the approaches in [31] and [167], [168]. The authors of [13] designed a tracking algorithm that uses the solutions of ODEs to converge to the true ASVD factors of a square matrix. Other variations of the ASVD can be found in [89] and [97].

Ordinary differential equations are also implemented in [73] to determine smooth QR, Schur, and singular value decompositions. Article [73] notes a smooth matrix does not guarantee a smooth decomposition, which is one reason the constant rank assumption is important [64]. In addition to the efficiency concern from using ODEs, the authors of [74] remark the iterative process for calculating the factors does not return exact bases. To help reduce numerical error in an ODE ASVD, a method to stabilize the algorithm is proposed in [16].

The smooth decomposition was chosen for our research problem because it took advantage of the smooth and constant rank assumptions that already existed for the linear time-varying systems of DAEs being studied. Furthermore, [109] provided a clear outline of how the first derivatives of the smooth factors could be computed.

Appendix B

MATLAB Code

B.1 Chapter 5 Code

```
1 function J = index(E,F)
2
3 %Function index.m determines the index of a linear time-invariant system of
4 % DAEs, E(x(t)') + Fx(t) = Bu(t).
5 %Matrices E and F are input by the user.
6
7 %To determine the index
8 %1) First check that [scriptE scriptF] is full row rank.
9 %If [scriptE scriptF] is full row rank, then
10 %2) Calculate svd(scriptE),
11 %      [U,S,V] = svd(scriptE), where scriptE = U*S*(V^T).
12 %3) Calculate rS = rank(S) using program Rounding.m.
13 %4) The last (nscriptE - rS) columns of V are a basis for the nullspace of
14 %   scriptE (N(scriptE)).
15 %5) Calculate the norm of a subvector of each of the basis vectors; this
16 %   subvector is the first mE rows of the basis vector (zerocheck).
17 %6) Add the norms together (zerosum); if zerosum = 0, then that value of k
18 %   is the index.
19
20 %k = 0: scriptE = E, scriptF = F
21 %k = 1: scriptE = [E 0;F E], scriptF = [F;0]
22 %k = 2: scriptE = [E 0 0;F E 0;0 F E], scriptF = [F;0;0]
23 %k = 3: scriptE = [E 0 0 0;F E 0 0;0 F E 0;0 0 F E], scriptF = [F;0;0;0]
24 %etc.
25 %*****
```

```
26
27 %Initialize index k; 100 is arbitrary, but value should be larger than
```

```

28 % expected index
29 for k = 0:1:100
30 %Determine the size of matrix E to identify the number of state variables
31 [mE,nE] = size(E);
32 %Initialize scriptE ( $\mathcal{E}$ ) and scriptF ( $\mathcal{F}$ ) from the
33 % derivative array equations
34 scriptE = zeros((k+1)*mE,(k+1)*nE); scriptF = zeros((k+1)*mE,nE);
35 %Construct scriptE and scriptF
36 for i = 1:k+1
37     scriptE(1+(i-1)*mE:i*mE,1+(i-1)*nE:i*nE) = E;
38 end
39 if k > 0
40     for i = 1:k
41         scriptE(1+i*mE:(i+1)*mE,1+(i-1)*nE:i*nE) = F;
42     end
43 else end
44 scriptF(1:mE,1:nE) = F;
45
46 %Determine the size of scriptE to identify the columns of V that are a
47 % basis for N(scriptE)
48 [mscriptE,nscriptE] = size(scriptE);
49 %Initialize zerosum so the first norm has something to which to add
50 zerosum = 0;
51 %Construct [scriptE scriptF]
52 main = [scriptE scriptF];
53 %If [scriptE scriptF] is full row rank
54 if rank(main) == (k+1)*mE
55     %Calculate svd(scriptE)
56     [U,S0,V] = svd(scriptE); S = Rounding(S0);
57     %Calculate rank(S)
58     rS = rank(S);
59     %Implement 5) and 6) from steps to determine the index
60     for i = (rS+1):nscriptE
61         zerocheck = norm(V(1:mE,i)); zerosum = zerosum + zerocheck;
62     end
63 else end
64 %If finalsum equals 0, define the index as k; otherwise, continue for loop
65 finalsum = Rounding(zerosum);
66 if finalsum == 0
67     index = k
68     break
69 else end
70 end %end for loop
71
72 %end      index.m

```

```

1 %script LTISystems.m
2
3 %In script LTISystems.m, the linear time-invariant system of DAEs is
4 % defined, E(x(t)') + Fx(t) = Bu(t), with output equation y(t) = Cx(t).
5
6 %LTISystems.m passes the system to MainLTIObs.m, a main function that calls
7 % other programs for calculating full-order, reduced-order, and maximally
8 % reduced observers.
9 %The goal of LTISystems.m is to keep MainLTIObs.m and the programs it calls
10 % as general as possible by defining example-specific components outside
11 % of their operations.
12 %*****
13
14 %Define a symbolic variable t for the control
15 syms t
16 %The example system is representative of a constrained mechanical system
17 %(Chapter 5 example)
18 % x1(t)' = x2(t)
19 % x2(t)' = K*x1(t) + S*x2(t) + (H^T)*x3(t) + G*u1(t)
20 % 0 = H*x1(t) + u2(t)
21 K = [-2 1;1 -2]; S = (1/4)*[1 0;0 1]; H = [1 -1]; G = [1;1];
22 %Symbolically define the control
23 u = [sin(t);sin(t)];
24 %Identity matrix and zero matrices and vectors for building the matrices of
25 % the matrix pencil {E,F}
26 I = eye(2); Z2b2 = zeros(2,2); Z2b1 = zeros(2,1); Z1b2 = zeros(1,2);
27 Z1b1 = zeros(1,1); Z5b2 = zeros(5,2); Z5b5 = zeros(5,5);
28 %Build the matrix pencil; E is 5x5 and F is 5x5
29 E = [I Z2b2 Z2b1;Z2b2 I Z2b1;Z1b2 Z1b2 Z1b1];
30 F = -[Z2b2 I Z2b1;K S H';H Z1b2 Z1b1];
31 %Build matrix B in system of DAEs; B is 5x2
32 B = [Z2b1 Z2b1;G Z2b1;Z1b1 1];
33
34 %Define output matrix C
35 C = [0 0 1 0 0;0 0 0 1 0]; %Case 1, Ca
36 %C = [0 0 1 -1 0]; %Case 2, Cb
37 %C = [0 0 1 1 0]; %Case 3, Cc
38 casenum = 1; %the case number (1, 2, or 3)
39 %Determine the size of C
40 [mC,nC] = size(C);
41 %Determine if output matrix C is full row rank; count keeps track of which
42 % rows of C have already been tested for linear independence
43 count = 1;
44 %Check the rank of output matrix C (with Rounding.m is understood)
45 [UC,svdC0,VC] = svd(C);

```

```

46 %Force elements that should be zero to be zero
47 svdC = Rounding(svdC0); rC = rank(svdC);
48 %If C is full row rank, let Cfr = C: y(t) = Cfr*x(t)
49 if rC == mC
50     Cfr = C; disp('In LTISystems.m, C is full row rank.')
51 %If C is not full row rank, take rC linearly independent rows from C to
52 % define a new matrix Cfr: yfr(t) = Cfr*x(t)
53 else
54     for i=1:rC
55         for j=count:mC
56             Cfr(i,:) = C(j,:);
57             [UCfr,svdCfr0,VCfr] = svd(Cfr); svdCfr = Rounding(svdCfr0);
58             if rank(svdCfr) == i
59                 count = j + 1;
60                 break
61             else end
62         end
63     end
64 disp('In LTISystems.m, C has been modified so it is now full row rank.')
65 end
66
67 %Define lambda for the differential polynomial (d/dt)+lambda; lambda = 0 or
68 % 2; other values considered are 1, 3, 4, 5
69 lambda = 2;
70 %rho is a scalar for building the desired eigenvalue matrix for placing
71 % observable eigenvalues, dEig=rho*eye(), assuming all of the desired
72 % eigenvalues for one observer are the same and the value does not change
73 % when considering the different observers; rho = -1,-2,-3,-4,-5
74 rho = -1;
75
76 %Define fn for saving .mat files and passing through switch
77 % fn = [case #, lambda value, abs(rho), Reduced Observer identifier,
78 %         Completion identifier, on1].
79 % note, Reduced Observer identifier may be 1 or 2: if 1, reduced-order
80 % observer; if 2, maximally reduced observer.
81 % note, Completion identifier may be 1 or 2: if 1, (S)LSC; if 2, ASC.
82 %fn(4) and fn(5) are assigned as 0 to pass to MainLTIObs.m.
83 %on1 = on/off switch for constructing maximally reduced observer's
84 % extended output matrix: on1 = 0, save C; on1 = 1, save GF.
85 on1 = 0; fn = [casenum lambda -rho 0 0 on1];
86
87 %Call MainLTIObs.m
88 MainLTIObs(E,F,B,Cfr,u,lambda,rho,fn)
89
90 %end    LTISystems.m

```

```

1 function J = MainLTIObs(E,F,B,C,u,lambda,rho,fn)
2
3 %Function MainLTIObs.m is called by LTISystems.m.
4
5 %This function constructs the full-order (FO), reduced-order (RO), and
6 % maximally reduced (MR) observers for a linear time-invariant system of
7 % DAEs, E(x(t)') + Fx(t) = Bu(t), with output equation y(t) = Cx(t).
8
9 %Description of inputs
10 %E,F,B,C - coefficient matrices from the system being considered.
11 % Note, C is modified in LTISystems.m so that it is full row rank.
12 %u - symbolically defined control.
13 %lambda - for the differential polynomial (d/dt)+lambda.
14 %rho - a scalar passed through by the user to build the desired eigenvalue
15 % matrix for placing observable eigenvalues, dEig=rho*eye().
16 %fn - values mostly for names of .mat files in various programs;
17 % fn(6) = on1 for on/off switch, saving C or GF in extended output matrix.
18
19 %This function may call LTIDAE_SCDerArray.m, LTIDAE_AltSC.m, ExtendC.m,
20 %FullOrdLTI_Calc.m, RedOrdLTI_Calc.m, LTIDAE_Results.m, LTIDAE_Resultsmod.m
21 %*****
22
23 %Define a symbolic variable t for the control
24 syms t
25 %Assign switch: save C if on1 = 0, save GF if on1 = 1
26 on1 = fn(6);
27 %Determine the size of C
28 [mC,nC] = size(C);
29
30 %Full-order observer *****
31 %
32 %Determine the derivative array equations for the (stabilized) least
33 % squares completion and the alternative stabilized completion
34 % (AtildeFOL, etc. for (S)LSC; AtildeFOA, etc. for ASC)
35 [AtildeFOL,BtildeFOL,scriptEL,scriptFL,scriptBL] = ...
    LTIDAE_SCDerArray(E,F,B,lambda);
36 [AtildeFOA,BtildeFOA] = LTIDAE_AltSC(E,F,B,lambda);
37 %Calculate the gain matrix Lf (for observable eigenvalue placement)
38 disp('FullOrdLTI_Calc.m has been called, (S)LSC')
39 fn(5) = 1; [LfL] = FullOrdLTI_Calc(AtildeFOL,C,rho,fn);
40 disp('FullOrdLTI_Calc.m has been called, ASC')
41 fn(5) = 2; [LfA] = FullOrdLTI_Calc(AtildeFOA,C,rho,fn);
42 %*****
43
44 %Reduced-order and maximally reduced observers *****

```

```

45 %
46 %Reduced-order observer's transformation matrix R
47 %Let Cr equal the transpose of the basis vectors for the nullspace of C
48 [UC,svdC,VC] = svd(C);
49 %Initialize transformation matrix R; the first mC rows of R are the output
50 % matrix; the last nC-mC rows are Cr = VC(:,mC+1:nC)'^T. Display R.
51 R = zeros(nC); R(1:mC,:) = C; R(mC+1:nC,:) = VC(:,mC+1:nC)'; R;
52 %Define the inverse of transformation matrix R
53 iR = inv(R);
54 %Define matrix CR (CR should equal [I 0]); rCR from the size of CR is used
55 % for dividing the system into 'measurable' and 'unmeasurable' parts.
56 CR = C*inv(R); [rCR,nCR] = size(CR);
57
58 %Maximally reduced observer's transformation matrix \overline{R}
59 %Find the derivative array equations without stabilized differentiation
60 [AtildeTM,BtildeTM,scriptETM,scriptFTM,scriptBTM] = LTIDAE_SCDerArray(E,F,B,0);
61 %scriptG is the last mscriptE-rscriptE columns of UscripETM transposed,
62 % where svd(scriptETM) = [UscripETM,DscriptETM0,VscriptETM],
63 % so that scriptG*scriptETM = 0
64 %'TM' is for transformation matrix
65 [mscriptETM,nscriptETM] = size(scriptETM);
66 [UscripETM,DscriptETM0,VscriptETM] = svd(scriptETM);
67 DscriptETM = Rounding(DscriptETM0); rscriptETM = rank(DscriptETM);
68 %Define scriptG
69 scriptG = (UscripETM(:,rscriptETM+1:mscriptETM))';
70 %Extend output matrix if possible
71 %Two methods are considered for extending C (on1 switch is fn(6))
72 % 1) If C is saved, linearly independent rows from GFprod are selected.
73 % 2) If GFprod is saved, linearly independent rows from C are selected;
74 % this method assumes GFprod is full row rank.
75 %ExtendC.m also determines \overline{R} and the rows from GBprod needed to
76 % solve the maximally reduced observer
77 %extend = 1 indicates an extended output matrix Cext exists, while
78 %extend = 0 indicates an extended output matrix Cext does not exist (the
79 % output matrix could not be extended)
80 [Cext,Rext,GBext,rDmodC,rowsused,extend] = ...
81 ExtendC(scriptG,scriptFTM,scriptBTM,C,fn);
82 %Define the inverse of \overline{R} (Rext)
83 iRext = inv(Rext);
84
85 %Transformed systems (transformed completions)
86 %Determine R*Atilde*inv(R) and R*Btilde of the transformed system
87 % q(t)' = R*Atilde*inv(R)*q(t) + R*Btilde*u(t)
88 % y(t) = C*inv(R)*q(t)
89 if extend == 1 %extended output Cext exists

```

```

89 %Transformation matrix R, (S)LSC
90 AtildeROL = R*AtildeFOL*iR; BtildeROL = R*BtildeFOL;
91 %Transformation matrix R, ASC
92 AtildeROA = R*AtildeFOA*iR; BtildeROA = R*BtildeFOA;
93 %Transformation matrix \overline{R}, (S)LSC
94 AtildeMRL = Rext*AtildeFOL*iRext; BtildeMRL = Rext*BtildeFOL;
95 %Transformation matrix \overline{R}, ASC
96 AtildeMRA = Rext*AtildeFOA*iRext; BtildeMRA = Rext*BtildeFOA;
97 else %extend == 0, extended output Cext does not exist
98 %Transformation matrix R, (S)LSC
99 AtildeROL = R*AtildeFOL*iR; BtildeROL = R*BtildeFOL;
100 %Transformation matrix R, ASC
101 AtildeROA = R*AtildeFOA*iR; BtildeROA = R*BtildeFOA;
102 end
103
104 %Find D,F (coefficient matrices in RO and MR observers), and calculate the
105 % gain matrix Lr (for observable eigenvalue placement)
106 if extend == 1 %extended output Cext exists
107 disp('RedOrdLTI_Calc.m has been called using C, (S)LSC')
108 fn(4) = 1; fn(5) = 1; %R, (S)LSC
109 [DROL,FROL,LrROL] = RedOrdLTI_Calc(AtildeROL,BtildeROL,rCR,rho,fn);
110 disp('RedOrdLTI_Calc.m has been called using C, ASC')
111 fn(5) = 2; %R, ASC
112 [DROA,FROA,LrROA] = RedOrdLTI_Calc(AtildeROA,BtildeROA,rCR,rho,fn);
113 disp('RedOrdLTI_Calc.m has been called using Cext, (S)LSC')
114 fn(4) = 2; fn(5) = 1; %\overline{R}, (S)LSC
115 [DMRL,FMRL,LrMRL] = RedOrdLTI_Calc(AtildeMRL,BtildeMRL,rDmodC,rho,fn);
116 disp('RedOrdLTI_Calc.m has been called using Cext, ASC')
117 fn(5) = 2; %\overline{R}, ASC
118 [DMRA,FMRA,LrMRA] = RedOrdLTI_Calc(AtildeMRA,BtildeMRA,rDmodC,rho,fn);
119 else %extend == 0, extended output Cext does not exist
120 disp('RedOrdLTI_Calc.m has been called using C, (S)LSC')
121 fn(4) = 1; fn(5) = 1; %R, (S)LSC
122 [DROL,FROL,LrROL] = RedOrdLTI_Calc(AtildeROL,BtildeROL,rCR,rho,fn);
123 disp('RedOrdLTI_Calc.m has been called using C, ASC')
124 fn(5) = 2; %R, ASC
125 [DROA,FROA,LrROA] = RedOrdLTI_Calc(AtildeROA,BtildeROA,rCR,rho,fn);
126 %Define LrMRL and LrMRA as 0 to bypass later if else statements
127 LrMRL = 0; LrMRA = 0;
128 end
129 %*****%
130
131 %Solutions ****%
132 %
133 %If neither the FO nor the RO observer can be constructed, but the MR

```

```

134 % observer can be constructed
135 if LfL == 0
136     if (norm(LrMRL) > 0)
137         disp('LTIDAE_Resultsmod.m has been called: MR observer, (S)LSC')
138         fn(4) = 2; fn(5) = 1; \%overline{R}, (S)LSC
139         LTIDAE_Resultsmod(AtildeFOL,BtildeFOL,C,AtildeMRL,BtildeMRL,DMRL, ...
140                         FMRL,LrMRL,Rext,GBext,u,fn)
141         disp('LTIDAE_Resultsmod.m has been called: MR observer, ASC')
142         fn(5) = 2; \%overline{R}, ASC
143         LTIDAE_Resultsmod(AtildeFOA,BtildeFOA,C,AtildeMRA,BtildeMRA,DMRA, ...
144                         FMRA,LrMRA,Rext,GBext,u,fn)
145     else end
146 %If the full-order and reduced-order observers can be constructed
147 else
148     if extend == 1 %extended output Cext exists
149         disp('LTIDAE_Results.m has been called: FO, RO, MR observers, (S)LSC')
150         fn(4) = 2; fn(5) = 1; %R and \overline{R}, (S)LSC
151         LTIDAE_Results(AtildeFOL,BtildeFOL,C,LfL,AtildeROL,BtildeROL, ...
152                         DROL,FROL,LrROL,R,AtildeMRL,BtildeMRL,DMRL,FMRL,LrMRL, ...
153                         Rext,GBext,u,rowused,extend,fn)
154         disp('LTIDAE_Results.m has been called: FO, RO, MR observers, ASC')
155         fn(5) = 2; %R and \overline{R}, ASC
156         LTIDAE_Results(AtildeFOA,BtildeFOA,C,LfA,AtildeROA,BtildeROA, ...
157                         DROA,FROA,LrROA,R,AtildeMRA,BtildeMRA,DMRA,FMRA,LrMRA, ...
158                         Rext,GBext,u,rowused,extend,fn)
159     else %extend == 0, extended output Cext does not exist
160         disp('LTIDAE_Results.m has been called: FO, RO observers, (S)LSC')
161         fn(4) = 1; fn(5) = 1; %R only, no \overline{R}, (S)LSC
162         LTIDAE_Results(AtildeFOL,BtildeFOL,C,LfL,AtildeROL,BtildeROL,DROL,FROL, ...
163                         LrROL,R,0,0,0,0,0,0,u,extend,fn)
164         disp('LTIDAE_Results.m has been called: FO, RO observers, ASC')
165         fn(5) = 2; %R only, no \overline{R}, ASC
166         LTIDAE_Results(AtildeFOA,BtildeFOA,C,LfA,AtildeROA,BtildeROA,DROA,FROA, ...
167                         LrROA,R,0,0,0,0,0,0,u,extend,fn)
168     end
169 end
170
171 %Save workspace
172 save(strcat('MainC',num2str(fn(1)), 'L', num2str(fn(2)), 'Rm', num2str(fn(3))))
173
174 %end MainLTIObs.m

```



```

1 function [Atilde,Btilde,scriptE,scriptF,scriptB] = LTIDAE_SCDerArray(E,F,B,lam)
2
3 %Function LTIDAE_SCDerArray.m is called by MainLTIObs.m, DAEManObs.m.

```

```

4
5 %The purpose of LTIDAE_SCDerArray.m is to
6 % 1) determine the derivative array equations for a linear time-invariant
7 %      system of DAEs and
8 % 2) define \widetilde{A} and \widetilde{B} in the (stabilized) least
9 %      squares completion ((S)LSC).
10
11 %Description of inputs
12 %E,F,B - coefficient matrices from the system being considered.
13 %lam - value of lambda used in the differential polynomial (d/dt)+lambda
14 % (lambda is zero for the least squares completion).
15
16 %Description of outputs
17 %Atilde,Btilde - coefficient matrices in the (stabilized) least squares
18 % completion.
19 %scriptE,scriptF,scriptB - coefficient matrices in the derivative array
20 % equations.
21
22 %** Note **
23 %Sections of this program are example specific.
24 %*****
25
26 %Identity matrix and zero matrices and vectors for building the coefficient
27 % matrices of the derivative array equations
28 I = eye(2); Z2b2 = zeros(2,2); Z2b1 = zeros(2,1); Z1b2 = zeros(1,2);
29 Z1b1 = zeros(1,1); Z5b2 = zeros(5,2); Z5b5 = zeros(5,5);
30 %Build the derivative array equations, scriptE*w + scriptF*x = scriptB*v
31 %scriptE is 20x20, scriptF is 20x5, scriptB is 20x8
32 scriptE = [E Z5b5 Z5b5 Z5b5; (lam*E)+F E Z5b5 Z5b5;
33             ((lam^2)*E)+(2*lam*F) (2*lam*E)+F E Z5b5;
34             ((lam^3))*E+(3*(lam^2)*F) (3*(lam^2)*E)+(3*lam*F) (3*lam*E)+F E];
35 scriptF = [F;lam*F;(lam^2)*F;(lam^3)*F];
36 scriptB = [B Z5b2 Z5b2 Z5b2; lam*B B Z5b2 Z5b2;
37             (lam^2)*B (2*lam)*B B Z5b2; (lam^3)*B (3*(lam^2)*B) (3*lam*B) B];
38 %Find the pseudoinverse of scriptE for the (S)LSC
39 PscriptE = pinv(scriptE);
40 %(S)LSC: wbar = -PscriptE*scriptF*x + PscriptE*scriptB*v
41 VA = -PscriptE*scriptF; WB = PscriptE*scriptB;
42 %Submatrices of VA and WB for (stabilized) least squares completion
43 %xtilde' = Atilde*xtilde + sum(Btilde_i*u^(i)),
44 % where the sum is from i=0 to i=k, u^(i) represents the ith derivative of
45 % u, and k is the index of the system of DAEs
46 Atilde = VA(1:5,:); Btilde = WB(1:5,:);
47 %The eigenvalues of Atilde (\widetilde{A}) should equal the matrix pencil
48 % eigenvalues and -lambda

```

```

49 anseigAtilde = eig(Atilde);
50
51 %end      LTIDAE_SCDerArray.m

1 function [Atilde,Btilde] = LTIDAE_AltSC(E,F,B,lambda)
2
3 %This function is called by MainLTIObs.m.
4
5 %The purpose of LTIDAE_AltSC.m is to define \widetilde{A} and \widetilde{B}
6 % in the alternative stabilized completion (ASC).
7
8 %Description of inputs
9 %E,F,B - coefficient matrices from the system being considered.
10 %lambda - value used in the differential polynomial (d/dt)+lambda
11 % (lambda is zero for the alternative least squares completion).
12
13 %Description of outputs
14 %Atilde,Btilde - coefficient matrices in the ASC.
15
16 %** Note **
17 %Sections of this program are example specific.
18 %***** ****
19
20 %Identity matrix and zero matrices and vectors for building the coefficient
21 % matrices of the derivative array equations
22 I = eye(2); Z2b2 = zeros(2,2); Z2b1 = zeros(2,1); Z1b2 = zeros(1,2);
23 Z1b1 = zeros(1,1); Z5b2 = zeros(5,2); Z5b5 = zeros(5,5); Z3b2 = zeros(3,2);
24 %Build the derivative array equations, scriptE*w + scriptF*x = scriptB*z
25 %scriptE is 20x20, scriptF is 20x5, scriptB is 20x8
26 scriptE = [E Z5b5 Z5b5 Z5b5; F E Z5b5 Z5b5; Z5b5 F E Z5b5; Z5b5 Z5b5 F E];
27 scriptF = [F; Z5b5; Z5b5; Z5b5];
28 scriptB = [B Z5b2 Z5b2 Z5b2; Z5b2 B Z5b2 Z5b2; Z5b2 Z5b2 B Z5b2;
29             Z5b2 Z5b2 Z5b2 B];
30 %Define matrices that include one less differentiation than required for
31 % building the derivative array equations (do not include last block row)
32 sEt = scriptE(1:15,:); %sEt=scriptEtilda
33 sFt = scriptF(1:15,:); %sFt=scriptFtilde
34 sBt = scriptB(1:15,:); %sBt=scriptBtilde
35
36 %Find an orthonormal basis Z2 for the nullspace of sEt^T
37 %Note, sEt = sEtU*sEtsvd*(sEtV^T)
38 [sEtU,sEtsvd0,sEtV] = svd(sEt);
39 %Force elements that should be zero to be zero
40 [msEtsvd,nsEtsvd] = size(sEtsvd0); sEtsvd = Rounding(sEtsvd0);
41 %Calculate the rank (rsEtsvd) of sEt (also sEtsvd)

```

```

42 rsEtsvd = rank(sEtsvd);
43 %Define Z2T (Z2T is 3x15) and confirm (Z2^T)*sEt=0
44 Z2 = sEtU(:,(rsEtsvd+1):msEtsvd); Z2T = Z2'; (Z2T)*sEt;
45
46 %Find an orthonormal basis T2 for the nullspace of FM=(Z2^T)*sFt
47 %Note, FM = FMU*FMsvd*(FMV^T) (FM=scriptFtilde modified)
48 FM = (Z2T)*(sFt); [FMU,FMsvd0,FMV] = svd(FM);
49 %Force elements that should be zero to be zero
50 [mFMsvd,nFMsvd] = size(FMsvd0); FMsvd = Rounding(FMsvd0);
51 %Calculate the rank (rFMsvd) of FM (also FMsvd)
52 rFMsvd = rank(FMsvd);
53 %Define T2 (T2 is 5x2) and confirm that FM*T2=0
54 T2 = FMV(:,(rFMsvd+1):nFMsvd); FM*T2;
55
56 %Find an orthonormal basis Z1 for the range of EM=E*T2
57 %Note, EM = EMU*EMsvd*(EMV^T) (EM=E modified)
58 EM = E*T2; [EMU,EMsvd0,EMV] = svd(EM);
59 %Force elements that should be zero to be zero
60 [mEMsvd,nEMsvd] = size(EMsvd0); EMsvd = Rounding(EMsvd0);
61 %Calculate the rank (rEMsvd) of EM (also EMsvd)
62 rEMsvd = rank(EMsvd);
63 %Define Z1T (Z1T is 2x5)
64 Z1 = EMU(:,1:rEMsvd); Z1T = Z1';
65
66 %Define square and invertible Xbar=[(Z1^T)*E;(Z2(:,1:5)^T)*F]
67 Xbar = [(Z1T)*E;Z2T(:,1:5)*F];
68 %Alternative Stabilized Completion
69 % x' = -inv([(Z1^T)*E;((Z-{2,0})^T)*F])*%
70 % ([ (Z1^T)*F;lambda*((Z-{2,0})^T)*F]*x-
71 % [(Z1^T)*f1;(Z2^T)*ftilde' + lambda*(Z2^T)*ftilde])
72 % where f1=Bu, ftilde=[Bu;Bu';Bu"], and ftilde'=[Bu';Bu",Bu'"]
73 iXbar = inv(Xbar);
74 xcoeff = [(Z1T)*F;lambda*(Z2T(:,1:5))*F];
75 ucoeff = [(Z1T)*B Z2b2 Z2b2 Z2b2;
76 lambda*(Z2T(:,1:5))*B (Z2T(:,1:5)+(lambda*(Z2T(:,6:10))))*B ...
(Z2T(:,6:10)+(lambda*(Z2T(:,11:15))))*B Z2T(:,11:15)*B];
77 %xtilde' = Atilde*xtilde + sum(Btilde_i*u^(i)),
78 % where the sum is from i=0 to i=k, u^(i) represents the ith derivative of
79 % u, and k is the index of the system of DAEs
80 Atilde = -iXbar*xcoeff; Btilde = iXbar*ucoeff;
81 %The eigenvalues of Atilde (\widetilde{A}) should equal the matrix pencil
82 % eigenvalues and -lambda
83 anseigAtilde = eig(Atilde);
84
85 %end LTIDAE_AltSC.m

```

```

1 function [Lf] = FullOrdLTI.Calc(Atilde,C,rho,fn)
2
3 %Function FullOrdLTI.Calc.m is called by MainLTIObs.m, DAEManObs.m.
4
5 %The purpose of FullOrdLTI.Calc.m is to determine gain matrix Lf for the
6 % full-order observer.
7
8 %Description of inputs
9 %Atilde,C -coefficient matrices from the pair of the system to be observed.
10 %rho -a scalar passed through by the user to build the desired eigenvalue
11 % matrix for placing observable eigenvalues, dEig=rho*eye().
12 %fn -values for name of .mat file.
13
14 %This function may call either EigPlace1.m or EigPlace2.m.
15 %*****
16
17 %Build the observability matrix O and calculate its rank (rO)
18 [mAtilde,nAtilde] = size(Atilde); [mC,nC] = size(C);
19 %Initialize the observability matrix
20 O = zeros(mC*mAtilde,nAtilde);
21 for i=1:mAtilde
22     O(((i-1)*mC)+1:(i*mC),:) = C*(Atilde^(i-1));
23 end
24 %Note, O = OU*Osvald0*(OV^T)
25 [OU,Osvald0,OV] = svd(O);
26 %Force elements that should be zero to be zero
27 Osvald = Rounding(Osvald0);
28 %Calculate the rank (rOsvald) of O (also Osvald)
29 rOsvald = rank(Osvald);
30
31 %Define the similarity transformation matrix Q as either an identity
32 % matrix or as OV, an orthogonal matrix where the last nOsvald-rOsvald
33 % columns span the nullspace of O, depending on the observability of O
34 [mOsvald,nOsvald] = size(Osvald);
35 if rOsvald == nOsvald
36     Q = eye(rOsvald);
37 else
38     Q = OV;
39 end
40 %Use the similarity transformation blockA=inv(Q)*Atilde*Q
41 blockA = (inv(Q))*Atilde*Q;
42 %to determine A1, observable block
43 A1 = blockA(1:rOsvald,1:rOsvald);
44 %check observable eigenvalues
45 anseigA1 = eig(A1);

```

```

46 %to determine A3, unobservable block
47 A3 = blockA(rOsvald+1:nAtilde,rOsvald+1:nAtilde);
48 %check unobservable eigenvalues
49 anseigA3 = eig(A3);
50 %Check to make sure any unobservable eigenvalues are negative
51 %If any of the unobservable eigenvalues are nonnegative, print out the
52 % unobservable eigenvalues and terminate FullOrdLTI_Calc.m.
53 [mA3,nA3] = size(A3);
54 tol = (1*(10^(-10)));
55 if mA3 > 0
56     for i=1:mA3
57         if anseigA3(i) >= -tol
58             disp('A full-order observer will not estimate the state.')
59             anseigA3
60             Lf = 0
61             return
62         else end
63     end
64 else end
65 %Use the similarity transformation blockC=C*Q
66 blockC = C*Q;
67 %to determine C1
68 C1 = blockC(:,1:rOsvald);
69
70 %Compute M in A1-M*C1 using program EigPlace1.m if mC1 = 1 and program
71 % EigPlace2.m if mC1 > 1
72 %M is used for eigenvalue placement
73 dEigL = rho*eye(rOsvald); [mC1,nC1] = size(C1);
74 if mC1 == 1 %for single output, C1 1xn
75     M = EigPlace1(rOsvald,A1,C1,dEigL);
76 else %mC1 > 1, for multi output, C1 mxn
77     M = EigPlace2(rOsvald,A1,C1,dEigL);
78 end
79 %Confirm the eigenvalues match the desired eigenvalues (dEigM)
80 anseigM = eig(A1-(M*C1));
81 %Define Lf; either Lf=Q*M or Lf=Q*[M;0]
82 [mM,nM] = size(M);
83 if mM == nOsvald
84     Lf = Q*M;
85 else
86     %Build a zero matrix
87     Lfa = zeros(nOsvald,nM);
88     %Assign M to the appropriate location for building Lf
89     Lfa(1:mM,:) = M;
90     %Calculate Lf

```

```

91      Lf = Q*Lfa;
92  end
93 %Confirm the observable eigenvalues match the desired eigenvalues (dEigLf)
94 anseigLf = eig(Atilde-(Lf*C));
95
96 %Save workspace
97 save(strcat('FullC',num2str(fn(1)), 'L', num2str(fn(2)), 'Rm', num2str(fn(3)), ...
98           'SC', num2str(fn(5))))
99 %end      FullOrdLTI.Calc.m

1 function [Cext,Rext,GBext,rDmodC,rowused,extend] = ...
2 ExtendC(scriptG,scriptF,scriptB,C,fn)
3
4 %Function ExtendC.m is called by MainLTIObs.m.
5
6 %The purpose of ExtendC.m is to extend the output matrix, if
7 % possible, with information from the characterization of the solution
8 % manifold for constructing the maximally reduced observer.
9
10 %Description of inputs
11 %scriptG - a full row rank matrix such that scriptG*scriptE=0.
12 %scriptF, scriptB - matrices from the derivative array equations,
13 % scriptE*w(t) + scriptF*x(t) = scriptB*v(t).
14 %C - original output matrix.
15
16 %Description of outputs
17 %Cext - extended output matrix.
18 %Rext - transformation matrix \overline{R} equal to [Cext;Cr].
19 %GBext - rows from scriptG*scriptB for extended output equation.
20 %rDmodC - rank of the extended output matrix.
21 %extend - extend = 1 indicates an extended output matrix Cext exists, while
22 % extend = 0 indicates an extended output matrix Cext does not exist.
23 %Note, if extend = 0, Cext, Rext, GBext, and rowused are returned as 0.
24 %*****
25 %Assign switch: save C if onl = 0, save GF if onl = 1
26 onl = fn(6);
27 %Determine the size of C
28 [mC,nC] = size(C);
29
30 %Calculate scriptG*scriptF for extended output matrix
31 GFprod0 = scriptG*scriptF;
32 %Force elements that should be zero to be zero

```

```

33 GFprod = Rounding(GFprod0);
34 %Determine the size of GFprod
35 [mGFprod,nGFprod] = size(GFprod);
36 %Determine the rank of GFprod
37 [UGF,DGF0,VGF] = svd(GFprod);
38 %Force elements that should be zero to be zero
39 DGF = Rounding(DGF0);
40 %Calculate the rank of DGF (rDGF)
41 rDGF = rank(DGF);
42
43 %Calculate scriptG*scriptB for extended output
44 GBprod0 = scriptG*scriptB;
45 %Force elements that should be zero to be zero
46 GBprod = Rounding(GBprod0);
47
48 %Extension of output matrix for rank check
49 modC = [C;GFprod];
50 %modC may not be full row rank; determine the rank of modC
51 [UmodC,DmodC0,VmodC] = svd(modC);
52 %Force elements that should be zero to be zero
53 DmodC = Rounding(DmodC0);
54 %Calculate the rank (rDmodC) of DmodC (also [C;GFprod])
55 rDmodC = rank(DmodC);
56
57 if on1 == 0 %save C
58 %
59 if rDmodC > mC %It is possible to extend C
60 %extend = 1 indicates the extended output matrix exists
61 extend = 1;
62 %Determine the size of modC
63 [mmodC,nmodC] = size(modC);
64 %count1 keeps track of which rows of GFprod have already been tested
65 % for linear independence; count2 is used to build rowused, a vector
66 % saving the row numbers of the rows from GFprod used to extend C
67 count1 = 1; count2 = 1;
68 %If modC is full row rank, let Cext = modC
69 if rDmodC == mmodC
70     disp('All rows of C and GFprod are linearly independent.')
71     %frr = full row rank switch; rowused is a dummy variable for this case
72     frr = 1; Cext = modC; rowused = mmodC + 1;
73 %If modC is not full row rank, define a new matrix Cext containing C
74 % and rows from GFprod so that the rows of Cext are linearly
75 % independent: yext(t) = Cext*x(t)
76 else
77     frr = 0; Cext = zeros(rDmodC,nC); Cext(1:mC,:) = C;

```

```

78     for i=(mC+1):rDmodC
79         for j=count1:mGFprod
80             Cext(i,:) = GFprod(j,:); [U,Dsvd0,V] = svd(Cext);
81             %Force elements that should be zero to be zero
82             Dsvd = Rounding(Dsvd0);
83             %Calculate the rank of Dsvd (rDsvd)
84             rDsvd = rank(Dsvd);
85             if rDsvd == i
86                 rowused(count2) = j; count1 = j + 1; count2 = count2 + 1;
87                 break
88             else end
89         end
90     end
91 disp('The extended output matrix saves C.')
92 end %end output matrix extension
93 %
94 %Pick out the rows of scriptG*scriptB corresponding with the rows of
95 % scriptG*scriptF
96 if frr == 1
97     GBext = GBprod;
98 else %frr == 0
99     for i=1:(count2-1)
100        GBext(i,:) = GBprod(rowused(i),:);
101    end
102 end
103 %
104 %For the maximally reduced observer, build the transformation matrix Rext
105 % (\overline{R}), where Rext=[Cext;Cr] is nonsingular
106 %Determine the size of Cext
107 [mCext,nCext] = size(Cext);
108 %Let Cr equal the transpose of the basis vectors for the nullspace of Cext
109 [UCext,svdCext,VCext] = svd(Cext);
110 %Initialize transformation matrix Rext
111 Rext = zeros(nCext);
112 %The first mCext rows of Rext are the extended output matrix, and the last
113 % nC-mCext rows are Cr = VCext(:,mCext+1:nCext)^T
114 Rext(1:mCext,:) = Cext; Rext(mCext+1:nCext,:) = VCext(:,mCext+1:nCext)';
115 else %It is not possible to extend C; extend = 0 indicates an extended
116 % output matrix does not exist
117 Cext = 0; Rext = 0; GBext = 0; rowused = 0; extend = 0;
118 end %end save C
119 %
120 else %onl == 1, save GF
121 %
122 if rDmodC > mGFprod %It is possible to extend GFprod

```

```

123 extend = 1; modGF = [GFprod;C]; [mmodGF,nmodGF] = size(modGF);
124 %count1 keeps track of which rows of C have already been tested for linear
125 % independence; count2 is used to build rowused, a vector
126 % saving the row numbers of the rows from C used to extend GFprod
127 count1 = 1; count2 = 1;
128 %If modGF is full row rank, let Cext = modGF
129 if rDmodC == mmodGF
130     disp('All rows of GFprod and C are linearly independent.')
131     frr = 1; Cext = modGF;
132     for i=1:mC
133         rowused(i) = i;
134     end
135 %If modGF is not full row rank, define a new matrix Cext containing GFprod
136 % and rows from C so that the rows of Cext are linearly
137 % independent: yext(t) = Cext*x(t)
138 else
139     frr = 0; Cext = zeros(rDmodC,nC); Cext(1:mGFprod,:) = GFprod;
140     for i=(mGFprod+1):rDmodC
141         for j=count1:mC
142             Cext(i,:) = C(j,:); [U,Dsvd0,V] = svd(Cext);
143             Dsvd = Rounding(Dsvd0); rDsvd = rank(Dsvd);
144             if rDsvd == i
145                 rowused(count2) = j; count1 = j + 1; count2 = count2 + 1;
146                 break
147             else end
148         end
149     end
150 disp('The extended output matrix saves GF.')
151 end %end output matrix extension
152 %
153 %Since GF is saved,
154 GBext = GBprod;
155 %
156 %Build the transformation matrix Rext
157 [mCext,nCext] = size(Cext); [UCext,svdCext,VCext] = svd(Cext);
158 Rext = zeros(nCext); Rext(1:mCext,:) = Cext;
159 Rext(mCext+1:nCext,:) = VCext(:,mCext+1:nCext)';
160 else %It is not possible to extend C; extend = 0 indicates an extended
161 % output matrix does not exist
162 Cext = 0; Rext = 0; GBext = 0; rowused = 0; extend = 0;
163 end %end save GF
164 %
165 end
166
167 %end      ExtendC.m

```

```

1 function [D,F,Lr] = RedOrdLTI_Calc(transA,transB,rCR,rho,fn)
2
3 %Function RedOrdLTI_Calc.m is called by MainLTIObs.m.
4
5 %The purpose of RedOrdLTI_Calc.m is to determine gain matrix Lr for the
6 % reduced-order or maximally reduced observer.
7
8 %Description of inputs
9 %transA,transB - coefficient matrices from the pair of the system to be
10 % observed.
11 %rCR - the rank of C*inv(R) or Cext*inv(Rext) (this rank is used for
12 % dividing the system into 'measurable' and 'unmeasurable' parts).
13 %rho - a scalar passed through by the user to build the desired eigenvalue
14 % matrix for placing observable eigenvalues, dEig=rho*eye().
15 %fn - values for name of .mat file.
16
17 %Description of outputs
18 %D,F - coefficient matrices for the reduced-order or maximally reduced
19 % observer.
20 %Gain matrix Lr.
21
22 %This function may call either EigPlace1.m or EigPlace2.m.
23 %*****
24
25 %Divide matrices transA (nxn) and transB (nxp) as follows
26 % transA = [A11 A12;A21 A22] and transB = [B1;B2] so that
27 % qm(t)' = A11*qm(t) + A12*qu(t) + B1*u(t)
28 % qu(t)' = A21*qm(t) + A22*qu(t) + B2*u(t)
29 %where A11 is (rCR)x(rCR) and A22 is (n-rCR)x(n-rCR)
30 %qm represents 'measurable' state and qu represents 'unmeasurable' state
31 [mtransA,ntransA] = size(transA);
32 A11 = transA(1:rCR,1:rCR); A12 = transA(1:rCR,(rCR+1):ntransA);
33 A21 = transA((rCR+1):ntransA,1:rCR);
34 A22 = transA((rCR+1):ntransA,(rCR+1):ntransA);
35 B1 = transB(1:rCR,:); B2 = transB((rCR+1):mtransA,:);
36 %Check eigenvalues of blocks A11, A22
37 anseigA11 = eig(A11); anseigA22 = eig(A22);
38
39 %Construct the reduced-order or maximally reduced observer
40 %Build the observability matrix O and calculate its rank (rO)
41 [mA22,nA22] = size(A22);
42 %Initialize the observability matrix
43 O = zeros(rCR*mA22,nA22);
44 for i=1:mA22
45     O(((i-1)*rCR)+1:(i*rCR),:) = A12*(A22^(i-1));

```

```

46 end
47 %Note, O = OU*Osvald0*(OV^T)
48 [OU,Osvald0,OV] = svd(O);
49 %Force elements that should be zero to be zero
50 Osvald = Rounding(Osvald0);
51 %Calculate the rank (rOsvald) of O (also Osvald)
52 rOsvald = rank(Osvald);
53
54 %Define the similarity transformation matrix Q as either an identity
55 % matrix or as OV, an orthogonal matrix where the last nOsvald-rOsvald
56 % columns span the nullspace of O, depending on the observability of O
57 [mOsvald,nOsvald] = size(Osvald);
58 if rOsvald == nOsvald
59     Q = eye(rOsvald);
60 else
61     Q = OV;
62 end
63 %Use the similarity transformation blockA=inv(Q)*A22*Q
64 blockA = (inv(Q))*A22*Q;
65 %to determine A1, observable block
66 A1 = blockA(1:rOsvald,1:rOsvald);
67 %check observable eigenvalues
68 anseigA1 = eig(A1);
69 %to determine A3, unobservable block
70 A3 = blockA(rOsvald+1:(ntransA-rCR),rOsvald+1:(ntransA-rCR));
71 %check unobservable eigenvalues
72 anseigA3 = eig(A3);
73 %Check to make sure any unobservable eigenvalues are negative
74 %If any of the unobservable eigenvalues are nonnegative, print out the
75 % unobservable eigenvalues and terminate RedOrdLTI_Calc.m.
76 [mA3,nA3] = size(A3);
77 tol = (1*(10^(-10)));
78 if mA3 > 0
79     for i=1:mA3
80         if anseigA3(i) >= -tol
81             disp('This observer will not estimate the state.')
82             anseigA3
83             D = 0; F = 0; Lr = 0
84             return
85         else end
86     end
87 else end
88 %Use the similarity transformation blockC=A12*Q
89 blockC = A12*Q;
90 %to determine C1

```

```

91 C1 = blockC(:,1:rOsvd);
92
93 %In order to continue with the eigenvalue placement, C1 needs to be full
94 % row rank; if it is not, remove the linearly dependent rows.
95 %Determine the rank of C1
96 [UC1,svdC10,VC1] = svd(C1); svdC1 = Rounding(svdC10); rC1 = rank(svdC1);
97 [mC1,nC1] = size(C1); C1check = Rounding(C1);
98 %Count keeps track of the rows of C1 checked for linear independence
99 count = 1;
100 %If C1 is full row rank, let Cfr = C1 (Cfr, or C1 full (row) rank)
101 if rC1 == mC1
102     Cfr = C1; disp('C1 is full row rank.')
103 %If C1 is not full row rank, define a new matrix Cfr containing rC1
104 % linearly independent rows from C1
105 elseif ((nC1 == 1) && (mC1 > 1)) %C1 is mx1
106 %Find the first nonzero component of C1 and define it to be Cfr
107 %Save the location of the first nonzero component
108     for i=1:mC1
109         if C1check(i) == 0
110             else
111                 Cfr = C1check(i); rowloc = i;
112                 break
113             end
114         end
115     else %rC1 does not equal mC1 and C1 is m by nC1, nC1 > 1
116 %Find the linearly independent rows of C1 and define them to be Cfr
117 %Save the locations of these rows
118     for i=1:rC1
119         for j=count:mC1
120             Cfr(i,:) = C1(j,:);
121             [UCfr,svdCfr0,VCfr] = svd(Cfr); svdCfr = Rounding(svdCfr0);
122             if rank(svdCfr) == i
123                 rowloc(i) = count; count = j + 1;
124                 break
125             else end
126         end
127     end
128     disp('In RedOrdLTI.Calc.m, C1 has been modified to be full row rank.')
129 end
130
131 %Compute M in A1-M*Cfr using program EigPlace1.m if mCfr = 1 and program
132 % EigPlace2.m if mCfr > 1
133 %M is used for eigenvalue placement
134 dEigL = rho*eye(rOsvd); [mCfr,nCfr] = size(Cfr);
135 if mCfr == 1 %for single output, Cfr 1xn

```

```

136      M = EigPlace1(rOsvd,A1,Cfr,dEigL);
137  else %mCfr > 1, for multi output, Cfr mxn
138      M = EigPlace2(rOsvd,A1,Cfr,dEigL);
139  end
140 %Confirm the eigenvalues match the desired eigenvalues (dEigM)
141 anseigM = eig(A1-(M*Cfr));
142 %Define Lr to be used in the observer
143 % z(t)' = D*z(t) + F*y(t) + G*u(t)
144 % qhu(t) = z(t) - Lr*y(t),
145 %where D=A22-(Lr*A12), F=(D*Lr)+(A21-(Lr*A11)), and y=qm;
146 %either Lr=Q*M, Lr=Q*[M;0], or Lr=Q*[M 0];
147 [mM,nM] = size(M); [mA12,nA12] = size(A12);
148 if (nM == mA12) & (mM == mA22)
149     Lr = Q*M;
150 else
151     %Initialize a zero matrix for putting M in the correct location
152     Lra = zeros(mA22,mA12);
153     if rC1 == mC1 %C1 is full row rank
154         Lra(1:mM,1:nM) = M; Lr = Q*Lra;
155     elseif nC1 == 1 %C1 is mx1, not full row rank
156         %Assign M to the appropriate location for building Lr
157         Lra(:,rowloc) = M;
158         %Calculate Lr
159         Lr = Q*Lra;
160     else %C1 is mxn, not full row rank
161         for i=1:rC1
162             %Assign M to the appropriate location for building Lr
163             Lra(:,rowloc(i)) = M(:,i);
164             %Calculate Lr
165             Lr = Q*Lra;
166         end
167     end
168 end
169 %Confirm the observable eigenvalues match the desired eigenvalues (dEigL)
170 anseigLr = eig(A22-(Lr*A12));
171
172 %Define matrices D and F found in z(t)' (the observer)
173 D = A22 - (Lr*A12); F = (D*Lr) + (A21 - (Lr*A11));
174
175 %Save workspace
176 save(strcat('RedC',num2str(fn(1)), 'L', num2str(fn(2)), 'Rm', num2str(fn(3)), ...
177 'G', num2str(fn(4)), 'SC', num2str(fn(5))))
178 %end RedOrdLTI.Calc.m

```

```

1 function J = EigPlace1(rOsvd,A,C,dEig)
2
3 %Function EigPlace1.m is called by FullOrdLTI_Calc.m, RedOrdLTI_Calc.m.
4
5 %The purpose of EigPlace1.m is to help compute the gain matrix.
6 %***** output matrix C is 1xn, single output *****
7
8 %Description of inputs
9 %rOsvd - the 'true' rank of the observability matrix.
10 %A,C - the coefficient matrices of pair (A1,C1).
11 %dEig - the matrix containing the desired eigenvalues for placing
12 % observable eigenvalues.
13 %The program is terminated when J is found. In this case, J is defined as
14 % -FD^T (appears as matrix M in the calling functions).
15 %*****
16
17 %Use the controller form to find L; find the dual
18 AD = A'; BD = C';
19 %Build the controllability matrix (scriptC)
20 for i=1:rOsvd
21     scriptC(:,i) = (AD^(i-1))*BD;
22 end
23 %Find the inverse of the controllability matrix
24 sCinv = inv(scriptC);
25 %Define q as the last row of sCinv
26 q = sCinv(rOsvd,:);
27
28 %Determine matrix P to reduce (AD,BD) to controller form
29 % P = [q; q*AD; q*(AD^2); ... ; q*(AD^(n-1))], AD is nxn
30 for i=1:rOsvd
31     P(i,:) = q*(AD^(i-1));
32 end
33 %Controller form
34 Adc = P*AD*inv(P); Bdc = P*BD;
35
36 %poly returns the coefficients of the characteristic polynomial
37 % poly(dEigL) = [d1 d2 ... dn+1], d1(x^n)+d2(x^(n-1))+...+(dn+1)
38 pdEig = poly(dEig);
39 %The order of these coefficients needs to be reversed
40 %Build a permutation matrix to reverse the order
41 rI = zeros(rOsvd+1,rOsvd+1);
42 for i=1:rOsvd+1
43     rI((rOsvd+1)-(i-1),i) = 1;
44 end
45 %Reverse the order of the desired coefficients

```

```

46 dcoeff = pdEig*rI;
47
48 %Determine the state feedback in controller form
49 for i=1:rOsvd
50     Fdc(1,i) = (-Adc(rOsvd,i)) - dcoeff(1,i);
51 end
52 %Determine the state feedback
53 FD = Fdc*P;
54 %Return matrix L (L = -FD^T)
55 J = -FD';
56
57 %end      EigPlace1.m

1 function J = EigPlace2(rOsvd,A,C,dEig)
2
3 %Function EigPlace2.m is called by FullOrdLTI_Calc.m, RedOrdLTI_Calc.m.
4
5 %The purpose of EigPlace2.m is to help compute the gain matrix.
6 %***** output matrix C is mxn, multi output *****
7
8 %Description of inputs
9 %rOsvd - the 'true' rank of the observability matrix.
10 %A,C - the coefficient matrices of pair (A1,C1).
11 %dEig - the matrix containing the desired eigenvalues for placing
12 % observable eigenvalues.
13 %The program is terminated when J is found. In this case, J is defined as
14 % -FD^T (appears as matrix M in the calling functions).
15 %*****
16
17 %Use the controller form to find L; find the dual
18 AD = A'; BD = C';
19 %Build the controllability matrix, scriptC
20 %scriptC is a matrix of the form:
21 % scriptC=[b1 b2 Ab1 Ab2 (A^2)b1 (A^2)b2 (A^3)b1 (A^3)b2 (A^4)b1 (A^4)b2]
22 [mBD,nBD] = size(BD); MBD = mBD - 1; NBD = nBD - 1;
23 for i=1:mBD
24     scriptC(:,((nBD*i)-NBD):(nBD*i)) = (AD^(i-1))*BD;
25 end
26
27 %After the for loop is complete, Qns will be a nonsingular matrix
28 % containing the first mBD linearly independent columns of scriptC
29 %Initialize matrix Qns
30 Qns = zeros(mBD,mBD);
31 %Clong is a matrix of the form:

```

```

32 % Clong = [b1 b2;Ab1 Ab2;(A^2)b1 (A^2)b2;(A^3)b1 (A^3)b2;(A^4)b1 (A^4)b2]
33 %If, for example, column b2 of scriptC becomes a column of Qns, then b2
34 % is appropriately placed in Clong; however, if column b2 of scriptC does
35 % not become a column of Qns, then b2 in Clong remains a vector of zeros.
36 % This matrix is meant to simplify the reordering of columns for Cbar.
37 Clong = zeros(mBD*mBD,nBD);
38 %c1 is counter 1; keeps track of the column of Qns being assigned
39 c1 = 1;
40 %This for loop picks off first mBD linearly independent columns of scriptC
41 % If index i is 1 and nBD=2, then index j goes from 1 to 2, testing the
42 % first two columns of scriptC for linear independence, etc.
43 for i=1:mBD
44     for j=((nBD*i)-NBD):(nBD*i)
45         Qns(:,c1) = scriptC(:,j);
46         %Confirm rank of Qns is accurate by checking its singular values
47         [UQns,svdQns0,VQns] = svd(Qns);
48         svdQns = Rounding(svdQns0); rQns = rank(svdQns);
49         %If the rank of Qns is equal to the column number being considered,
50         % keep that column and increase the counter by 1
51         if rQns >= c1
52             c1 = c1+1;
53             Clong(((mBD*i)-MBD):(mBD*i),(j-((nBD*i)-NBD)+1)) = scriptC(:,j);
54             %This break command is included to end the for loop after the
55             % mBD linearly independent columns have been chosen
56             if c1 >= mBD+1
57                 break
58             else end
59         else end
60     end
61     if c1 >= mBD+1
62         break
63     else end
64 end
65
66 %Initialize a vector for the controllability indices
67 mu = zeros(1,nBD);
68 %c2 is counter 2; keeps track of the column of Cbar being assigned
69 c2 = 1;
70 %This for loop builds Cbar, a matrix with the same linearly independent
71 % columns as Qns, but the columns are reordered so that *b1 is listed
72 % first and *b2 is listed second (for example: [b1 Ab1 (A^2)b1 b2 Ab2])
73 %If Clong is defined so its first column contains *b1 vectors and its
74 % second column contains *b2 vectors, for example, this for loop runs
75 % through the first column and then the second column of Clong
76 for i=1:nBD

```

```

77     for j=1:mBD
78         %Rank is checked to make sure only nonzero vectors of Clong are
79         % assigned to Cbar
80         rv = rank(Clong((mBD*j)-MBD):(mBD*j),i);
81         if rv >= 1
82             Cbar(:,c2) = Clong((mBD*j)-MBD):(mBD*j),i);
83             %The last j passed through to this if else statement represents
84             % how many nonzero *b1,*b2,etc. vectors there are individually
85             mu(i) = j; c2 = c2+1;
86         else end
87     end
88 end
89 %Find the inverse of Cbar
90 iCbar = inv(Cbar);
91
92 %Initialize sigma for use in picking out specific rows of iCbar
93 sigma = zeros(1,nBD);
94 %c3 is counter 3; the value of sigma(i)
95 c3 = 0;
96 for i=1:nBD
97     c3 = c3 + mu(i); sigma(i) = c3;
98 end
99 %Pick out specific rows of iCbar using sigma values; initialize matrix q
100 q = zeros(nBD,mBD);
101 for i=1:nBD
102     q(i,:) = iCbar(sigma(i),:);
103 end
104
105 %Build similarity transformation matrix P
106 % P = [q1;q1A;...;q1(A^(mu(1)-1));...]
107 %c4 is counter 4; keeps track of the row of P being assigned
108 c4 = 1;
109 for i=1:nBD
110     for j=1:mu(i)
111         P(c4,:) = q(i,:)*(AD^(j-1)); c4 = c4+1;
112     end
113 end
114 %Find the inverse of P
115 iP = inv(P);
116 %Controller form
117 Adc = P*AD*iP; Bdc = P*BD;
118
119 %Pick out specific rows of Adc and Bdc using sigma values
120 for i=1:nBD
121     Adcm(i,:) = Adc(sigma(i),:); Bdcm(i,:) = Bdc(sigma(i),:);

```

```

122 end
123 %Find the inverse of Bdcm
124 iBdcm = inv(Bdcm);
125
126 %poly returns the coefficients of the characteristic polynomial
127 % poly(dEig) = [d1 d2 ... dn+1], d1(x^n)+d2(x^(n-1))+...+(dn+1)
128 pdEig = poly(dEig);
129 %The order of these coefficients needs to be reversed
130 %Build a permutation matrix to reverse the order
131 rI = zeros(rOsbd+1,rOsbd+1);
132 for i=1:rOsbd+1
133     rI((rOsbd+1)-(i-1),i) = 1;
134 end
135 %Reverse the order of the desired coefficients
136 dcoeff = pdEig*rI; %Note, not interested in last coefficient
137
138 %Build a matrix in multivariable companion form with desired eigenvalues
139 %Initialize W
140 W = zeros(mBD,mBD);
141 for i=1:MBD
142     W(i,i+1) = 1;
143 end
144 for i=1:mBD
145     W(mBD,i) = -dcoeff(i);
146 end
147 %Pick out specific rows of W using sigma values
148 for i=1:nBD
149     Wm(i,:) = W(sigma(i),:);
150 end
151
152 %Determine the state feedback in controller form
153 Fdc = iBdcm*(Wm - Adcm);
154 %Determine the state feedback
155 FD = Fdc*P;
156 %Return matrix L (L = -FD^T)
157 J = -FD';
158
159 %end    EigPlace2.m

```

```

1 function J = LTIDAE_Results(Atilde,Btilde,C,Lf,transAR,transBR,DR,FR,LrR,R, ...
2 transAM,transBM,DM,FM,LrM,Rext,GBext,u,rowused,extend,fn)
3 %Function LTIDAE_Results.m is called by MainLTIObs.m.
4

```

```

5 %The purpose of LTIDAE_Results.m is to
6 %1) Solve the example system
7 %2) Construct the full-order, reduced-order, maximally reduced observers
8 %3) Plot the results
9
10 %Description of inputs
11 %Atilde,Btilde,C - coefficient matrices from the full-order system.
12 %transAR,transBR - coefficient matrices from the reduced-order system.
13 %transAM,transBM - coefficient matrices from the system used to construct
14 % the maximally reduced observer.
15 %DR,FR - matrices found in z(t)' (reduced-order observer).
16 %DM,FM - matrices found in zext(t)' (maximally reduced observer).
17 %Lf,LrR,LrM - gain matrices for full-order, reduced-order, and maximally
18 % reduced observers, respectively.
19 %R,Rext - transformation matrices.
20 %GBext - for defining the extended output equation.
21 %u - symbolically defined control.
22 %extend = 1 indicates an extended output matrix Cext exists, while
23 % extend = 0 indicates an extended output matrix Cext does not exist.
24 %fn - values for name of .mat file in LTIDAE_Results.m.
25
26 %This function may call either LTIDAE_Solve1.m or LTIDAE_Solve2.m.
27
28 %** Note **
29 %Sections of this program are example specific.
30 %***** ****
31
32 %Define a symbolic variable t for the control
33 syms t
34 %Assign switch: save C if onl = 0, save GF if onl = 1
35 onl = fn(6);
36 %Determine the sizes of C, DR, DM, and FM
37 [mC,nC] = size(C);
38 [mDR,nDR] = size(DR); [mDM,nDM] = size(DM); [mFM,nFM] = size(FM);
39 %Example specific system of DAEs
40 % x1(t)' = x2(t)
41 % x2(t)' = K*x1(t) + S*x2(t) + (H^T)*x3(t) + G*u1(t)
42 % 0 = H*x1(t) + u2(t)
43 %K,S,H,G are constant matrices
44 K = [-2 1;1 -2]; S = (1/4)*[1 0;0 1]; H = [1 -1]; G = [1;1];
45 %Time span; keep the number of elements in resulting vectors uniform
46 t0 = 0; tf = 15; tspan = t0:0.02:tf;
47
48 %Symbolically find the needed derivatives of control u
49 [mu,nu] = size(u);

```

```

50 for i=1:3 %This three is example specific and corresponds with the index
51 du(((i-1)*mu)+1:(i*mu),:) = diff(u,i);
52 end
53 %Evaluate u and its derivatives at t0: u(t0), du(t0)
54 %subs substitutes in a numerical value for syms
55 u0 = subs(u,{t},{t0}); du0 = subs(du,{t},{t0});
56 %Build an initial condition vector for x1 and x2, where x=[x1;x2;x3]
57 %x1 is 2x1, x2 is 2x1, and x3 is 1x1
58 x0 = zeros(4,1);
59 %x1(t0) = -pinv(H)*u2(t0), x2(t0) = -pinv(H)*(u2(t0))'
60 x0(1:2,:) = -pinv(H)*u0(2); x0(3:4,:) = -pinv(H)*du0(2);
61 %Determine initial condition for x3
62 %Since the equation for x3 is not a differential equation, x3 can be solved
63 % outside of the program called by ode45, but its initial condition is
64 % still required as an input initial condition for xtilde
65 x3o = -inv(H*(H'))*[H*K*(x0(1:2,:))+H*S*(x0(3:4,:))+H*G*u0(1)+du0(4)];
66 %Define initial conditions for xtilde, xhat, qtilde, and z
67 xtilde0 = [x0;x3o]; %completion (full-order system)
68 xhat0 = [6;7;8;9;10]; %full-order observer
69 qtilde0 = R*xtilde0; %transformed completion (reduced-order system)
70 qhat0 = R*xhat0; %transformed full-order observer
71 quhat0 = qhat0((mC+1):5,:); %unmeasurable components
72 y0 = C*xtilde0; %output
73 z0 = quhat0 - LrR*y0; %reduced-order observer
74
75 %Define a new tolerance level for ode45
76 options = odeset('RelTol',1e-6);
77 if extend == 1 %the extended output matrix Cext exists
78 %Define initial conditions for qtildext and zext
79 qtildext0 = Rext*xtilde0; %transformed completion
80 qhatext0 = Rext*xhat0; %transformed full-order observer
81 quhatext0 = qhatext0((6-mDM):5,:); %unmeasurable components
82 %Initial condition maximally reduced observer
83 if on1 == 0
84 zext0 = quhatext0 - LrM*[y0;GBext*[u0;du0]];
85 else %on1 == 1
86 %Pick appropriate rows of y for extended output
87 [mrowused,nrowused] = size(rowused); count1 = 1;
88 for i = 1:mrowused
89 yext0(count1) = y0(rowused(i)); count1 = count1 + 1;
90 end
91 zext0 = quhatext0 - LrM*[GBext*[u0;du0];yext0];
92 end
93 %Concatenate initial condition vector to pass to LTIDAE_Solve1.m
94 x0long = [x0;xtilde0;xhat0;qtilde0;z0;qtildext0;zext0];

```

```

95      %Find solutions for x, xtilde, xhat, qtilde, z, qtildext, and zext
96      [tlong,xlong] = ode45(@LTIDAE_Solve1,tspan,x0long,options,K,S,H,G, ...
100         Atilde,Btilde,C,Lf,transAR,transBR,DR,FR,LrR,transAM,transBM, ...
101         DM,FM,LrM,GBext,u,du,rowused,fn);
102    else %extend == 0, the extended output Cext does not exist
103        %Concatenate initial condition vector to pass to LTIDAE_Solve2.m
104        x0long = [x0;xtilde0;xhat0;qtilde0;z0];
105        %Find solutions for x, xtilde, xhat, qtilde, and z
106        [tlong,xlong] = ode45(@LTIDAE_Solve2,tspan,x0long,options,K,S,H,G, ...
107         Atilde,Btilde,C,Lf,transAR,transBR,DR,FR,LrR,u,du);
108    end
109
110    %xlong is returned as a ??x(24+mDR+mDM) or a ??x(19+mDR) matrix, depending
111    % on the number of observers being designed
112    %Transpose xlong so XLONG is (24+mDR+mDM)x?? or (19+mDR)x??
113    XLONG = xlong';
114    %Use the appropriate rows of XLONG to help define x
115    %Returned x1 and x2
116    retx = XLONG(1:4,:);
117    %Using the returned time vector, evaluate u and its derivatives for use in
118    % calculating x3 and, if extend = 1, GB (for the extended output)
119    [mtlong,ntlong] = size(tlong);
120    for i=1:mtlong
121        ut = subs(u,{t},{tlong(i)}); dut = subs(du,{t},{tlong(i)});
122        x3(i)=-inv(H*(H'))*[H*K*(retx(1:2,i))+H*S*(retx(3:4,i))+H*G*ut(1)+dut(4)];
123        if extend == 1
124            GB(:,i) = GBext*[ut;dut];
125        else end
126    end
127    %Solution for x, x=[x1;x2;x3]
128    x = [retx;x3];
129    %
130    %Solution for xtilde, the completion
131    xtilde = XLONG(5:9,:);
132    %Solution for xhat, the full-order observer
133    xhat = XLONG(10:14,:);
134    %
135    %Solution for qtilde, the transformed completion using C
136    qtilde = XLONG(15:19,:);
137    %Solution for z, the reduced-order observer
138    z = XLONG(20:(19+mDR),:);
139    %with 'unmeasurable' qtilde
140    y = C*x; qu = z + LrR*y;
141    %qr = [qm;qu]
142    qr = [y;qu];

```

```

137 %Transform qtilde back to xtilde
138 xq = inv(R)*qtilde;
139 %Transform qr back to xqr
140 xqr = inv(R)*qr;
141
142 if extend == 1 %the extended output Cext exists
143     %Solution for qtildext, the transformed completion using Cext
144     qtildext = XLONG((20+mDR):(24+mDR),:);
145     %Solution for zext, the maximally reduced observer
146     zext = XLONG((25+mDR):(24+mDR+mDM),:);
147     %
148     %with 'unmeasurable' qtildext
149     if on1 == 0 %[C;GFprod]
150         qextu = zext + LrM*[y;GB]; qextr = [y;GB;qextu];
151     else %on1 == 1, [GFprod;C]
152         [mrowused,nrowused] = size(rowused); count2 = 1;
153         for i = 1:mrowused
154             yext(count2,:) = y(rowused(i),:); count2 = count2 + 1;
155         end
156         qextu = zext + LrM*[GB;yext]; qextr = [GB;yext;qextu];
157     end
158     %
159     %Transform qtildext back to xtilde
160     xextq = inv(Rext)*qtildext;
161     %Transform qextr back to xextqr
162     xextqr = inv(Rext)*qextr;
163 else end
164
165 %Save workspace
166 save(strcat('ResultsC',num2str(fn(1)), 'L', num2str(fn(2)), 'Rm', ...
167     num2str(fn(3)), 'Ex', num2str(fn(4)), 'SC', num2str(fn(5))))
168 %Plots ****
169 %
170 %The color output is defined by MATLAB: for the graph of a 5x1 vector, the
171 % 1st component is blue, the 2nd component is green, the 3rd component is
172 %  red, the 4th component is cyan, and the 5th component is magenta
173
174 %Plot x, example system's solution, and the completion on the same graph
175 %figure 1
176 figure
177 hold on
178 plot(tlong,x,'LineWidth',2)
179 plot(tlong,xtilde,'--','LineWidth',2)
180 hold off

```

```

181 hgsave(strcat('xandxtildeFOC',num2str(fn(1)), 'L', num2str(fn(2)), 'Rm', ...
182     num2str(fn(3)), 'SC', num2str(fn(5))));  

183 %Plot the difference between x and the completion  

184 %figure 2  

185 figure  

186 plot(tlong,(x-xtilde), 'LineWidth',2)  

187 hgsave(strcat('xminusxtildeFOC',num2str(fn(1)), 'L', num2str(fn(2)), 'Rm', ...
188     num2str(fn(3)), 'SC', num2str(fn(5))));  

189 %Plot x and its full-order observer on the same graph  

190 %figure 3  

191 figure  

192 hold on  

193 plot(tlong,x, 'LineWidth',2)  

194 plot(tlong,xhat, '—', 'LineWidth',2)  

195 hold off  

196 hgsave(strcat('xandxhatFOC',num2str(fn(1)), 'L', num2str(fn(2)), 'Rm', ...
197     num2str(fn(3)), 'SC', num2str(fn(5))));  

198 %Plot the difference between x and its full-order observer  

199 %figure 4  

200 figure  

201 plot(tlong,(x-xhat), 'LineWidth',2)  

202 hgsave(strcat('xminusxhatFOC',num2str(fn(1)), 'L', num2str(fn(2)), 'Rm', ...
203     num2str(fn(3)), 'SC', num2str(fn(5))));  

204 %Plot qtilde and its reduced-order observer on the same graph  

205 %figure 5  

206 figure  

207 hold on  

208 plot(tlong,qtilde, 'LineWidth',2)  

209 plot(tlong,qr, '—', 'LineWidth',2)  

210 hold off  

211 hgsave(strcat('qandqrROC',num2str(fn(1)), 'L', num2str(fn(2)), 'Rm', ...
212     num2str(fn(3)), 'SC', num2str(fn(5))));  

213 %Plot the difference between qtilde and its reduced-order observer  

214 %figure 6  

215 figure  

216 plot(tlong,(qtilde-qr), 'LineWidth',2)  

217 hgsave(strcat('qminusqrROC',num2str(fn(1)), 'L', num2str(fn(2)), 'Rm', ...
218     num2str(fn(3)), 'SC', num2str(fn(5))));  

219 %Plot the difference between x and system qtilde transformed back

```

```

220 %figure 7
221 figure
222 plot(tlong,(x-xq), 'LineWidth',2)
223
224 %Plot x and its reduced-order observer on the same graph
225 %figure 8
226 figure
227 hold on
228 plot(tlong,x, 'LineWidth',2)
229 plot(tlong,xqr,'--','LineWidth',2)
230 hold off
231 hgsave(strcat('xandxqrROC',num2str(fn(1)), 'L', num2str(fn(2)), 'Rm', ...
    num2str(fn(3)), 'SC', num2str(fn(5)))); 
232
233 %Plot the difference between x and its reduced-order observer
234 %figure 9
235 figure
236 plot(tlong,(x-xqr), 'LineWidth',2)
237 hgsave(strcat('xminusxqrROC',num2str(fn(1)), 'L', num2str(fn(2)), 'Rm', ...
    num2str(fn(3)), 'SC', num2str(fn(5)))); 
238
239 if extend == 1 %the extended output Cext exists
240 %Plot qtildext and its maximally reduced observer on the same graph
241 %figure 10
242 figure
243 hold on
244 plot(tlong,qtildext, 'LineWidth',2)
245 plot(tlong,qextr,'--','LineWidth',2)
246 hold off
247 hgsave(strcat('qandqrMRC',num2str(fn(1)), 'L', num2str(fn(2)), 'Rm', ...
    num2str(fn(3)), 'SC', num2str(fn(5)))); 
248
249 %Plot the difference between qtildext and its maximally reduced observer
250 %figure 11
251 figure
252 plot(tlong,(qtildext-qextr), 'LineWidth',2)
253 hgsave(strcat('qminusqrMRC',num2str(fn(1)), 'L', num2str(fn(2)), 'Rm', ...
    num2str(fn(3)), 'SC', num2str(fn(5)))); 
254
255 %Plot the difference between x and system qtildext transformed back
256 %figure 12
257 figure
258 plot(tlong,(x-xextq), 'LineWidth',2)
259
260 %Plot x and its maximally reduced observer on the same graph

```

```

261 %figure 13
262 figure
263 hold on
264 plot(tlong,x,'LineWidth',2)
265 plot(tlong,xextqr,'--','LineWidth',2)
266 hold off
267 hgsave(strcat('xandxqrMRC',num2str(fn(1)), 'L', num2str(fn(2)), 'Rm', ...
    num2str(fn(3)), 'SC', num2str(fn(5))));
```

268

```

269 %Plot the difference between x and its maximally reduced observer
270 %figure 14
271 figure
272 plot(tlong,(x-xextqr),'LineWidth',2)
273 hgsave(strcat('xminusxqrMRC',num2str(fn(1)), 'L', num2str(fn(2)), 'Rm', ...
    num2str(fn(3)), 'SC', num2str(fn(5))));
```

274 else end

275

276 end

277

278 %end LTIDAE_Results.m


```

1 function dxlong = LTIDAE_Solve1(tspan,xlong,K,S,H,G,Atilde,Btilde,C,Lf, ...
    transAR,transBR,DR,FR,LrR,transAM,transBM,DM,FM,LrM,GBext,u,du,rowused,fn)
2
3 %Function LTIDAE_Solve1.m is called by LTIDAE_Results.m when extend = 1
4 % (reduced-order and maximally reduced observers are constructed).
5
6 %The purpose of LTIDAE_Solve1.m is to find solutions for x
7 % (the example system), xtilde (the completion), xhat (the full-order
8 % observer), qtilde (the transformed completion using C), z (the
9 % reduced-order observer), qtildext (the transformed completion using
10 % Cext), and zext (the maximally reduced observer)
11
12 %Description of inputs
13 %tspan – the values of time at which ode45 is evaluating LTIDAE_Solve1.m.
14 %xlong – the initial condition vector.
15 %K,S,H,G – coefficient matrices from the original system of DAEs.
16 %GBext – additional information needed to define the extended output.
17 %u,du – symbolically defined control and its derivatives.
18 %See LTVcircuit.m for description of other inputs.
19
20 %** Note **
21 %Sections of this program are example specific.
22 %*****
```

23

```

24 %Define a symbolic variable t for the control
25 syms t
26 %Assign switch: save C if onl = 0, save GF if onl = 1
27 onl = fn(6);
28 %Variables in equation for dx2/dt (dxlong(3:4,:))
29 Hdagger = (H')*inv(H*(H')); I = eye(2); P = (I - (Hdagger*H));
30 %Evaluating u and its derivatives at tspan: u(tspan), du(tspan)
31 %subs substitutes in a numerical value for syms
32 ut = subs(u,{t},{tspan}); dut = subs(du,{t},{tspan}); udu = [ut;dut];
33 %Calculate the extended output and its size
34 GB = GBext*udu; [mGB,nGB] = size(GB);
35 %The number of state variables vary depending on how many outputs are
36 % 'measurable' or can be calculated by additional methods
37 %Determine the sizes of DR,FR,LrR,DM,FM, and LrM
38 [mDR,nDR] = size(DR); [mFR,nFR] = size(FR); [mLrR,nLrR] = size(LrR);
39 [mDM,nDM] = size(DM); [mFM,nFM] = size(FM); [mLrM,nLrM] = size(LrM);
40
41 %The system to be solved
42 dxlong = zeros((24+mDR+mDM),1);
43 %
44 %x1 of x=[x1;x2]
45 dxlong(1:2,:) = xlong(3:4,:);
46 %x2 of x=[x1;x2]
47 dxlong(3:4,:) = P*K*xlong(1:2,:) + P*S*xlong(3:4,:) + P*G*ut(1) - ...
    Hdagger*dut(4);
48 %
49 %xtilde, completion (full-order observer)
50 dxlong(5:9,:) = Atilde*xlong(5:9,:) + Btilde*udu;
51 %
52 %xhat, full-order observer
53 dxlong(10:14,:) = Atilde*xlong(10:14,:) + Btilde*udu + ...
    Lf*C*[xlong(5:9,:)-xlong(10:14,:)];
54 %
55 %qtilde, completion (transformed system, reduced-order observer)
56 dxlong(15:19,:) = transAR*xlong(15:19,:) + transBR*udu;
57 %
58 %z, reduced-order observer
59 dxlong(20:(19+mDR),:) = DR*xlong(20:(19+mDR),:) + FR*xlong(15:(14+nFR),:) + ...
    (transBR((nLrR+1):5,:)) - LrR*transBR(1:nLrR,:)*udu;
60 %
61 %qtildeext, completion (transformed system, maximally reduced observer
62 dxlong((20+mDR):(24+mDR),:) = transAM*xlong((20+mDR):(24+mDR),:) + transBM*udu;
63 %
64 %zext, maximally reduced observer
65 ytil = C*xlong(5:9,:);

```

```

66 if on1 == 0 %C was saved
67     dxlong((25+mDR) : ((24+mDR)+mDM), :) = DM*xlong((25+mDR) : ((24+mDR)+mDM), :) ...
+ FM*[ytil;GB] + (transBM((nLrM+1):5,:)) - LrM*transBM(1:nLrM,:))*udu;
68 else %on1 == 1, GF was saved
69     [mrowused,nrowused] = size(rowused); count = 1;
70     %Pick appropriate rows of y for extended output
71     for i = 1:mrowused
72         ytilext(count) = ytil(rowused(i)); count = count + 1;
73     end
74     dxlong((25+mDR) : ((24+mDR)+mDM), :) = DM*xlong((25+mDR) : ((24+mDR)+mDM), ...) ...
+ FM*[GB;ytilext] + (transBM((nLrM+1):5,:)) - LrM*transBM(1:nLrM,:))*udu;
75 end
76
77 %end      LTIDAE_Solve1.m

```

```

1 %script DAEManObs.m
2
3 %In script DAEManObs.m, the linear time-invariant system of DAEs is
4 % defined, E(x(t)') + Fx(t) = Bu(t), with output equation y(t) = Cx(t).
5 % Then this observer (and program) takes advantage of the system resulting
6 % from the canonical form of the matrix pencil:
7 %           PEQ\dot{p}(t) + PFQp(t) = PBu(t),
8 % where PEQ=[I 0;0 N], PFQ=[Om 0;0 I] (Om = \Omega), p(t)=[p1(t);p2(t)].
9
10 %The purpose of DAEManObs.m is to
11 % 1) solve for p2(t) explicitly using information from the
12 % characterization of the solution manifold and
13 % 2) observe p1(t) with a full-order observer.
14 %No completions are required.
15
16 %This function calls LTIDAE_SCDerArray.m, FullOrdLTI_Calc.m,
17 % SolveDAEManObs.m.
18
19 %** Note **
20 %This program, particularly matrices P and Q, is example specific.
21 %***** ****
22
23 %Define a symbolic variable s for the matrix pencil
24 syms s
25 %Example specific system of DAEs
26 %x1(t)' = x2(t)
27 %x2(t)' = K*x1(t) + S*x2(t) + (H^T)*x3(t) + G*u1(t)
28 %    0 = H*x1(t) + u2(t)
29 K = [-2 1;1 -2]; S = (1/4)*[1 0;0 1]; H = [1 -1]; G = [1;1];

```

```

30 %Define coefficient matrices E, A = -F, and B from the linear
31 % time-invariant system of DAEs
32 E = [1 0 0 0 0;0 1 0 0 0;0 0 1 0 0;0 0 0 1 0;0 0 0 0 0];
33 A = [0 0 1 0 0;0 0 0 1 0;-2 1 1/4 0 1;1 -2 0 1/4 -1;1 -1 0 0 0];
34 B = [0 0;0 0;1 0;1 0;0 1];
35 %Select output matrix C
36 C = [0 0 1 0 0;0 0 0 1 0]; %Case 1, Ca
37 %C = [0 0 1 -1 0]; %Case 2, Cb
38 %C = [0 0 1 1 0]; %Case 3, Cc
39 %C = [0 0 0 1 0;0 1 0 0 0]; %Case 4, Cd
40 casenum = 1; %the case number (1, 2, 3, or 4)
41 %Confirm the matrix pencil is regular (the system of DAEs is solvable)
42 det(s*E - A);
43 %Only two values of s cause the determinant to equal zero
44 rts = roots([-2 1/2 -2]);
45 %Determine the size of E
46 [mE,nE] = size(E);
47 %Find the derivative array equations without stabilized differentiation
48 [Atilde,Btilde,scriptE,scriptF,scriptB] = LTIDAE_SCDerArray(E,-A,B,0);
49 %Determine scriptG (\mathcal{G}) from characterization of solution manifold
50 [mscriptE,nscriptE] = size(scriptE);
51 [UscriptE,DscriptE0,VscriptE] = svd(scriptE);
52 DscriptE = Rounding(DscriptE0); rscriptE = rank(DscriptE);
53 %Define scriptG
54 scriptG = (UscriptE(:,rscriptE+1:mscriptE))';
55 %d = dimension of solution manifold
56 [mscriptG,nscriptG] = size(scriptG); d = nE - mscriptG;
57
58 %Matrices P and Q come from Scilab version 5.3.3: [Q,P,i1]=pencan(E,A)
59 % Note, scilab returns P,Q for PEQ\dot{p}(t) = PAQp(t) + PBu(t), which is
60 % why earlier A = -F. Also, Scilab uses format(17) instead of format long.
61 Prow1 = [-0.70710678118655 -0.70710678118655 0 0 0];
62 Prow2 = [0 0.000000000000001 0.70710678118655 0.70710678118655 0];
63 Prow3 = [0.12595432509999 -0.12595432509999 -0.34688676974388 ...
    0.34688676974388 -1.23263232428599];
64 Prow4 = [0.69933067332080 -0.69933067332080 0.02162311456102 ...
    -0.02162311456102 0.13074937113084];
65 Prow5 = [0.00740899773506 -0.00740899773506 0.34802394877121 ...
    -0.34802394877121 -0.19436729949907];
66 P = [Prow1;Prow2;Prow3;Prow4;Prow5];
67 %
68 Qrow1 = [-0.70710678118655 0 -0.34397013759505 0.06562687010241 ...
    -0.34692367486636];
69 Qrow2 = [-0.70710678118655 0 0.34397013759505 -0.06562687010242 ...
    0.34692367486636];

```

```

70 Qrow3 = [0 0.70710678118655 0.07029594418665 0.70202835944400 ...
    0.02644844814892];
71 Qrow4 = [0 0.70710678118655 -0.07029594418666 -0.70202835944401 ...
    -0.02644844814893];
72 Qrow5 = [0 0 -1.24308420563197 0.04308004672982 0.19498373845406];
73 Q = [Qrow1;Qrow2;Qrow3;Qrow4;Qrow5];
74
75 %Time span; keep the number of elements in resulting vectors uniform
76 t0 = 0; tf = 15; tspan = t0:0.02:tf;
77 %Calculate invertible block GG (scriptG*scriptF*Q = [0 GG])
78 sGsFQ = scriptG*scriptF*Q;
79 GG = sGsFQ(:,d+1:nE);
80 %Solve for p2(t)
81 [mspan,ntspan] = size(tspan);
82 for i = 1:ntspan
83     ti = tspan(i);
84     p2(:,i) = inv(GG)*scriptG*scriptB*[sin(ti);sin(ti);cos(ti);cos(ti); ...
        -sin(ti);-sin(ti);-cos(ti);-cos(ti)];
85 end
86
87 %The system to be observed resulting from the process to construct the DAE
88 % manifold observer is
89 %          (p1(t)') = -(Omega)*p1(t) + P*B1*u(t)
90 %      \overline{y}(t) = CmObs*p1(t),
91 % where CmObs is defined below and should be full row rank
92 %Canonical system coefficients
93 Emod = P*E*Q; Amod = P*A*Q; Bmod = P*B;
94 %System-to-be-observed coefficients
95 EmObs = Emod(1:d,1:d); AmObs = Amod(1:d,1:d); BmObs = Bmod(1:d,:);
96 %Output matrix for system to be observed
97 C(:,1:d)*Q(1:d,1:d); C(:,d+1:nE)*Q(d+1:nE,1:d);
98 CmObs = C(:,1:d)*Q(1:d,1:d) + C(:,d+1:nE)*Q(d+1:nE,1:d);
99 %Check if the DAE manifold observer can be constructed
100 if norm(CmObs) == 0
101     disp('The DAE manifold observer cannot be constructed.')
102     disp('The norm of \overline{C} is zero.')
103     return
104 else end
105 %Determine the size of CmObs
106 [mC,nC] = size(CmObs);
107 %Determine if output matrix CmObs is full row rank; count keeps track of
108 % which rows of CmObs have already been tested for linear independence
109 count = 1;
110 %Check the rank of output matrix CmObs
111 [UC,svdC0,VC] = svd(CmObs);

```

```

112 %Force elements that should be zero to be zero
113 svdC = Rounding(svdC0); rC = rank(svdC);
114 %If CmObs is full row rank, let Cfr = CmObs: \overline{y}(t) = Cfr*p1(t)
115 if rC == mC
116     Cfr = CmObs; disp('CmObs is full row rank.')
117 %If CmObs is not full row rank, take rC linearly independent rows from
118 % CmObs to define a new matrix Cfr: \overline{yfr}(t) = Cfr*p1(t)
119 else
120     for i=1:rC
121         for j=count:mC
122             Cfr(i,:) = CmObs(j,:);
123             [UCfr,svdCfr0,VCfr] = svd(Cfr); svdCfr = Rounding(svdCfr0);
124             if rank(svdCfr) == i
125                 count = j + 1;
126                 break
127             else end
128         end
129     end
130 disp('In DAEManObs.m, CmObs has been modified so it is now full row rank.')
131 end
132
133 %Build an initial condition vector for x1 and x2, where x=[x1;x2;x3]
134 %x1(t0) = -pinv(H)*u2(t0), x2(t0) = -pinv(H)*(u2(t0) ')
135 x10 = -pinv(H)*[sin(t0)]; x20 = -pinv(H)*[cos(t0)];
136 %Determine initial condition for x3
137 %Since the equation for x3 is not a differential equation, x3 can be solved
138 % outside of the program called by ode45, but its initial condition is
139 % still required as an input initial condition for xtilde
140 x30 = -inv(H*(H'))*(H*K*x10 + H*S*x20 + H*G*sin(t0) + (-sin(t0)));
141 %Concatenate initial condition vector for x(t)
142 x0 = [x10;x20;x30];
143 %Define initial conditions for p1(t) and its observer
144 xhat0 = [6;7;8;9;10]; %full-order observer
145 p0 = inv(Q)*x0; %transformed system
146 p10 = p0(1:d); %just p1(t) from transformed system
147 phat0 = inv(Q)*xhat0; %transformed full-order observer
148 p1hat0 = phat0(1:d); %just p1(t) from observer
149 %Concatenate initial condition vector to pass to SolveDAEManObs.m
150 x0long = [x10;x20;p10;p1hat0];
151
152 %rho is a scalar for building the desired eigenvalue matrix for placing
153 % observable eigenvalues, dEig=rho*eye()
154 rho = -1;
155 %Define fn for saving .mat files
156 % fn = [case #, 99, abs(rho), 99, 99, 99];

```

```

157 %99 is numerical placeholder to identify DAE manifold observer since other
158 % variables lambda, Reduced Observer identifier, Completion identifier,
159 % on1 are irrelevant to this observer
160 fn = [casenum 99 -rho 99 99 99];
161 %Calculate the gain matrix Lf (for observable eigenvalue placement)
162 [Lf] = FullOrdLTI_Calc(AmObs,Cfr,rho,fn);
163 %Define a new tolerance level for ode45
164 options = odeset('RelTol',1e-6);
165 %Find solutions for x1(t), x2(t), p1(t) and its observer
166 [tlong,xlong] = ...
    ode45(@SolveDAEManObs,tspan,x0long,options,K,S,H,G,AmObs,BmObs,Cfr,Lf);
167
168 %Transpose xlong
169 XLONG = xlong';
170 %Use the appropriate rows of XLONG to help define x
171 %Returned x1 and x2
172 retx = XLONG(1:4,:);
173 %Evaluate u and its derivatives at time ti for use in calculating x3
174 for i=1:tspan
175     ti = tspan(i);
176     u = [sin(ti);sin(ti)];
177     du = [cos(ti);cos(ti);-sin(ti);-sin(ti);-cos(ti);-cos(ti)];
178     x3(1,i) = -inv(H*(H'))*[H*K*(retx(1:2,i)) + H*S*(retx(3:4,i)) + ...
        H*G*u(1) + du(4)];
179 end
180 %Solution for x, x=[x1;x2;x3]
181 x = [retx;x3];
182 %Solution for p, the transformed system
183 p = [XLONG(5:6,:);p2];
184 %Transform p back to x
185 xp = Q*p;
186 %phat = [p1hat;p2];
187 phat = [XLONG(7:8,:);p2];
188 %Transform phat back to xhat
189 xhat = Q*phat;
190
191 %Save workspace
192 save(strcat('DAEManObsC',num2str(fn(1)), 'Rm', num2str(fn(3))))
193
194 %Plots ****
195 %
196 %The color output is defined by MATLAB: for the graph of a 5x1 vector, the
197 % 1st component is blue, the 2nd component is green, the 3rd component is
198 % red, the 4th component is cyan, and the 5th component is magenta
199

```

```

200 %Plot p and its DAE manifold observer on the same graph
201 %Figure 1
202 figure
203 hold on
204 plot(tlong,p,'LineWidth',2)
205 plot(tlong,phat,'--','LineWidth',2)
206 hold off
207 hgsave(strcat('pandphatC',num2str(fn(1)), 'Rm', num2str(fn(3))));%
208
209 %Plot the difference between p and its DAE manifold observer
210 %Figure 2
211 figure
212 plot(tlong,(p-phat),'LineWidth',2)
213 hgsave(strcat('pminusphatC',num2str(fn(1)), 'Rm', num2str(fn(3))));%
214
215 %Plot x and its DAE manifold observer on the same graph
216 %Figure 3
217 figure
218 hold on
219 plot(tlong,x,'LineWidth',2)
220 plot(tlong,xhat,'--','LineWidth',2)
221 hold off
222 hgsave(strcat('xandxhatC',num2str(fn(1)), 'Rm', num2str(fn(3))));%
223
224 %Plot the difference between x and its DAE manifold observer
225 %Figure 4
226 figure
227 plot(tlong,(x-xhat),'LineWidth',2)
228 hgsave(strcat('xminusxhatC',num2str(fn(1)), 'Rm', num2str(fn(3))));%
229
230 %end      DAEManObs.m

```

```

1 function dxlong = SolveDAEManObs(tspan,xlong,K,S,H,G,AmObs,BmObs,Cfr,Lf)
2
3 %Function SolveDAEManObs.m is called by DAEManObs.m.
4
5 %The purpose of SolveDAEManObs.m is to find solutions for x1, x2 (the
6 % example system), p1 (from the canonical system), and p1hat (the estimate
7 % of p1)
8
9 %Description of inputs
10 %tspan – the values of time at which ode45 is evaluating SolveDAEManObs.m.
11 %xlong – the initial condition vector.
12 %K,S,H,G – coefficient matrices from the original system of DAEs.
13 %AmObs,BmObs,Cfr – coefficient matrices from the DAE manifold observer.

```

```

14 %Lf - gain matrix.
15
16 %** Note **
17 %Sections of this program are example specific.
18 %*****
19
20 %Let t designate tspan
21 t = tspan;
22 %Variables in equation for dx2/dt (dxlong(3:4,:))
23 Hdagger = (H')*inv(H*(H')); I = eye(2); PH = (I - (Hdagger*H));
24 %Evaluating u and its derivatives at t
25 u = [sin(t);sin(t)]; du = [cos(t);cos(t);-sin(t);-sin(t);-cos(t);-cos(t)];
26
27 %The system to be solved
28 dxlong = zeros(8,1);
29 %
30 %x1 of x=[x1;x2]
31 dxlong(1:2,:) = xlong(3:4,:);
32 %x2 of x=[x1;x2]
33 dxlong(3:4,:) = PH*K*xlong(1:2,:) + PH*S*xlong(3:4,:) + PH*G*u(1) - ...
    Hdagger*du(4);
34 %
35 %p1 from canonical system
36 dxlong(5:6,:) = AmObs*xlong(5:6,:) + BmObs*u;
37 %
38 %p1hat
39 dxlong(7:8,:) = AmObs*xlong(7:8,:) + BmObs*u + ...
    Lf*Cfr*[xlong(5:6,:)-xlong(7:8,:)];
40
41 %end      SolveDAEManObs.m

```

B.2 Chapter 6 Code

```

1 %script RunLTV.m, main program for linear time-varying system of DAEs
2 % (Chapter 6, circuit example) in which the user defines his inputs.
3
4 %RunLTV.m passes the information to LTVcircuit.m, a main function that
5 % calls other programs for calculating either a full-order, reduced-order,
6 % or maximally reduced observer.
7 %MATLAB command tic toc is included to monitor run time
8 %*****
9 tic
10 %Step 1: The state vector is nx by 1; define nx, an example specific value
11 nx = 6;

```

```

12
13 %Step 2: Define an output matrix C
14 %Case 1, output matrix Ca
15 C = [0 1 0 0 0 0;0 0 1 0 0 0;0 0 0 1 0 0;0 0 0 0 0 1]
16 %Case 2, output matrix Cb
17 %C = [0 0 1 0 0 0;0 0 0 0 0 1]
18 %Case 2, output matrix Cc (positive resistors only)
19 %C = [0 0 1 0 0 1]
20 %Case 3, output matrix Cd
21 %C = [0 0 1 0 0 0;0 0 0 1 0 0]
22 %Case 4, output matrix Ce (positive resistors only)
23 %C = [1 0 0 0 0 0;0 0 0 1 0 0]
24 %Case 4, output matrix Cf
25 %C = [0 0 0 1 0 0]
26
27 %Step 3: Choose an observer to construct
28 % obs = 1 for the full-order observer
29 % obs = 2 for the reduced-order observer
30 % obs = 3 for the maximally reduced observer
31 obs = 1
32
33 %Step 4: Choose a completion for constructing the observer and for finding
34 %a solution of the LTV DAE
35 % comp = 1 for the stabilized least squares completion (SLSC)
36 % comp = 2 for the alternative stabilized completion (ASC)
37 %For the least squares completion (LSC), define \Lambda as a zero matrix in
38 %the programs for the SLSC and then let comp = 1
39 comp = 1
40
41 %Step 5: Select diagonal entries of stabilization parameter matrix
42 %Stabilized Least Squares Completion, \Lambda = {11,12,13,14,15,16}
43 SPML = [1.2;1.4;1.6;1.8;2.0;2.2]
44 %Alternative Stabilized Completion, \Lambda = {11,12,13,14}
45 %SPML = [4.0;3.5;3.0;2.5]
46 %Least Squares Completion, \Lambda = {0,0,0,0,0,0}
47 %SPML = [0;0;0;0;0;0]
48
49 %Step 6: Define the initial time (t0) and the final time (tf) for the time
50 % interval used when calling ode45
51 t0 = 0, tf = 45
52
53 %Step 7: Define s,m,q in S0 = sI, M = mI, and QL = qI - (C^T)MC (or AR12
54 % for C) when constructing the gain matrix
55 s = 0.1, m = 100, q = 0.1
56

```

```

57 %Step 8: Assign switches
58 %1)Decide between positive resistors and negative resistors
59 % On/off switch for Pos/Neg resistor code
60 % Define 'on1' as 0 if positive resistor code should run
61 % Define 'on1' as 1 if negative resistor code should run
62 on1 = 0;
63 %2)If constructing the full-order observer (obs = 1) using the
64 % alternative stabilized completion (comp = 2), choose between
65 % symbolically defined method (SYM) and smooth decomposition method (SDM)
66 %Note, if SDM is chosen, MATLAB version must include Control System Toolbox
67 % Set on2 = 0 to use SYM
68 % Set on2 = 1 to use SDM (lyap command)
69 on2 = 0;
70 %3)If on2 = 1
71 % Set on3 = 0 to use MATLAB's svd command to find T2 and Z1T
72 % Set on3 = 1 to use SDM to find T2 and Z1T
73 on3 = 0;
74 %4)Four-criteria check for \Lambda selection
75 % Set on4 = 0 to skip check of criteria 2 and 3
76 % Set on4 = 1 to check criteria 2 and 3
77 on4 = 0;
78 %5)Observability check
79 %Note, if on5 = 1 and the system is not observable, MATLAB version must
80 % include Control System Toolbox
81 % Set on5 = 0 to skip observability check
82 % Set on5 = 1 to check observability (may require lyap command)
83 on5 = 0;
84 %6)Observer construction
85 % Set on6 = 0 to construct observers
86 % Set on6 = 1 if only on4 = 1 or on5 = 1 checks should be implemented
87 on6 = 0;
88 %7)Moving horizon
89 % Set on7 = 0 to skip moving horizon/reset of SL from the Riccati equation
90 % Set on7 = 1 to reset SL at time treset
91 on7 = 0; treset = 10;
92 %Concatenate switches vector
93 sv = [on1;on2;on3;on4;on5;on6;on7];
94
95 %Step 9:
96 %If constructing the maximally reduced observer (obs = 3)
97 if obs == 3
98     ExtendOutput(nx,C,obs,comp,SPML,t0,tf,treset,s,m,q,sv)
99 else %otherwise (obs = 1, obs = 2)
100    LTVcircuit(nx,C,0,0,obs,comp,SPML,t0,tf,treset,s,m,q,sv,0,0)
101 end

```

```

102 toc
103 %end      RunLTV.m

1 function J = ...
    LTVcircuit(nx,C,Cext,Cr,obs,comp,SPML,t0,tf,treset,s,m,q,sv,rowcount,rowsC)
2
3 %Function LTVcircuit.m is called by either RunLTV.m or ExtendOutput.m.
4
5 %Function LTVcircuit.m does the following:
6 %1) checks Criteria 2 and 3 from \Lambda Selection;
7 %2) calls WoRank.m to check the observability of the chosen completion and
8 %   output matrix;
9 %3) defines initial conditions and calls LTVDAE_Solveplus.m to find
10 %   solutions for the chosen completion and observer;
11 %4) transforms the results returned from LTVDAE_Solveplus.m if necessary
12 %   and produces a variety of plots.
13
14 %Description of inputs
15 %nx -the number of state variables.
16 %C -output matrix. Note, only constant output matrices have been tested.
17 % Trying a time-varying output matrix is an item for future research.
18 %Cext -the submatrix of C used to extend the output matrix when
19 %   constructing the maximally reduced observer (obs = 3). Cext is input as
20 %   zero for the full-order (obs = 1) and reduced-order (obs = 2) observers.
21 %Cr -row(s) linearly independent from the extended output matrix to make
22 %   the transformation matrix R ( $\overline{R}$ ) nonsingular when obs = 3. Cr
23 %   is input as zero for obs = 1,2.
24 %obs -designates which observer is being constructed (1 for FO, 2 for
25 %   RO, 3 for MR).
26 %comp -designates which completion is being used (1 for (S)LSC, 2 for ASC).
27 %SPML -diagonal entries for stabilization parameter matrix \Lambda.
28 %t0,tf -initial, final times in the time interval used when calling ode45.
29 %treset -time at which SL(t) is reset for moving horizon.
30 %s,m,q -values used when constructing the gain matrix.
31 %sv -switch vector (on/off identifiers to let MATLAB know what section of
32 %   code to run).
33 %rowcount -the number of rows from C that have been used to extend the
34 %   output matrix when obs = 3. rowCount is input as zero for obs = 1,2.
35 %rowsC -the vector containing the row numbers of the saved rows of output
36 %   matrix C when obs = 3. rowsC is input as zero for obs = 1,2.
37
38 %This program is currently set up for the LTV circuit example, Chapter 6.
39 %Derivatives of time-varying matrices were determined outside of this
40 %   function to eliminate finding symbolic derivatives within LTVcircuit.m.
41 %*****

```

```

42
43 %Assign switch: positive resistor if on1 = 0, negative resistor if on1 = 1
44 on1 = sv(1);
45 %Determine the size of C
46 [mC,nC] = size(C);
47 %Define the number of unmeasurable/unknown components nu as well as mmodC
48 %Note, mmodC (m = number of rows, modC = extended output matrix), from the
49 % size of the extended output matrix, is defined as zero for obs = 1,2
50 if obs == 1
51     nu = nx; mmodC = 0;
52 elseif obs == 2
53     nu = nx - mC; mmodC = 0;
54 else %obs == 3
55     %GF = scriptG*scriptF was determined symbolically using MATLAB and Maple;
56     % this GF is example specific
57     %GF0 = GF(t0) is used to determine R (\overline{R}) at the initial time for
58     % calculating initial conditions qtilde0 and qhat0
59     if on1 == 0 %Positive Resistors
60         GF0 = [-1 1 0 -4-2*sin(t0) 0 0;1 0 0 0 -2-sin(t0) 0;1 0 0 0 0 0;
61             -1/3*sin(1/3*t0) 0 0 -1 1 -1];
62     else %on1 == 1, Negative Resistors
63         GF0 = [-1 1 0 4+2*sin(t0) 0 0;1 0 0 0 2+sin(t0) 0;1 0 0 0 0 0;
64             -1/3*sin(1/3*t0) 0 0 -1 1 -1];
65     end
66 %Build the extended output matrix at the initial time and compute nu
67 modC0 = [GF0;Cext]; [mmodC,nmodC] = size(modC0); nu = nx - mmodC;
68 end
69 %Time span; keep the number of elements in resulting vectors uniform
70 tspan = t0:0.01:tf; [mtspan,ntspan] = size(tspan);
71
72 %*****Criteria 2 and 3 of four-criteria check for \Lambda selection*****
73 %Criteria 2 and 3 of four-criteria check for \Lambda selection
74 on4 = sv(4);
75 if on4 == 1 %Check criteria 2 and 3
76 %Find the completion and
77 %1) plot the real part of each eigenvalue to check if negative;
78 %2) plot the norms of \widetilde{A}, \widetilde{B}, and C to check for
79 % bounded matrices. Note, qB returned by ASCcoeff is not \widetilde{B}
80 % but is the product of \widetilde{B} and the input vector.
81 for i = 1:ntspan
82     if comp == 1
83         [qA,qdA,qB] = SLSCcoeff(tspan(i),nx,C,Cext,Cr,obs,SPML,sv);
84     else %comp == 2
85         [qA,qdA,qB] = ASCcoeff(tspan(i),nx,C,Cext,Cr,obs,SPML,sv);
86     end

```

```

87 %Calculate and store the eigenvalues of qA (6 values is example specific)
88 %EVV contains the eigenvectors, EVE is a diagonal matrix with entries equal
89 % to the eigenvalues of qA
90 [EVV,EVE] = eig(qA);
91 eigK1(i) = EVE(1,1); eigK2(i) = EVE(2,2); eigK3(i) = EVE(3,3);
92 eigK4(i) = EVE(4,4); eigK5(i) = EVE(5,5); eigK6(i) = EVE(6,6);
93 %Calculate and store the norms
94 NormA(i) = norm(qA); NormB(i) = norm(qB); NormC(i) = norm(C);
95 %Include norm command check: calculate and store the largest singular
96 % values of qA and qB
97 [UA,DA,VA] = svd(qA); sinvalqA1(i) = DA(1,1);
98 [UB,DB,VB] = svd(qB); sinvalqB1(i) = DB(1,1);
99 end
100
101 %Criterion 3: Plot the eigenvalues of qA (\widetilde{A})
102 figure
103 hold on
104 plot(tspan,real(eigK1)), plot(tspan,real(eigK2)), plot(tspan,real(eigK3))
105 plot(tspan,real(eigK4)), plot(tspan,real(eigK5)), plot(tspan,real(eigK6))
106 hold off
107 hgsave('eigwtilA');
108
109 %Criterion 2: Plot the norms of qA (\widetilde{A}), qB (\widetilde{B})
110 %(norms of qA,qB are plotted with their respective largest singular values)
111 figure
112 hold on
113 plot(tspan,NormA), plot(tspan,sinvalqA1)
114 hold off
115 hgsave('normwtilA');
116 figure
117 hold on
118 plot(tspan,NormB), plot(tspan,sinvalqB1)
119 hold off
120 hgsave('normwtilB');
121
122 %Note, output matrix C is also supposed to be bounded and is for this
123 % example since it is constant
124 figure
125 plot(tspan,NormC)
126 hgsave('normC');
127
128 display('Criteria 2 and 3 computations are completed')
129 else end %else on4 == 0, Do not check criteria 2 and 3
130 %*****
```

```

132 %*****
133 %Check observability for the chosen completion and output matrix
134 %If the (observability) Gramian's rank equals the number of unmeasurable/
135 % unknown components by some finite time, then the system is observable
136 on5 = sv(5);
137 if on5 == 1 %Perform observability check
138 %Define Gramian at t0, the initial condition
139 %Wo0 has been defined as a vector rather than a matrix since ode45 accepts
140 % only an initial condition vector
141 Wo0 = zeros(nu^2,1);
142 %Define the state transition matrix at t0, the initial condition
143 Phi0 = eye(nu);
144 %Reshape Phi0 to be passed through to WoRank.m (called by ode45)
145 Phi0R = reshape(Phi0,nu^2,1);
146 %On/off switch for checking unobservable system. onU = 0, switch is off.
147 % Also, the 2 following onU in inputs for WoRank.m is arbitrary
148 % placeholder when onU = 0
149 onU = 0;
150 %Concatenate initial condition vector to pass to WoRank.m
151 WonPhi = [Wo0;Phi0R];
152 %Define a new tolerance level for ode45
153 options = odeset('RelTol',1e-9);
154 %Find solutions for Wo
155 [tlongWo,Wolong] = ...
    ode45(@origWoRank,tspan,WonPhi,options,nx,mmodC,C,Cext,Cr, ...
    obs,comp,SPML,sv,onU,2);
156 %Determine the size of tlongWo
157 [mtnlongWo,ntnlongWo] = size(tlongWo);
158 %Wolong is returned as an mtnlongWo by 2*nu^2 matrix
159 %Transpose Wolong so WoLONG is 2*nu^2 by mtnlongWo
160 WoLONG = Wolong';
161 %Wo, Phi are returned in vector form
162 Wov = WoLONG(1:nu^2,:); Phiv = WoLONG(nu^2+1:2*(nu^2),:);
163 %Reshape WoLONG as nu by nu matrices for mtnlongWo values of time
164 Wo = reshape(Wov,nu,nu,mtnlongWo); Phi = reshape(Phiv,nu,nu,mtnlongWo);
165
166 %Find the rank of Wo at each value of time > 0 using the svd
167 for i = 2:mtnlongWo
168     [UWo,DWo0,VWo] = svd(Wo(:,:,i));
169     %Assign 0 to the singular values that should be recognized as 0 by MATLAB
170     DWo = Rounding(DWo0);
171     %Calculate the rank of DWo
172     rDWo(i) = rank(DWo);
173 end
174 %Plot the Gramian's rank to view the results of the observability check

```

```

175 figure
176 plot(tspan,rDWo,'d')
177 hgsave('rankWo');
178
179 %Compute the norm of Phi
180 for i = 1:mtlongWo
181     N(i) = norm(Phi(:,:,i)); [UPhi,DPhi,VPhi] = svd(Phi(:,:,i));
182 end
183 %Plot the norm of Phi
184 figure
185 plot(tspan,N)
186 hgsave('normPhi');
187
188 %If the Gramian's rank is less than the number of unmeasurable/unknown
189 % components at tf = 45 (example specific), check for unobservable block
190 %Note, MATLAB version must include Control System Toolbox for this check -
191 % the lyap command is required
192 %Assign the rank at tf=45
193 r45 = rDWo(4501);
194 if r45 < nu
195 %Turn switch on
196     onU = 1;
197 %Initial condition matrices from Wo, Phi at t0
198     Wo1 = Wo(:,:,1); Phil = Phi(:,:,1);
199 %Reshape nu by nu matrices into nu^2 by 1 vectors for ode45
200     Wo1R = reshape(Wo1,nu^2,1); PhilR = reshape(Phil,nu^2,1);
201 %Define initial condition for unobservable system's state transition matrix
202     x30 = eye(nu-r45); x30R = reshape(x30,(nu-r45)^2,1);
203 %Concatenate initial condition vector to pass to WoRank.m
204     WonPhil = [Wo1R;PhilR;x30R];
205 %Find solutions for Wo
206     [tlongWo1,Wolong1] = ...
207         ode45(@origWoRank,tspan,WonPhil,options,nx,mmodC,C, ...
208             Cext,Cr,obs,comp,SPML,sv,onU,r45);
209     WoLONG1 = Wolong1'; [mtlongWo1,ntlongWo1] = size(tlongWo1);
210 %Wo, x3 are returned in vector form
211     Wolv = WoLONG1(1:nu^2,:); x3v = WoLONG1(2*(nu^2)+1:2*(nu^2)+(nu-r45)^2,:);
212 %Reshape
213     Wo1m = reshape(Wolv,nu,nu,mtlongWo1); x3 = ...
214         reshape(x3v,nu-r45,nu-r45,mtlongWo1);
215 %Initialize an (m,n,k) matrix for storing the basis vectors of N(Wo) (the
216 % Gramian's nullspace) at each time t; this information is used in
217 % function CompUnobs.m
218     NullWo = zeros(nu,nu,mtlongWo1);
219 %Since the rank of the Gramian may not be constant, assign a placeholder

```

```

217 % vector so NullWo remains nu by nu
218 nvector = zeros(nu,1);
219 for i = 1:mtlongWo1
220 %Compute the norm of x3
221 Nx3(i) = norm(x3(:,:,i));
222 %Find basis vectors of N(Wo); Woholder is an (m,n) matrix
223 Woholder = null(Wo1m(:,:,i));
224 %If the number of basis vectors is less than nu, assign placeholders
225 [mnull,nnull] = size(Woholder);
226 if nnull < nu
227 for j = (nnull+1):nu
228 Woholder(:,j) = nvector;
229 end
230 else end %else nnull == nu
231 %Assign (m,n) matrix Woholder to (m,n,k) matrix NullWo
232 NullWo(:,:,:,i) = Woholder;
233 end
234 %Plot the norm of the unobservable system's state transition matrix
235 figure
236 plot(tspan,Nx3)
237 hgsave('normPhiunobs');
238 else end %else r45 == nu
239
240 save(strcat('OBSDETINFO'))
241 else end %else on5 == 0, Do not perform observability check
242 %*****
243 %*****
244 %*****
245 %Observer Construction
246 on6 = sv(6);
247 if on6 == 0 %Construct observer
248 %Define an initial condition for matrix SL from the Riccati equation (RE)
249 % used in constructing the gain matrix
250 SL0mbm = s*eye(nu);
251 %Reshape the matrix as a vector since ode45 accepts only one initial
252 % condition vector
253 SL0 = reshape(SL0mbm,nu^2,1);
254
255 %Define initial conditions for the completion and observer
256 xtilde0 = [0;0;0;0;0;-16/5]; %completion
257 xhat0 = [1;2;3;4;5;6]; %observer
258 %Concatenate initial condition vector x0long to pass to LTVDAE_Solveplus.m
259 if obs == 1
260 %x0long = [completion;observer;RE]
261 x0long = [xtilde0;xhat0;SL0];

```

```

262 elseif obs == 2
263 %Since C is constant, transformation matrix R = [C;Cr] is as well
264 %Let Cr equal the transpose of the basis vectors for N(C)
265 [UC,svdC,VC] = svd(C);
266 %Initialize transformation matrix R
267 R = zeros(nC);
268 %Define the first mC rows of R as C
269 R(1:mC,:) = C;
270 %Define the remaining rows of R (Cr) as the transpose of the last nx-mC
271 % columns of VC from the svd
272 R(mC+1:nx,:) = VC(:,mC+1:nx)';
273 %Define initial conditions for transformed systems
274 qtilde0 = R*xtilde0; %for the transformed completion
275 qhat0 = R*xhat0; %for the transformed observer
276 quhat0 = qhat0(mC+1:nx,:); %for the unmeasurable components
277 %A change of variable is required, z0 = quhat0 - V20*y0
278 y0 = C*xtilde0;
279 %Calculate V20 = (1/2)*SL0mbm*(K^T)*M, the gain matrix at initial time t0
280 M = m*eye(mC);
281 if comp == 1
282     [qA,qdA,qB] = SLSCcoeff(0,nx,C,Cext,Cr,obs,SPML,sv);
283 else %comp == 2
284     [qA,qdA,qB] = ASCcoeff(0,nx,C,Cext,Cr,obs,SPML,sv);
285 end
286 %Assign AR12 to K
287 K = qA(1:mC,mC+1:nx); V20 = (1/2)*SL0mbm*(K')*M;
288 %Initial condition for reduced-order observer (ROO)
289 z0 = quhat0 - V20*y0;
290 %x0long = [completion;transformed completion;ROO;RE]
291 x0long = [xtilde0;qtilde0;z0,SL0];
292 else %obs == 3
293 %Define transformation matrix R (\overline{R}) at time t0
294 R0 = [modC0;Cr];
295 %Confirm R at t0 is nonsingular
296 rR = rank(R0);
297 if rR == nx
298 else
299     %If the transformation matrix is singular, end program
300     return
301 end
302 %Define initial conditions for transformed systems
303 qtilde0 = R0*xtilde0; %for the transformed completion
304 qhat0 = R0*xhat0; %for the transformed observer
305 quhat0 = qhat0(mmodC+1:nx,:); %for the unmeasurable components
306 %A change of variable is required, z0 = quhat(:,i) - V2*[GB*[V;dV];ye(:,i)]

```

```

307      y0 = C*xtilde0;
308 %Save the rows of y0 corresponding with the rows of C saved for
309 % extending the output matrix
310     for i = 1:(rowcount - 1)
311         ye0(i,:) = y0(rowsC(i),:);
312     end
313 %GB0, evaluating GB at t0
314     GB0 = [0 0 0;0 0 0;-1 0 0;0 3+cos(1/3*t0) 0];
315 %[V(t0);dV(t0)] will be used in constructing the output for the extended
316 % output equation
317     V0 = 4*cos(2*t0)*sin(t0/5);
318     d1V0 = -8*sin(2*t0)*sin(t0/5)+(4/5)*cos(2*t0)*cos(t0/5);
319     d2V0 = -(404/25)*cos(2*t0)*sin(t0/5)-(16/5)*sin(2*t0)*cos(t0/5);
320     dV0 = [d1V0;d2V0];
321 %Calculate V20 = (1/2)*SL0mbm*(K^T)*M, the gain matrix at initial time t0
322     M = m*eye(mmodC);
323     if comp == 1
324         [qA,qdA,qB] = SLSCcoeff(0,nx,C,Cext,Cr,obs,SPML,sv);
325     else %comp == 2
326         [qA,qdA,qB] = ASCcoeff(0,nx,C,Cext,Cr,obs,SPML,sv);
327     end
328 %Assign AR12 to K
329     K = qA(1:mmodC,mmodC+1:nx); V20 = (1/2)*SL0mbm*(K')*M;
330 %Initial condition for maximally reduced observer (MRO)
331     z0 = quhat0 - V20*[V0;dV0];ye0];
332 %x0long = [completion;transformed completion;MRO;RE]
333     x0long = [xtilde0;qtilde0;z0;SL0];
334 end
335
336 %Moving horizon, SL reset
337 on7 = sv(7);
338 if on7 == 1 %Reset SL(t) at t = treset to SL(t0)
339 %Define time intervals
340     tspan1 = 0:0.01:treset; tspan2 = treset:0.01:tf;
341 %Determine length of tspan1 and tspan2
342     [mtspan1,ntspan1] = size(tspan1); [mtspan2,ntspan2] = size(tspan2);
343 %Define a new tolerance level for ode45
344     options = odeset('RelTol',1e-9);
345 %Find solutions for the completion and observer on tspan1
346     [tlong1,xlong1] = ode45(@LTVDAE_Solveplus,tspan1,x0long,options,nx, ...
347     mmodC,C,Cext,Cr,obs,comp,SPML,m,q,sv,rowcount,rowsC);
348 %Define initial conditions at reset time from the returned results
349     XLONG = xlong1'; xtilde0 = XLONG(1:nx,ntspan1);
350     qtilde0 = XLONG(nx+1:2*nx,ntspan1); z0 = XLONG(2*nx+1:3*nx-mC,ntspan1);
351 %Reset SL(reset) value to SL(t0)

```

```

351     PL0mbm = s*eye(nu); PL0 = reshape(PL0mbm,nu^2,1);
352 %Concatenate initial condition vector
353     x0long = [xtilde0;qtilde0;z0;PL0];
354 %Find solutions for the completion and observer on tspan2
355     [tlong2,xlong2] = ode45(@LTVDAE_Solveplus,tspan2,x0long,options,nx, ...
356     mmodC,C,Cext,Cr,obs,comp,SPML,m,q,sv,rowcount,rowsC);
357 %Assign remaining returned results to vector XLONG
358     XLONG(:,ntspan1:ntspan1+ntspan2-1) = xlong2';
359 else %on7 == 0, No moving horizon
360     %Define a new tolerance level for ode45
361     options = odeset('RelTol',1e-9);
362 %Find solutions for the completion and observer
363     [tlong,xlong] = ode45(@LTVDAE_Solveplus,tspan,x0long,options,nx,mmodC, ...
364     C,Cext,Cr,obs,comp,SPML,m,q,sv,rowcount,rowsC);
365 %xlong is returned as an ntspan by ?? matrix
366 %Transpose xlong so XLONG is ?? by ntspan
367     XLONG = xlong';
368 end
369
370 %Use the appropriate rows of XLONG to define x (xtilde)
371 x = XLONG(1:nx,:);
372 %Calculate the output required for constructing the reduced-order and
373 % maximally reduced observers. This step is unnecessary if data is
374 % available from a physical system.
375 if obs == 2
376     y = C*x;
377 elseif obs == 3
378     y = C*x;
379 %Save the rows of y corresponding with the rows of C saved for
380 % extending the output matrix
381     for i = 1:(rowcount - 1)
382         ye(i,:) = y(rowsC(i),:);
383     end
384 else end %else obs == 1
385
386 %Determine the observer in terms of the original state variable
387 if obs == 1
388 %Use the appropriate rows of XLONG to define xhat
389     xhat = XLONG(nx+1:2*nx,:);
390 %Riccati equation's SL is defined here in case 'initial condition' is
391 % needed for starting at time later than t0
392     SL = reshape(XLONG(2*nx+1:8*nx,:),nu,nu,ntspan);
393 else %obs == 2 or obs == 3
394 %Define the number of rows in the (extended) output matrix
395     if obs == 2

```

```

394         mom = mC;
395     else %if obs == 3
396         mom = mmodC;
397     end
398 %Use the appropriate rows of XLONG to define qtilde, the transformed
399 % completion
400     qtilde = XLONG(nx+1:2*nx,:);
401 %Solution for z, the transformed reduced-order observer
402     z = XLONG(2*nx+1:3*nx-mom,:);
403 %Riccati equation's SL
404     SL = reshape(XLONG(3*nx-mom+1:3*nx-mom+nu^2,:),nu,nu,ntspan);
405 %Define M used in constructing the gain matrix
406     M = m*eye(mom);
407 %Initialize quhat, the observed unmeasurable components
408     quhat = zeros(nu,ntspan);
409     for i = 1:ntspan
410         %Let ti = tspan(i) for formatting in LaTeX
411         ti = tspan(i);
412         %Determine the coefficient matrices of the completion at the time
413         % values in tspan
414         if comp == 1
415             [qA,qdA,qB] = SLSCcoeff(ti,nx,C,Cext,Cr,obs,SPML,sv);
416         else %comp == 2
417             [qA,qdA,qB] = ASCcoeff(ti,nx,C,Cext,Cr,obs,SPML,sv);
418         end
419         %Assign AR12 to K and calculate the gain matrix
420         K = qA(1:mom,mom+1:nx); V2 = (1/2)*SL(:,:,i)*(K')*M;
421         if obs == 2
422             %Calculate quhat at the time values in tspan
423             quhat(:,i) = V2*y(:,i) + z(:,i);
424             %check1 will be used to confirm x-inv(R)*qtilde=0
425             check1(:,i) = inv(R)*qtilde(:,i);
426         else %obs == 3
427             %GB = scriptG*scriptB was determined symbolically using MATLAB and
428             % Maple; this GB is example specific
429             %Evaluate GB at each time value in tspan
430             GB = [0 0 0;0 0 0;-1 0 0;0 3+cos(1/3*ti) 0];
431             %[V;dV] will be used in constructing the output for the extended
432             % output equation
433             V = 4*cos(2*ti)*sin(ti/5);
434             d1V = -8*sin(2*ti)*sin(ti/5)+(4/5)*cos(2*ti)*cos(ti/5);
435             d2V = -(404/25)*cos(2*ti)*sin(ti/5)-(16/5)*sin(2*ti)*cos(ti/5);
436             dV = [d1V;d2V];
437             %Calculate quhat at the time values in tspan
438             quhat(:,i) = V2*[GB*[V;dV];ye(:,i)] + z(:,i);

```

```

439      %Concatenate the transformed state vector, qr = [qm;qu]
440      qr(:,i) = [GB*[V;dV];ye(:,i);quhat(:,i)];
441      %Evaluate GF at each time value in tspan
442      if onl == 0
443          GF = [-1 1 0 -4-2*sin(ti) 0 0;1 0 0 0 -2-sin(ti) 0;1 0 0 0 ...
444              0 0;-1/3*sin(1/3*ti) 0 0 -1 1 -1];
445      else %onl == 1
446          GF = [-1 1 0 4+2*sin(ti) 0 0;1 0 0 0 2+sin(ti) 0;1 0 0 0 ...
447              0;-1/3*sin(1/3*ti) 0 0 -1 1 -1];
448      end
449      %Construct R at each time value in tspan
450      R = [GF;Cext;Cr];
451      %Transform qr back to xqr (in terms of the original state vector)
452      xqr(:,i) = inv(R)*qr(:,i);
453      %check1 will be used to confirm x=inv(R)*qtilde=0
454      check1(:,i) = inv(R)*qtilde(:,i);
455      %check2 will be used to confirm qm=GBv=0
456      check2(:,i) = [GB*[V;dV]];
457      %check3 will be used to confirm -GFx+GBv=0
458      check3(:,i) = [GF*x(:,i)];
459      end
460  end
461  if obs == 2
462      %Concatenate the transformed state vector, qr = [qm;qu]
463      qr = [y;quhat];
464      %Transform qr back to xqr (in terms of the original state vector)
465      xqr = inv(R)*qr;
466  else end %else obs == 3
467
468 %***** Plot the solutions *****
469 figure
470 plot(tspan,x)
471 hgsave(strcat('xobs',num2str(obs),'comp',num2str(comp)));
472
473 if obs == 1
474 %Plot example system solution and full-order observer on the same graph
475 figure
476 hold on
477 plot(tspan,x), plot(tspan,xhat,'--')
478 hold off
479 hgsave(strcat('xandxhatobs',num2str(obs),'comp',num2str(comp)));
480
481 %Plot the estimation error

```

```

482     figure
483     plot(tspan, (x-xhat))
484     hgsave(strcat('xminusxhatobs',num2str(obs), 'comp', num2str(comp)));;
485 %Display estimation error at final time
486     ee = x(:,ntspan)-xhat(:,ntspan)
487 else %obs == 2 or obs == 3
488 %Confirm x-inv(R)*qtilde=0
489     figure
490     plot(tspan, (x-check1))
491
492 if obs == 2
493 %Confirm y-qm=0
494     figure
495     plot(tspan, (y-qtilde(1:mC,:)))
496 else %obs == 3
497 %Confirm y-qm=0
498     figure
499     plot(tspan, (ye-qtilde(4+1:mmC,:)))
500 %Confirm qm=GBv=0
501     figure
502     plot(tspan, (qtilde(1:4,:)-check2))
503 %Confirm -GFx+GBv=0
504     figure
505     plot(tspan, (-check3+check2))
506 end
507
508 %Plot solution of the transformed system and observer on the same graph
509 figure
510 hold on
511 plot(tspan,qtilde), plot(tspan,qr,'--')
512 hold off
513 hgsave(strcat('qandqrobs',num2str(obs), 'comp', num2str(comp)));
514 %Plot the estimation error for the transformed system
515 figure
516 plot(tspan,qtilde-qr)
517 hgsave(strcat('qminusqrobs',num2str(obs), 'comp', num2str(comp)));
518 %Plot solution of the example system and the observer on the same graph
519 figure
520 hold on
521 plot(tspan,x), plot(tspan,xqr,'--')
522 hold off
523 hgsave(strcat('xandxqrobs',num2str(obs), 'comp', num2str(comp)));
524 %Plot the estimation error
525 figure
526 plot(tspan, (x-xqr))

```

```

527      hgsave(strcat('xminusxqrobs',num2str(obs),'comp',num2str(comp)));
528 %Display estimation error at final time
529      ee = x(:,ntspan)-xqr(:,ntspan)
530 end
531
532 save(strcat('ObsConInfo'))
533 else end %else on6 == 1, Do not construct observer
534 %*****%
535
536 %end      LTVcircuit.m

1 function [AtildeR,dAtildeR,BtildeR] = SLSCcoeff(tnow,nx,C,Cext,Cr,obs,SPML,sv)
2
3 %Function SLSCcoeff.m is called by LTVcircuit.m, WoRank.m,
4 % LTVDAE_Solveplus.m, LTVComp_Solve.m.
5
6 %Function SLSCcoeff.m finds the coefficient matrices
7 % \widetilde{A} (Atilde) and \widetilde{B} (Btilde)
8 %in the (stabilized) least squares completion (S)LSC
9 % \dot{\widetilde{x}}(t) =
10 % \widetilde{A}(t)*\widetilde{x}(t) + \widetilde{B}(t)*v(t)
11 %for either the system associated with the full-order observer, the
12 % reduced-order observer, or the maximally reduced observer.
13
14 %Input tnow -the values of time at which SLSCcoeff.m is being evaluated.
15 %See LTVcircuit.m for description of other inputs.
16
17 %Description of outputs (final letter R is for 'returned' values)
18 %AtildeR, BtildeR - coefficient matrices in the completion.
19 %dAtildeR - the first derivative of AtildeR (even though the first
20 % derivative of AtildeR is not zero for the completion associated with the
21 % full-order observer, its first derivative is not required and will be
22 % returned as zero to eliminate unnecessary calculations).
23
24 %This program is currently set up for the LTV circuit example, Chapter 6.
25 %Derivatives of time-varying matrices were determined outside of this
26 % function to eliminate finding symbolic derivatives within SLSCcoeff.m.
27 %*****%
28
29 %Define time variable t as tnow
30 t = tnow;
31 %Assign switch: positive resistor if on1 = 0, negative resistor if on1 = 1
32 on1 = sv(1);
33 %Assign lambda values 11, 12, 13, 14, 15, 16
34 l1 = SPML(1); l2 = SPML(2); l3 = SPML(3);

```

```

35 14 = SPML(4); 15 = SPML(5); 16 = SPML(6);
36 %Build stabilization parameter matrix \Lambda
37 % Note, if least squares completion, 11=12=13=14=15=16=0
38 lambda = [11 0 0 0 0 0;0 12 0 0 0 0;0 0 13 0 0 0;
39 0 0 0 14 0 0;0 0 0 0 15 0;0 0 0 0 0 16];
40
41 %Construct the coefficient matrices from the derivative array equations
42 %The coefficient matrix of \dot{x} from the LTV example system of DAEs and
43 % its necessary derivatives
44 E = [3+cos(t/3) 0 0 0 0 0;0 2-cos(2*t) 0 0 0 0;0 0 2-exp(-t) 0 0 0;
45 0 0 0 0 0 0;0 0 0 0 0 0;0 0 0 0 0 0];
46 dE = [-(1/3)*sin(t/3) 0 0 0 0 0;0 2*sin(2*t) 0 0 0 0;0 0 exp(-t) 0 0 0;
47 0 0 0 0 0 0;0 0 0 0 0 0;0 0 0 0 0 0];
48 ddE = [-(1/9)*cos(t/3) 0 0 0 0 0;0 4*cos(2*t) 0 0 0 0;0 0 -exp(-t) 0 0 0;
49 0 0 0 0 0 0;0 0 0 0 0 0;0 0 0 0 0 0];
50 %The coefficient matrix of x from the LTV example system of DAEs and
51 % its necessary derivatives
52 if on1 == 0
53 F = [-(1/3)*sin(t/3) 0 0 -1 1 -1;0 2*sin(2*t) 1 1 0 0;0 -1 exp(-t) 0 0 0;
54 -1 1 0 -(4+2*sin(t)) 0 0;1 0 0 0 -(2+sin(t)) 0;1 0 0 0 0 0];
55 dF = [-(1/9)*cos(t/3) 0 0 0 0 0;0 4*cos(2*t) 0 0 0 0;0 0 -exp(-t) 0 0 0;
56 0 0 0 -2*cos(t) 0 0;0 0 0 0 -cos(t) 0;0 0 0 0 0 0];
57 ddF = [(1/27)*sin(t/3) 0 0 0 0 0;0 -8*sin(2*t) 0 0 0 0;0 0 exp(-t) 0 0 0;
58 0 0 0 2*sin(t) 0 0;0 0 0 sin(t) 0;0 0 0 0 0 0];
59 else %on1 == 1
60 F = [-(1/3)*sin(t/3) 0 0 -1 1 -1;0 2*sin(2*t) 1 1 0 0;0 -1 exp(-t) 0 0 0;
61 -1 1 0 (4+2*sin(t)) 0 0;1 0 0 0 (2+sin(t)) 0;1 0 0 0 0 0];
62 dF = [-(1/9)*cos(t/3) 0 0 0 0 0;0 4*cos(2*t) 0 0 0 0;0 0 -exp(-t) 0 0 0;
63 0 0 0 2*cos(t) 0 0;0 0 0 0 cos(t) 0;0 0 0 0 0 0];
64 ddF = [(1/27)*sin(t/3) 0 0 0 0 0;0 -8*sin(2*t) 0 0 0 0;0 0 exp(-t) 0 0 0;
65 0 0 0 -2*sin(t) 0 0;0 0 0 0 -sin(t) 0;0 0 0 0 0 0];
66 end
67 %The coefficient matrix of u from the LTV example system of DAEs
68 B = [0;0;0;0;-1];
69 %Initialize a 6 by 6 zero matrix and a 6 by 1 zero vector
70 z6b6 = zeros(6); z6b1 = zeros(6,1);
71 %Build scriptE, scriptF, and scriptB from the derivative array equations
72 scriptE = [E z6b6 z6b6;dE+F+(lambda*E) E z6b6;
73 ddE+(2*dF)+(2*lambda*dE)+(2*lambda*F)+((lambda^2)*E) ...
74 (2*dE)+F+(2*lambda*E) E];
75 scriptF = [F;dF+(lambda*F);ddF+(2*lambda*dF)+((lambda^2)*F)];
76 scriptB = [B z6b1 z6b1;lambda*B B z6b1;(lambda^2)*B (2*lambda*B) B];
77 %Is the rank of scriptE constant?
78 rscriptE = rank(scriptE);
79

```

```

79 %Find the pseudoinverse of scriptE
80 pE = pinv(scriptE);
81 %(Stabilized) least squares completion
82 %wbar = -PscriptE*scriptF*x + PscriptE*scriptB*v
83 VA = -pE*scriptF; WB = pE*scriptB;
84 %Submatrix of V for (stabilized) least squares completion
85 Atilde = VA(1:nx,:); Btilde = WB(1:nx,:);
86 if obs == 1
87     AtildeR = Atilde; dAtildeR = 0; BtildeR = Btilde;
88 else %obs == 2 or obs == 3
89     %Additional derivatives of E, F from LTV example system of DAEs
90     dddE = [(1/27)*sin(t/3) 0 0 0 0 0;0;-8*sin(2*t) 0 0 0 0;0;0 exp(-t) 0 0 0;
91         0 0 0 0 0 0;0 0 0 0 0 0;0 0 0 0 0 0];
92     if onl == 0
93         dddF = [(1/81)*cos(t/3) 0 0 0 0 0;0;-16*cos(2*t) 0 0 0 0;
94             0 0 -exp(-t) 0 0 0;0 0 0 2*cos(t) 0 0;0;0 0 0 cos(t) 0;0 0 0 0 0 0];
95     else %onl == 1
96         dddF = [(1/81)*cos(t/3) 0 0 0 0 0;0;-16*cos(2*t) 0 0 0 0;
97             0 0 -exp(-t) 0 0 0;0 0 0 -2*cos(t) 0 0;0;0 0 0 -cos(t) 0;0 0 0 0 ...
98                 0 0];
99     end
100    %Build the first derivatives of scriptE and of scriptF
101    dscriptE = [dE z6b6 z6b6;ddE+dF+(lambda*dE) dE ...
102        z6b6;dddE+(2*ddF)+(2*lambda*ddE)+(2*lambda*dF)+((lambda^2)*dE) ...
103            (2*ddE)+dF+(2*lambda*dE) dE];
104    dscriptF = [dF;ddF+(lambda*dF);dddF+(2*lambda*ddF)+((lambda^2)*dF)];
105    if obs == 2
106        %Determine the size of C
107        [mC,nC] = size(C);
108        %Since C is constant, transformation matrix R = [C;Cr] is as well
109        %Let Cr equal the transpose of the basis vectors for N(C)
110        [UC,svdC,VC] = svd(C);
111        %Initialize transformation matrix R; define its first mC as C; define
112        % the remaining rows of R (Cr) as the transpose of the last nx-mC
113        % columns of VC from the svd
114        R = zeros(nC); R(1:mC,:) = C; R(mC+1:nC,:) = VC(:,mC+1:nC)';
115        %Since R is constant, all of its derivatives will equal zero
116        dR = zeros(nx); ddR = zeros(nx);
117    else %obs == 3
118        %Since R is variable, determine GF, dGF, ddGF at each time t
119        if onl == 0
120            %GF = scriptG*scriptF was determined symbolically using MATLAB and
121            % Maple; this GF is example specific
122            GF = [-1 1 0 -4-2*sin(t) 0 0;1 0 0 0 -2-sin(t) 0;
123                  1 0 0 0 0;-1/3*sin(1/3*t) 0 0 -1 1 -1];

```

```

121      %dGF, the first derivative of GF
122      dGF = [0 0 0 -2*cos(t) 0 0;0 0 0 0 -cos(t) 0;
123                  0 0 0 0 0;-1/9*cos(1/3*t) 0 0 0 0];
124      %ddGF, the second derivative of GF
125      ddGF = [0 0 0 2*sin(t) 0 0;0 0 0 0 sin(t) 0;
126                  0 0 0 0 0;1/27*sin(1/3*t) 0 0 0 0];
127      else %on1 == 1
128          GF = [-1 1 0 4+2*sin(t) 0 0;1 0 0 0 2+sin(t) 0;
129                  1 0 0 0 0;-1/3*sin(1/3*t) 0 0 -1 1 -1];
130          dGF = [0 0 0 2*cos(t) 0 0;0 0 0 0 cos(t) 0;
131                  0 0 0 0 0;-1/9*cos(1/3*t) 0 0 0 0];
132          ddGF = [0 0 0 -2*sin(t) 0 0;0 0 0 0 -sin(t) 0;
133                  0 0 0 0 0;1/27*sin(1/3*t) 0 0 0 0];
134      end
135      %Define transformation matrix R
136      R = [GF;Cext;Cr];
137      %Confirm R at t is nonsingular
138      rR = rank(R);
139      if rR == nx
140      else
141          %If the transformation matrix is singular, end program
142          return
143      end
144      %Define dR and ddR, the first and second derivatives of R, respectively
145      % Note, Cext and Cr are assumed constant
146      dR = [dGF;zeros(2,6)]; ddR = [ddGF;zeros(2,6)];
147      end
148      %Transformed coefficient matrices AtildeR and BtildeR
149      AtildeR = R*Atilde*inv(R) + dR*inv(R); BtildeR = R*Btilde;
150      %Define the transpose of the pseudoinverse of scriptE
151      pET = pE';
152      %Define the transpose of the first derivative of scriptE
153      dET = dscriptE';
154      %Calculate the first derivative of the pseudoinverse of scriptE
155      d1pE = -pE*dscriptE*pE + dET*pET*pE - pE*scriptE*dET*pET*pE + ...
156          pE*pET*dET - pE*pET*dET*scriptE*pE;
157      %Determine the first derivative of VA from above
158      d1VA = -(d1pE*scriptF + pE*dscriptF);
159      %Calculate the first derivative of \widetilde{A}
160      d1A = d1VA(1:nx,1:nx);
161      %Transformed first derivative of coefficient matrix
162      dAtildeR = dR*Atilde*inv(R) + R*d1A*inv(R) - R*Atilde*inv(R)*dR*inv(R) ...
163          + ddR*inv(R) - dR*inv(R)*dR*inv(R);
164  end
165  %end    SLSCcoeff.m

```

```

1 function [AtildeR,dAtildeR,BtildeR]=ASCcoeff(tnow,nx,C,Cext,Cr,obs,SPML,sv)
2
3 %Function ASCcoeff.m is called by LTVcircuit.m, WoRank.m,
4 % LTVDAE_Solveplus.m, LTVComp_Solve.m.
5
6 %Function ASCcoeff.m finds the coefficient matrices
7 % \widetilde{A} (Atilde) and \widetilde{B} (Btilde)
8 %in the alternative stabilized completion ASC
9 % \dot{\widetilde{x}}(t) =
10 % \widetilde{A}(t)*\widetilde{x}(t) + \widetilde{B}(t)*v(t)
11 %for either the system associated with the full-order observer, the
12 % reduced-order observer, or the maximally reduced observer.
13
14 %*** IMPORTANT ***
15 %The completion, or full-order system, can be found using either the smooth
16 %decomposition method (SDM) or the symbolically defined method (SYM). If
17 %SDM is chosen, this program requires command lyap from the Control System
18 %Toolbox. The smooth decomposition method has not been developed for either
19 %the reduced-order or maximally reduced observer.
20 %*****
21
22 %Input tnow -the values of time at which ASCcoeff.m is being evaluated.
23 % Also, the current and the "initial" time used in calculating the star
24 % variables for the smooth decomposition method.
25 %See LTVcircuit.m for description of other inputs.
26
27 %Description of outputs (final letter R is for 'returned' values)
28 %AtildeR, BtildeR - coefficient matrices in the completion.
29 %dAtildeR - the first derivative of AtildeR (even though the first
30 % derivative of AtildeR is not zero for the completion associated with the
31 % full-order observer, its first derivative is not required and will be
32 % returned as zero to eliminate unnecessary calculations).
33
34 %This program is currently set up for the LTV circuit example, Chapter 6.
35 %Derivatives of time-varying matrices were determined outside of this
36 % function to eliminate finding symbolic derivatives within ASCcoeff.m.
37
38 %Tab alignment eliminated for formatting in LaTeX.
39 %*****
40
41 %Define time variable t as tnow
42 t = tnow;
43 %Assign switches:
44 %Positive resistor if on1 = 0, negative resistor if on1 = 1
45 on1 = sv(1);

```

```

46 %SYM Z2T if on2 = 0, SDM Z2T if on2 = 1
47 on2 = sv(2);
48 %SYM T2 and Z1T if on3 = 0, SDM T2 and Z1T if on3 = 1
49 on3 = sv(3);
50 %Assign lambda values 11, 12, 13, 14
51 l1 = SPML(1); l2 = SPML(2); l3 = SPML(3); l4 = SPML(4);
52 %Define capacitors and inductor and their derivatives
53 C1 = 3 + cos(t/3); C2 = 2 - cos(2*t); L = 2 - exp(-t);
54 dC1 = -1/3*sin(t/3); dC2 = 2*sin(2*t); dL = exp(-t);
55 %Define input (control, voltage source) and its 1st and 2nd derivatives
56 V = 4*cos(2*t)*sin(t/5); d1V = -8*sin(2*t)*sin(t/5)+4/5*cos(2*t)*cos(t/5);
57 d2V = -404/25 *cos(2*t)*sin(t/5)-16/5*sin(2*t)*cos(t/5);
58 %Other input and derivative vectors
59 dV = [d1V;d2V]; VdV = [V;dV];
60
61 if on2 == 0 %SYM Z2T
62 %These completion coefficient matrices were determined symbolically using
63 % MATLAB and Maple
64 %1st, 2nd, 3rd rows of Atilde, Btilde are same for Pos and Neg resistors
65 Arow1 = [-l3 0 0 0 0 0]; Arow2 = [0 -2*sin(2*t)/C2 -1/C2 -1/C2 0 0];
66 Arow3 = [0 1/L -exp(-t)/L 0 0 0];
67 Brow1 = -d1V - l3*V; Brow2 = 0; Brow3 = 0;
68 %4th row, 5th col; 4th row, 6th col; 6th row, 6th col of Atilde are same
69 % for Pos and Neg Resistors
70 r4c5 = 0; r4c6 = 0; r6c6 = -14;
71 %
72 if on1 == 0 %Atilde and Btilde SYM Z2T, Positive Resistors
73 %Define positive resistors
74 pR1 = 4 + 2*sin(t); pR2 = 2 + sin(t);
75 %4th row and 1st col, 2nd col, 3rd col, 4th col
76 r4c1 = -1/2*l1/pR2 + 1/2*l3/pR2; r4c2 = -sin(2*t)/(C2*pR2) + 1/2*l1/pR2;
77 r4c3 = -1/2/(C2*pR2); r4c4 = -1/2/(C2*pR2) + 1/2*(-2*cos(t)-l1*pR1)/pR2;
78 %4th row, 5th row
79 Arow4 = [r4c1 r4c2 r4c3 r4c4 r4c5 r4c6];
80 Arow5 = [l2/pR2-l3/pR2 0 0 0 (-cos(t)-l2*pR2)/pR2 0];
81 %6th row and 1st col, 2nd col, 3rd col, 4th col, 5th col
82 r6c1 = 1/2*l1/pR2+l2/pR2+1/6*(2*sin(t/3)*sin(t)-9+4*sin(t/3))*l3/pR2 ...
    -1/9*cos(t/3)-1/3*l4*sin(t/3);
83 r6c2 = sin(2*t)/(C2*pR2) - 1/2*l1/pR2; r6c3 = 1/2/(C2*pR2);
84 r6c4 = 1/2/(C2*pR2) - 1/2*(-2*cos(t)-l1*pR1)/pR2 - 14;
85 r6c5 = (-cos(t)-l2*pR2)/pR2 + 14;
86 %6th row
87 Arow6 = [r6c1 r6c2 r6c3 r6c4 r6c5 r6c6];
88 %Build Atilde
89 Atilde = [Arow1;Arow2;Arow3;Arow4;Arow5;Arow6];

```

```

90 %4th row, 5th row, 6th row
91 Brow4 = -1/2*Brow1/pR2; Brow5 = Brow1/pR2;
92 Brow6 = -1/6*(2*sin(t/3)*sin(t)-9+4*sin(t/3))*Brow1/pR2+1/3*sin(t/3)*d1V ...
    -C1*d2V-14*C1*d1V;
93 %Build Btilde
94 Btilde = [Brow1;Brow2;Brow3;Brow4;Brow5;Brow6];
95 else %on1 == 1, Atilde and Btilde SYM Z2T, Negative Resistors
96 %Define negative resistors
97 nR1 = -4 - 2*sin(t); nR2 = -2 - sin(t);
98 r4c1 = -1/2*11/nR2 + 1/2*13/nR2; r4c2 = -sin(2*t)/(C2*nR2) + 1/2*11/nR2;
99 r4c3 = -1/2/(C2*nR2); r4c4 = -1/2/(C2*nR2) + 1/2*(2*cos(t)-11*nR1)/nR2;
100 Arow4 = [r4c1 r4c2 r4c3 r4c4 r4c5 r4c6];
101 Arow5 = [12/nR2-13/nR2 0 0 0 (cos(t)-12*nR2)/nR2 0];
102 r6c1 = 1/2*11/nR2+12/nR2-1/6*(2*sin(t/3)*sin(t)+9+4*sin(t/3))*13/nR2 ...
    -1/9*cos(t/3)-1/3*14*sin(t/3);
103 r6c2 = sin(2*t)/(C2*nR2) - 1/2*11/nR2; r6c3 = 1/2/(C2*nR2);
104 r6c4 = 1/2/(C2*nR2) - 1/2*(2*cos(t)-11*nR1)/nR2 - 14;
105 r6c5 = (cos(t)-12*nR2)/nR2 + 14;
106 Arow6 = [r6c1 r6c2 r6c3 r6c4 r6c5 r6c6];
107 %Build Atilde
108 Atilde = [Arow1;Arow2;Arow3;Arow4;Arow5;Arow6];
109 Brow4 = -1/2*Brow1/nR2; Brow5 = Brow1/nR2;
110 Brow6 = 1/6*(2*sin(t/3)*sin(t)+9+4*sin(t/3))*Brow1/nR2+1/3*sin(t/3)*d1V ...
    -C1*d2V-14*C1*d1V;
111 %Build Btilde
112 Btilde = [Brow1;Brow2;Brow3;Brow4;Brow5;Brow6];
113 end %end Atilde and Btilde SYM Z2T if/else
114 %
115 else %on2 == 1, Derivative Array Equations, SDM Z2T
116 %Construct the coefficient matrices from the derivative array equations
117 %The coefficient matrix of \dot{x} from the LTV example system of DAEs and
118 % its necessary derivatives
119 E = [3+cos(t/3) 0 0 0 0 0;0 2-cos(2*t) 0 0 0 0;0 0 2-exp(-t) 0 0 0;
120     0 0 0 0 0;0 0 0 0 0;0 0 0 0 0];
121 dE = [-(1/3)*sin(t/3) 0 0 0 0 0;0 2*sin(2*t) 0 0 0 0;0 0 exp(-t) 0 0 0;
122     0 0 0 0 0;0 0 0 0 0;0 0 0 0 0];
123 ddE = [-(1/9)*cos(t/3) 0 0 0 0 0;0 4*cos(2*t) 0 0 0 0;
124     0 0 -exp(-t) 0 0 0;0 0 0 0 0;0 0 0 0 0];
125 dddE = [(1/27)*sin(t/3) 0 0 0 0 0;0 -8*sin(2*t) 0 0 0 0;
126     0 0 exp(-t) 0 0 0;0 0 0 0 0;0 0 0 0 0];
127 %
128 %The coefficient matrix of x from the LTV example system of DAEs and
129 % its necessary derivatives
130 if on1 == 0 %F and derivatives, Positive Resistors
131 F = [-(1/3)*sin(t/3) 0 0 -1 1 -1;0 2*sin(2*t) 1 1 0 0;0 -1 exp(-t) 0 0 0;

```

```

132      -1 1 0 -(4+2*sin(t)) 0 0;1 0 0 -(2+sin(t)) 0;1 0 0 0 0 0];
133 dF = [-(1/9)*cos(t/3) 0 0 0 0 0;0 4*cos(2*t) 0 0 0 0;0 0 -exp(-t) 0 0 0;
134      0 0 0 -2*cos(t) 0 0;0 0 0 0 -cos(t) 0;0 0 0 0 0];
135 ddF = [(1/27)*sin(t/3) 0 0 0 0 0;0 -8*sin(2*t) 0 0 0 0;0 0 exp(-t) 0 0 0;
136      0 0 0 2*sin(t) 0 0;0 0 0 0 sin(t) 0;0 0 0 0 0];
137 dddF = [(1/81)*cos(t/3) 0 0 0 0 0;0 -16*cos(2*t) 0 0 0 0;
138      0 0 -exp(-t) 0 0 0;0 0 0 2*cos(t) 0 0;0 0 0 0 cos(t) 0;0 0 0 0 0];
139 else %on1 == 1, F and derivatives, Negative Resistors
140 F = [-(1/3)*sin(t/3) 0 0 -1 1 -1;0 2*sin(2*t) 1 1 0 0;0 -1 exp(-t) 0 0 0;
141      -1 1 0 (4+2*sin(t)) 0 0;1 0 0 0 (2+sin(t)) 0;1 0 0 0 0];
142 dF = [-(1/9)*cos(t/3) 0 0 0 0 0;0 4*cos(2*t) 0 0 0 0;0 0 -exp(-t) 0 0 0;
143      0 0 0 2*cos(t) 0 0;0 0 0 0 cos(t) 0;0 0 0 0 0];
144 ddF = [(1/27)*sin(t/3) 0 0 0 0 0;0 -8*sin(2*t) 0 0 0 0;0 0 exp(-t) 0 0 0;
145      0 0 0 -2*sin(t) 0 0;0 0 0 0 -sin(t) 0;0 0 0 0 0];
146 dddF = [(1/81)*cos(t/3) 0 0 0 0 0;0 -16*cos(2*t) 0 0 0 0;
147      0 0 -exp(-t) 0 0 0;0 0 0 -2*cos(t) 0 0;0 0 0 0 -cos(t) 0;0 0 0 0 0];
148 end %end F and derivatives
149 %
150 %The coefficient matrix of u from the LTV example system of DAEs
151 B = [0;0;0;0;0;-1];
152 %Initialize a 6 by 6 zero matrix and a 6 by 1 zero vector
153 z6b6 = zeros(6); z6b1 = zeros(6,1);
154 %Build scriptE, scriptF, and scriptB, no stabilized differentiation
155 scriptE = [E z6b6 z6b6;dE+F E z6b6;ddE+(2*dF) (2*dE)+F E];
156 scriptF = [F;dF;ddF]; scriptB = [B z6b1 z6b1;z6b1 B z6b1;z6b1 z6b1 B];
157 %Build the first derivatives of scriptE and of scriptF
158 dscriptE = [dE z6b6 z6b6;ddE+dF dE z6b6;dddE+(2*ddF) (2*ddE)+dF dE];
159 dscriptF = [dF;ddF;dddf]; dscriptB = zeros(3*nx,3);
160 %Is the rank of scriptE constant?
161 rscriptE = rank(scriptE);
162 end %end Atilde, Btilde SYM; Derivative Array Equations SDM
163
164 if obs == 1 %Full-order observer AtildeR, dAtildeR, BtildeR
165 %
166 if on2 == 0 %AtildeR, dAtildeR, BtildeR SYM
167 AtildeR = Atilde; dAtildeR = 0; BtildeR = Btilde;
168 else %on2 == 1, AtildeR, dAtildeR, BtildeR SDM
169 %Determine the size of scriptE
170 [mscriptE,nscriptE] = size(scriptE);
171 %Find the singular value decomposition of scriptE and define
172 % Util = [Util1 Util2],
173 % Util1 = an orthonormal basis for R(scriptE)
174 % Util2 = an orthonormal basis for N(scriptE^T)
175 [Q1Z2T,S10Z2T,PiQ2Z2T] = svd(scriptE);
176 Util2 = Q1Z2T(:,(rscriptE+1):mscriptE);

```

```

177 %Find the singular value decomposition of product (Util2^T)*Util2
178 %Then use program Rounding.m to make sure any singular values that
179 % should be zero are zero
180 [hatU2,hatS20,hatV2] = svd((Util2')*Util2); hatS2 = Rounding(hatS20);
181 %Define matrix U2 - the desired smooth matrix; Z2T = U2'
182 U2 = Util2*hatU2*(hatV2'); Z2T = U2';
183 %%
184 %Construct the projection PU used in calculating the derivative of U2
185 PU = scriptE*pinv(scriptE);
186 %Calculate the first derivative of PU
187 dPU = ((eye(mscriptE)-PU)*dscriptE*pinv(scriptE)) + ...
    (((eye(mscriptE)-PU)*dscriptE*pinv(scriptE))');
188 %Calculate the first derivative of U2
189 %The generalized Lyapunov equation
190 % L*X*E^T + E*X*L^T + Q = 0
191 %solves for X using X = lyap(L,Q,[],E).
192 %The equation we're solving does not have a matrix L - use identity
193 %eye(mscriptE0-rsE)*GU2*hats2 + hats2*GU2*eye(mscriptE0-rsE)
194 % +((hatV2')*(Ustar2')*dPU*Ustar2*hatV2) = 0
195 GammaU20 = (hatV2')*(Util2')*dPU*Util2*hatV2; GammaU2 = Rounding(GammaU20);
196 GU2 = lyap(eye((mscriptE-rsE)),GammaU2,[],hats2);
197 HU2 = hatV2*hats2*(hatV2'); dHU2 = hatV2*GU2*(hatV2');
198 dU2 = (((-dPU)*Util2) - (U2*dHU2))*(inv(HU2)); dZ2T = dU2';
199 %%
200 %Calculate Z2^T*scriptF for use in calculating T2
201 Z2TsF = Z2T*scriptF;
202 %Find the rank of Z2TsF
203 rZ2TsF = rank(Z2TsF);
204 %Determine the size of Z2TsF
205 [mZ2TsF,nZ2TsF] = size(Z2TsF);
206 %
207 if on3 == 0 %Find T2 SYM
208 %Find the singular value decomposition of Z2TsF and define
209 % Vtil = [Vstar1 Vstar2],
210 % Vtil1 = an orthonormal basis for R(Z2TsF^T)
211 % Vtil2 = an orthonormal basis for N(Z2TsF)
212 [Q1T2,S10T2,PiQ2T2] = svd(Z2TsF);
213 %Define matrix T2
214 T2 = PiQ2T2(:,(rZ2TsF+1):nZ2TsF);
215 else %on3 == 1, Find T2 SDM
216 %Find T2 using a smooth decomposition
217 [Q1T2,S10T2,PiQ2T2] = svd(Z2TsF); Vtil2 = PiQ2T2(:,(rZ2TsF+1):nZ2TsF);
218 %Find the singular value decomposition of product (Vtil2^T)*Vtil2
219 %Then use program Rounding.m to make sure any singular values that
220 % should be zero are zero

```

```

221 [barU2,barS20,barV2] = svd((Vtil2')*Vtil2); barS2 = Rounding(barS20);
222 %Define matrix V2 - the desired smooth matrix; T2 = V2
223 T2 = Vtil2*barU2*(barV2');
224 end %end Find T2
225 %
226 %Calculate E*T2 for use in calculating Z_{1,0}^T = Z1T
227 ET2 = E*T2;
228 %Find the rank of ET2
229 rET2 = rank(ET2);
230 %
231 if on3 == 0 %Find Z1T SYM
232 %Find the singular value decomposition of ET2 and define
233 % Util = [Util1 Util2],
234 % Util1 = an orthonormal basis for R(ET2)
235 % Util2 = an orthonormal basis for N(ET2^T)
236 [Q1Z1T,S10Z1T,PiQ2Z1T] = svd(ET2);
237 %Define matrix Z1T
238 Z1T = Q1Z1T(:,1:rET2)';
239 else %on3 == 1, Find Z1T SDM
240 %Find Z1T using a smooth decomposition
241 [Q1Z1T,S10Z1T,PiQ2Z1T] = svd(ET2); Util1 = Q1Z1T(:,1:rET2);
242 %Find the singular value decomposition of product (Util1^T)*Util1
243 %Then use program Rounding.m to make sure any singular values that
244 % should be zero are zero
245 [hatU1,hatS10,hatV1] = svd((Util1')*Util1); hatS1 = Rounding(hatS10);
246 %Define matrix U1 - the desired smooth matrix; Z1T = U1^T
247 U1 = Util1*hatU1*(hatV1'); Z1T = U1';
248 end %end Find Z1T
249 %
250 %Determine the coefficient matrices of the ASC
251 %Atilde = -inv([Z1T*E;Z2T*scriptF])*[Z1T*F;dZ2T*scriptF + Z2T*dscriptF
252 % + lambda*Z2T*scriptF]
253 %Btilde = inv([Z1T*E;Z2T*scriptF])*[Z1T*B*V;dZ2T*scriptB*[V;dV]
254 % + Z2T*dscriptB*[V;dV] + Z2T*scriptB*[Vp;dVp]
255 % + lambda*Z2T*scriptB*[V;dV]]
256 %Matrices and other quantities needed to calculate Atilde and Btilde
257 %Determine the size of V
258 [mV,nV] = size(V);
259 %Vp = first derivative of V
260 Vp = dV(1);
261 %dVp = [second derivative of V;third derivative of V]
262 dVp = [dV(2);824/25*sin(2*t)*sin(1/5*t)-1204/125*cos(2*t)*cos(1/5*t)];
263 %Define lambda, an mZ2TsF by mZ2TsF matrix
264 % (note, the dimensions are not nx by nx)
265 lambda1 = [11 0 0 0;0 12 0 0;0 0 13 0;0 0 0 14];

```

```

266 % [Z1T*E;Z2T*scriptF] is the coefficient of \dot{x}; calculate its
267 % inverse in order to be able to solve for \dot{x}
268 cAB = inv([Z1T*E;Z2T*scriptF]);
269 %Calculate AtildeR and BtildeR
270 AtildeR = -cAB*[Z1T*F; (dZ2T*scriptF)+(Z2T*dscriptF)+(lambda1*Z2T*scriptF)];
271 dAtildeR = 0;
272 BtildeR = cAB*[Z1T*B*V; (dZ2T*scriptB*VdV)+(Z2T*dscriptB*VdV) ...
+ (Z2T*scriptB*[Vp;dVp])+(lambda1*Z2T*scriptB*VdV)];
273 end %end AtildeR, dAtildeR, BtildeR SYM, SDM Full-order observer
274 %
275 else %obs == 2 or obs == 3, AtildeR, dAtildeR, and BtildeR
276 %
277 if obs == 2 %Transformation matrix R, Reduced-order observer
278 %Determine the size of C
279 [mC,nC] = size(C);
280 %Since C is constant, transformation matrix R = [C;Cr] is as well
281 %Let Cr equal the transpose of the basis vectors for N(C)
282 [UC,svdC,VC] = svd(C);
283 %Initialize transformation matrix R; define its first mC as C; define the
284 % remaining rows of R (Cr) as the transpose of the last nx-mC columns of
285 % VC from the svd
286 R = zeros(nC); R(1:mC,:) = C; R(mC+1:nC,:) = VC(:,mC+1:nC)';
287 %Since R is constant, all of its derivatives will equal zero
288 dR = zeros(nx); ddR = zeros(nx);
289 else %obs == 3, Transformation matrix R, Maximally reduced observer
290 %
291 %Since R is variable, determine GF, dGF, ddGF at each time t
292 if on1 == 0
293 %GF = scriptG*scriptF was determined symbolically using MATLAB and Maple;
294 % this GF is example specific
295 GF = [-1 1 0 -4-2*sin(t) 0 0;1 0 0 0 -2-sin(t) 0;1 0 0 0 0 0;
296 -1/3*sin(1/3*t) 0 0 -1 1 -1];
297 %dGF, the first derivative of GF
298 dGF = [0 0 0 -2*cos(t) 0 0;0 0 0 0 -cos(t) 0;0 0 0 0 0 0;
299 -1/9*cos(1/3*t) 0 0 0 0 0];
300 %ddGF, the second derivative of GF
301 ddGF = [0 0 0 2*sin(t) 0 0;0 0 0 0 sin(t) 0;0 0 0 0 0 0;
302 1/27*sin(1/3*t) 0 0 0 0 0];
303 else %on1 == 1
304 GF = [-1 1 0 4+2*sin(t) 0 0;1 0 0 0 2+sin(t) 0;1 0 0 0 0 0;
305 -1/3*sin(1/3*t) 0 0 -1 1 -1];
306 dGF = [0 0 0 2*cos(t) 0 0;0 0 0 0 cos(t) 0;0 0 0 0 0 0;
307 -1/9*cos(1/3*t) 0 0 0 0 0];
308 ddGF = [0 0 0 -2*sin(t) 0 0;0 0 0 0 -sin(t) 0;0 0 0 0 0 0;
309 1/27*sin(1/3*t) 0 0 0 0 0];

```

```

310 end %end Determining GF, dGF, ddGF
311 %
312 %Define transformation matrix R
313 R = [GF;Cext;Cr];
314 %
315 %Confirm R at t is nonsingular
316 rR = rank(R);
317 if rR == nx
318 else
319 %If the transformation matrix is singular, end program
320 return
321 end %end R confirmation
322 %
323 %Define dR and ddR, the first and second derivatives of R, respectively
324 % Note, Cext and Cr are assumed constant
325 dR = [dGF;zeros(2,6)]; ddR = [ddGF;zeros(2,6)];
326 end %end Transformation R
327 %
328 %Transformed coefficient matrices AtildeR and BtildeR
329 AtildeR = R*Atilde*inv(R) + dR*inv(R); BtildeR = R*Btilde;
330 %The first derivative of Atilde determined symbolically using Maple
331 %1st, 2nd, 3rd rows of dAtilde are same for Pos and Neg resistors
332 dArow1 = [0 0 0 0 0 0];
333 dArow2 = [0 (dC2)^2/(-C2)^2-4*cos(2*t)/C2 dC2/(-C2)^2 dC2/(-C2)^2 0 0];
334 dArow3 = [0 -dL/(-L)^2 (dL)^2/(-L)^2+dL/L 0 0 0];
335 %4th row, 5th col; 4th row, 6th col; 6th row, 6th col of dAtilde are same
336 % for Pos and Neg Resistors
337 dr4c5 = 0; dr4c6 = 0; dr6c6 = 0;
338 %
339 if on1 == 0 %dAtilde Positive Resistors
340 %Define 1st derivative of positive resistors
341 dpR1 = 2*cos(t); dpR2 = cos(t);
342 %4th row and 1st col, 2nd col, 3rd col, 4th col
343 dr4c1 = 1/2*11*dpR2/(pR2)^2 - 1/2*13*dpR2/(pR2)^2;
344 dr4c2 = 2*sin(2*t)^2/((-C2)^2*pR2)+sin(2*t)*dpR2/(C2*(pR2)^2) ...
-2*cos(2*t)/(C2*pR2)-1/2*11*dpR2/(pR2)^2;
345 dr4c3 = sin(2*t)/((-C2)^2*pR2) + 1/2*dpR2/(C2*(pR2)^2);
346 dr4c4 = sin(2*t)/((-C2)^2*pR2)+1/2*dpR2/(C2*(pR2)^2) ...
+1/2*(dpR1+11*pR1)*dpR2/(pR2)^2+1/2*(2*sin(t)-11*dpR1)/pR2;
347 %4th row, 5th row
348 dArow4 = [dr4c1 dr4c2 dr4c3 dr4c4 dr4c5 dr4c6];
349 dArow5 = [-12*dpR2/(pR2)^2+13*dpR2/(pR2)^2 0 0 0 ... 
(dpR2+12*pR2)*dpR2/(pR2)^2+(sin(t)-12*dpR2)/pR2 0];
350 %6th row and 1st col, 2nd col, 3rd col, 4th col, 5th col
351 dr6c1 = -1/2*11*dpR2/(pR2)^2-12*dpR2/(pR2)^2 ...

```

```

+1/6*(2/3*cos(t/3)*sin(t)+sin(t/3)*dpR1+4/3*cos(t/3))*l3/pR2 ...
-1/6*(2*sin(t/3)*sin(t)-9+2*sin(t/3))*l3*dpR1/(pR2)^2 ...
+1/27*sin(t/3)-1/9*l4*cos(t/3);
352 dr6c2 = -2*sin(2*t)^2/((-C2)^2*pR2)-sin(2*t)*dpR2/(C2*(pR2)^2) ...
+2*cos(2*t)/(C2*pR2)+1/2*l1*dpR2/(pR2)^2;
353 dr6c3 = -sin(2*t)/((-C2)^2*pR2) - 1/2*dpR2/(C2*(pR2)^2);
354 dr6c4 = -sin(2*t)/((-C2)^2*pR2)-1/2*dpR2/(C2*(pR2)^2) ...
-1/2*(dpR1+l1*pR1)*dpR2/(pR2)^2-1/2*(2*sin(t)-l1*dpR1)/pR2;
355 dr6c5 = (dpR2+l2*pR2)*dpR2/(pR2)^2 + (sin(t)-l2*dpR2)/pR2;
356 %6th row
357 dArow6 = [dr6c1 dr6c2 dr6c3 dr6c4 dr6c5 dr6c6];
358 %Build dAtilde
359 dAtilde = [dArow1;dArow2;dArow3;dArow4;dArow5;dArow6];
360 else %on1 == 1, dAtilde Negative Resistors
361 %Define 1st derivative of negative resistors
362 dnR1 = -2*cos(t); dnR2 = -cos(t);
363 dr4c1 = 1/2*l1*dnR2/(-nR2)^2 - 1/2*l3*dnR2/(-nR2)^2;
364 dr4c2 = 2*sin(2*t)^2/((-C2)^2*nR2)+sin(2*t)*dnR2/(C2*(-nR2)^2) ...
-2*cos(2*t)/(C2*nR2)-1/2*l1*dnR2/(-nR2)^2;
365 dr4c3 = sin(2*t)/((-C2)^2*nR2) + 1/2*dnR2/(C2*(-nR2)^2);
366 dr4c4 = sin(2*t)/((-C2)^2*nR2)+1/2*dnR2/(C2*(-nR2)^2) ...
+1/2*(dnR1+l1*nR1)*dnR2/(-nR2)^2-1/2*(2*sin(t)+l1*dnR1)/dnR2;
367 dArow4 = [dr4c1 dr4c2 dr4c3 dr4c4 dr4c5 dr4c6];
368 dArow5 = [-12*dnR2/(-nR2)^2+l3*dnR2/(-nR2)^2 0 0 0 ...
(dnR2+l2*nR2)*dnR2/(-nR2)^2-(sin(t)+l2*dnR2)/nR2 0];
369 dr6c1 = -1/2*l1*dnR2/(-nR2)^2-12*dnR2/(-nR2)^2 ...
-1/6*(2/3*cos(t/3)*sin(t)-sin(t/3)*dnR1+4/3*cos(t/3))*l3/nR2 ...
-1/6*(2*sin(t/3)*sin(t)+9-2*sin(t/3))*l3*dnR1/(-nR2)^2 ...
+1/27*sin(t/3)-1/9*l4*cos(t/3);
370 dr6c2 = -2*sin(2*t)^2/((-C2)^2*nR2)-sin(2*t)*dnR2/(C2*(-nR2)^2) ...
+2*cos(2*t)/(C2*nR2)+1/2*l1*dnR2/(-nR2)^2;
371 dr6c3 = -sin(2*t)/((-C2)^2*nR2) - 1/2*dnR2/(C2*(-nR2)^2);
372 dr6c4 = -sin(2*t)/((-C2)^2*nR2)-1/2*dnR2/(C2*(-nR2)^2) ...
-1/2*(dnR1+l1*nR1)*dnR2/(-nR2)^2+1/2*(2*sin(t)+l1*dnR1)/nR2;
373 dr6c5 = (dnR2+l2*nR2)*dnR2/(-nR2)^2 - (sin(t)+l2*dnR2)/nR2;
374 dArow6 = [dr6c1 dr6c2 dr6c3 dr6c4 dr6c5 dr6c6];
375 %Build dAtilde
376 dAtilde = [dArow1;dArow2;dArow3;dArow4;dArow5;dArow6];
377 end %end dAtilde Reduced-order observer, Maximally reduced-observer
378 %
379 %Transformed first derivative of coefficient matrix
380 dAtildeR = dR*dAtilde*inv(R) + R*dAtilde*inv(R) - R*dAtilde*inv(R)*dR*inv(R) ...
+ ddR*inv(R) - dR*inv(R)*dR*inv(R);
381 end %end AtildeR, dAtildeR, BtildeR all observers
382 %end ASCcoeff.m

```

```

1 function dWoR2 = WoRank(tspan,Wo0,nx,mmodC,C,Cext,Cr,obs,comp,SPML,sv,onU,rDWo)
2
3 %Function WoRank.m is called by LTVcircuit.m.
4
5 %The purpose of WoRank.m is to solve two differential equations in order to
6 % calculate the rank of the (observability) Gramian on the time interval
7 % being considered (observability check).
8
9 %*** IMPORTANT ***
10 %If this program is called to determine the standard form for the
11 %unobservable system, then command lyap from the Control System Toolbox
12 %is required.
13 %*****
14
15 %Input onU is on/off switch for checking unobservable system.
16 %If onU = 0, rDWo = 2; otherwise, rDWo is the Gramian's rank at tf=45.
17 %See LTVcircuit.m for description of other inputs.
18 %*****
19
20 %Determine the size of C
21 [mC,nC] = size(C);
22
23 %Determine A,K in pair (A,K) for observability check
24 if obs == 1 & comp == 1
25 %Determine the number of unmeasurable/unknown components
26 nu = nx;
27 %Wo0 is passed through as a vector; reshape into an initial condition
28 % matrix for Wo (the Gramian), for Phi (state transition matrix), and for
29 % x3 (unobservable system's state transition matrix) if onU = 1
30 Wo = reshape(Wo0(1:(nu^2)),nu,nu);
31 Phi = reshape(Wo0((nu^2)+1:2*(nu^2)),nu,nu);
32 if onU == 1
33 x3 = reshape(Wo0(2*(nu^2)+1:2*(nu^2)+(nu-rDWo)^2),(nu-rDWo),(nu-rDWo));
34 else
35 end
36 %Calculate the coefficient matrices in the completion
37 [A,dA,B] = SLSCcoeff(tspan,nx,C,Cext,Cr,obs,SPML,sv);
38 %Assign the output matrix to a general variable
39 K = C;
40 elseif obs == 2 & comp == 1
41 nu = nx - mC;
42 Wo = reshape(Wo0(1:(nu^2)),nu,nu);
43 Phi = reshape(Wo0((nu^2)+1:2*(nu^2)),nu,nu);
44 if onU == 1 %System has unobservable block to be checked
45 x3 = reshape(Wo0(2*(nu^2)+1:2*(nu^2)+(nu-rDWo)^2),(nu-rDWo),(nu-rDWo));

```

```

46    else end %else onU == 0, only observability check of whole system
47    [qA,qdA,qB] = SLSCcoeff(tspan,nx,C,Cext,Cr,obs,SPML,sv);
48    A = qA(mC+1:nx,mC+1:nx);
49    K = qA(1:mC,mC+1:nx);
50 elseif obs == 3 & comp == 1
51    nu = nx - mmodC;
52    Wo = reshape(Wo0(1:(nu^2)),nu,nu);
53    Phi = reshape(Wo0((nu^2)+1:2*(nu^2)),nu,nu);
54 if onU == 1
55    x3 = reshape(Wo0(2*(nu^2)+1:2*(nu^2)+(nu-rDWo)^2),(nu-rDWo),(nu-rDWo));
56 else end %else onU == 0
57 [qA,qdA,qB] = SLSCcoeff(tspan,nx,C,Cext,Cr,obs,SPML,sv);
58 A = qA(mmodC+1:nx,mmodC+1:nx);
59 K = qA(1:mmodC,mmodC+1:nx);
60 elseif obs == 1 & comp == 2
61    nu = nx;
62    Wo = reshape(Wo0(1:(nu^2)),nu,nu);
63    Phi = reshape(Wo0((nu^2)+1:2*(nu^2)),nu,nu);
64 if onU == 1
65    x3 = reshape(Wo0(2*(nu^2)+1:2*(nu^2)+(nu-rDWo)^2),(nu-rDWo),(nu-rDWo));
66 else end %else onU == 0
67 [A,dA,B] = ASCcoeff(tspan,nx,C,Cext,Cr,obs,SPML,sv);
68 K = C;
69 elseif obs == 2 & comp == 2
70    nu = nx - mC;
71    Wo = reshape(Wo0(1:(nu^2)),nu,nu);
72    Phi = reshape(Wo0((nu^2)+1:2*(nu^2)),nu,nu);
73 if onU == 1
74    x3 = reshape(Wo0(2*(nu^2)+1:2*(nu^2)+(nu-rDWo)^2),(nu-rDWo),(nu-rDWo));
75 else end %else onU == 0
76 [qA,qdA,qB] = ASCcoeff(tspan,nx,C,Cext,Cr,obs,SPML,sv);
77 A = qA(mC+1:nx,mC+1:nx);
78 K = qA(1:mC,mC+1:nx);
79 else %obs == 3 & comp == 2
80    nu = nx - mmodC;
81    Wo = reshape(Wo0(1:(nu^2)),nu,nu);
82    Phi = reshape(Wo0((nu^2)+1:2*(nu^2)),nu,nu);
83 if onU == 1
84    x3 = reshape(Wo0(2*(nu^2)+1:2*(nu^2)+(nu-rDWo)^2),(nu-rDWo),(nu-rDWo));
85 else end %else onU == 0
86 [qA,qdA,qB] = ASCcoeff(tspan,nx,C,Cext,Cr,obs,SPML,sv);
87 A = qA(mmodC+1:nx,mmodC+1:nx);
88 K = qA(1:mmodC,mmodC+1:nx);
89 end
90
```

```

91 %The differential equation for the Gramian
92 dWo = (Phi')*(K')*K*Phi;
93 %The differential equation for the state transition matrix
94 dPhi = A*Phi;
95
96 %The differential equation for the unobservable system's state transition
97 % matrix
98 if onU == 1
99     %Determine the size of Wo
100    [mWo,nWo] = size(Wo);
101    %Find the singular value decomposition of Wo and define
102    % Vtil = [Vtil1 Vtil2],
103    %   Vtil1 = an orthonormal basis for R(Wo^T)
104    %   Vtil2 = an orthonormal basis for N(Wo)
105    [UWo,SWo,VWo] = svd(Wo);
106    Vtil1 = VWo(:,1:rDWo); Vtil2 = VWo(:,(rDWo+1):nWo);
107    %Find the singular value decomposition of products
108    % (Vtil1^T)*Vtil1 and (Vtil2^T)*Vtil2
109    %Then use program Rounding.m to make sure any singular values that
110    % should be zero are zero
111    [barU1,barS10,barV1] = svd((Vtil1')*Vtil1); barS1 = Rounding(barS10);
112    [barU2,barS20,barV2] = svd((Vtil2')*Vtil2); barS2 = Rounding(barS20);
113    %Define matrix V1 and V2; the desired smooth matrix is VSDM=[V1 V2]
114    V1 = Vtil1*barU1*(barV1'); V2 = Vtil2*barU2*(barV2');
115    %Define Q(t) = VSDM and calculate its inverse
116    Q = [V1 V2]; iQ = inv(Q);
117    %%
118    %Construct the projection PV used in calculating the derivative of VSDM
119    PV = pinv(Wo)*Wo;
120    %Calculate the first derivative of PV
121    dPV = ((pinv(Wo)*dWo*(eye(nWo) - PV))' + (pinv(Wo)*dWo*(eye(nWo) - PV)));
122    %Calculate the first derivatives of V1 and V2
123    %The generalized Lyapunov equation
124    % L*X*E^T + E*X*L^T + Q = 0
125    %solves for X using X = lyap(L,Q,[],E).
126    %The equation we're solving does not have a matrix L - use identity
127    %eye(rDWo)*GV1*barS1 + barS1*GV1*eye(rDWo)
128    % + (- (barV1') * (Vtil1') * dPV * Vtil1 * barV1) = 0
129    GammaV10 = (- (barV1') * (Vtil1') * dPV * Vtil1 * barV1);
130    GammaV1 = Rounding(GammaV10);
131    GV1 = lyap(eye(rDWo),GammaV1,[],barS1);
132    HV1 = barV1*barS1*(barV1'); dHV1 = barV1*GV1*(barV1');
133    dV1 = ((dPV*Vtil1) - (V1*dHV1))*(inv(HV1));
134    %
135    %eye (nWo-rDWo)*GV2*barS2 + barS2*GV2*eye (nWo-rDWo)

```

```

136      % + ((barV2')*(Vtil2'))*dPV*Vtil2*barV2) = 0
137      GammaV20 = (barV2')*(Vtil2')*dPV*Vtil2*barV2;
138      GammaV2 = Rounding(GammaV20);
139      GV2 = lyap(eye((nWo-rDWo)),GammaV2,[],barS2);
140      HV2 = barV2*barS2*(barV2'); dHV2 = barV2*GV2*(barV2');
141      dV2 = (((-dPV)*Vtil2) - (V2*dHV2))*(inv(HV2));
142      %Define the first derivative of Q(t)
143      dQ = [dV1 dV2];
144      %Compute the standard form for the unobservable system
145      Au = iQ*A*Q - iQ*dQ; Cu = K*Q;
146      %Confirm Au, Cu represent the desired standard form
147      CuAu = Cu*Au;
148      %Au3, the unobservable block
149      Au3 = Au(rDWo+1:nu,rDWo+1:nu); dx3 = Au3*x3;
150  else end %else onU == 0
151
152 %Reshape dWo, dPhi, and maybe dx3 in order to pass back to LTVcircuit.m
153 dWoR = reshape(dWo,nu^2,1); dPhiR = reshape(dPhi,nu^2,1);
154 if onU == 1
155     dx3R = reshape(dx3,(nu-rDWo)^2,1); dWoR2 = [dWoR;dPhiR;dx3R];
156 else %onU == 0
157     dWoR2 = [dWoR;dPhiR];
158 end
159
160 %end      WoRank.m

1 function dx = LTVDAE_Solveplus(tspan,x0,nx,mmodC,C,Cext,Cr,obs,comp,SPML,m, ...
2 q,sv,rowcount,rowsC)
3 %Function LTVDAE_Solveplus.m is called by LTVcircuit.m.
4
5 %For the full-order observer, this function finds the solutions for xtilde
6 % (the completion, the solution of the example system) and xhat (the
7 % full-order observer).
8 %For the reduced-order observer and maximally reduced observer, this
9 % function finds the solutions for xtilde (the completion, the solution of
10 % the example system), qtilde (the completion of the transformed system)
11 % and z (the observer).
12
13 %Input tspan -the values of time at which ode45 is evaluating
14 % LTVDAE_Solveplus.m.
15 %Input x0 - the initial condition vector.
16 %Input mmodC - the size (number of rows) of the extended output matrix when
17 % obs = 3 (defined as zero for obs = 1,2).
18 %See LTVcircuit.m for description of other inputs.

```

```

19
20 %This program is currently set up for the LTV circuit example, Chapter 6.
21 %Derivatives of time-varying matrices were determined outside of this
22 %function to eliminate finding symbolic derivatives within LTVDAE_Solveplus.m.
23 %*****
24
25 %Define time variable t as tspan
26 t = tspan;
27 %Determine the size of C
28 [mC,nC] = size(C);
29 %Determine the size of x0, the initial condition vector
30 [mx0,nx0] = size(x0);
31 %Initialize a zero vector to contain the results
32 dx = zeros(mx0,1);
33
34 %Define the number of unmeasurable/unknown components nu and the number of
35 % rows in the (extended) output matrix when appropriate
36 %Also, assign the initial conditions unrelated to the Riccati equation to x
37 if obs == 1
38     nu = nx; x = x0(1:2*nx,:);
39 elseif obs == 2
40     nu = nx - mC; mom = mC; x = x0(1:2*nx+nu,:);
41 else %obs == 3
42     nu = nx - mmodC; mom = mmodC; x = x0(1:2*nx+nu,:);
43 end
44
45 %Determine the coefficient matrices of the completion and calculate the
46 % solution for xtilde, the completion (solution of example system)
47 if comp == 1
48     [A,dA,B] = SLSCcoeff(t,nx,C,Cext,Cr,1,SPML,sv);
49 %Calculate V, dV for the SLSC, v=[V;dV]
50     V = 4*cos(2*t)*sin(t/5);
51     d1V = -8*sin(2*t)*sin(t/5)+(4/5)*cos(2*t)*cos(t/5);
52     d2V = -(404/25)*cos(2*t)*sin(t/5)-(16/5)*sin(2*t)*cos(t/5);
53     VdV = [V;d1V;d2V];
54     dx(1:nx,:) = A*x(1:nx,:) + B*VdV;
55 else %comp == 2
56     [A,dA,B] = ASCcoeff(t,nx,C,Cext,Cr,1,SPML,sv);
57 %For the ASC, the B that is returned comes from \mathcal{B}*v
58     dx(1:nx,:) = A*x(1:nx,:) + B;
59 end
60
61 %Calculate the output required for constructing the reduced-order and
62 % maximally reduced observers. This step is unnecessary if data is
63 % available from a physical system.

```

```

64 if obs == 2
65     y = C*x(1:nx,:);
66 elseif obs == 3
67     y = C*x(1:nx,:);
68 %Save the rows of y corresponding with the rows of C saved for
69 % extending the output matrix
70 for i = 1:(rowcount - 1)
71     ye(i,:) = y(rowsC(i),:);
72 end
73 else
74 end
75
76 %Find the solution of the observer and transformed completion when
77 % appropriate
78 if obs == 1
79 %Reshape SL0 from a vector to a matrix for the Riccati equation used in
80 % constructing the gain matrix
81     SL = reshape(x0(2*nx+1:mx0),nx,nx);
82 %Display the eigenvalues of SL
83     eig(SL);
84 %Define M, Q used in constructing the gain matrix.Notice, Q is constant.
85     M = m*eye(mC); Q = q*eye(nu) + (C')*M*C;
86 %The Riccati equation used in determining the gain matrix
87     dSL = A*SL + SL*(A') + SL*(Q - ((C')*M*C))*SL;
88 %Define gain matrix V2
89     V2 = (1/2)*SL*(C')*M;
90 %Calculate the solution for xhat, the full-order observer
91 %if/else not in proper alignment for formatting in LaTeX
92 if comp == 1
93 dx(nx+1:2*nx,:) = A*x(nx+1:2*nx,:) + B*VdV + V2*C*(x(1:nx,:)-x(nx+1:2*nx,:));
94 else %comp == 2
95 dx(nx+1:2*nx,:) = A*x(nx+1:2*nx,:) + B + V2*C*(x(1:nx,:)-x(nx+1:2*nx,:));
96 end
97 %Rewrite dSL in vector form to pass back to LTVcircuit.m
98     dx(2*nx+1:mx0,:) = reshape(dSL,nx^2,1);
99 else %obs == 2 or obs == 3
100 %Reshape SL0 from a vector to a matrix for the Riccati equation used in
101 % constructing the gain matrix
102     SL = reshape(x0(2*nx+nu+1:mx0),nu,nu);
103 %Display the eigenvalues of SL
104     eig(SL);
105 %Determine the coefficient matrices of the transformed completion
106 if comp == 1
107     [qA,qdA,qB] = SLSCcoeff(t,nx,C,Cext,Cr,obs,SPML,sv);
108 else %comp == 2

```

```

109      [qA,qdA,qB] = ASCcoeff(t,nx,C,Cext,Cr,obs,SPML,sv);
110      end
111 %Identify the submatrices of the transformed \widetilde{A}
112 AR11 = qA(1:mom,1:mom); AR12 = qA(1:mom,mom+1:nx);
113 AR21 = qA(mom+1:nx,1:mom); AR22 = qA(mom+1:nx,mom+1:nx);
114 %Display eigenvalues of AR22
115 eig(AR22);
116 %Identify the submatrices of transformed \widetilde{B} (Recall,
117 % \widetilde{B} comes from \mathcal{B}*v for ASC)
118 BR1 = qB(1:mom,:); BR2 = qB(mom+1:nx,:);
119 %Calculate the solution for qtilde, the transformed completion (solution of
120 % transformed system)
121 if comp == 1
122     dx(nx+1:2*nx,:) = qA*x(nx+1:2*nx,:) + qB*VdV;
123 else %comp == 2
124     dx(nx+1:2*nx,:) = qA*x(nx+1:2*nx,:) + qB;
125 end
126 %Define M, Q used in constructing the gain matrix. Notice, Q is time-varying
127 % since AR12 is time-varying.
128 M = m*eye(mom); Q = q*eye(nu) + (AR12')*M*AR12;
129 %The derivative of M is zero since we are choosing M to be constant
130 dM = 0;
131 %The Riccati equation used in determining the gain matrix
132 dSL = AR22*SL + SL*(AR22') + SL*(Q - ((AR12')*M*AR12))*SL;
133 %Define gain matrix V2
134 V2 = (1/2)*SL*(AR12')*M;
135 %The variable FR in the reduced-order observer
136 % \dot{z} = DR*x + FR*y + GR*v
137 %requires the first derivative of gain matrix V2.
138 %Calculate the first derivative of gain matrix V2
139 % dV2 = (1/2)*[dSL*(AR12')*M + SL*(dAR12')*M + SL*(AR12')*dM]
140 %dAR12^T = (dAR12)^T
141 dAR12T = (qdA(1:mom,mom+1:nx))';
142 %Define the first derivative of gain matrix V2
143 dV2 = (1/2)*(dSL*(AR12')*M + SL*(dAR12T)*M + SL*(AR12')*dM);
144 %Calculate the solution for z, the reduced-order observer in terms of the
145 % transformed system (let P2 be the identity)
146 DR = AR22 - V2*AR12; FR = AR21 - V2*AR11 + AR22*V2 - V2*AR12*V2 - dV2;
147 GR = BR2 - V2*BR1;
148 if obs == 2
149     if comp == 1
150         dx(2*nx+1:3*nx-mom,:) = DR*x(2*nx+1:3*nx-mom,:) + FR*y + GR*VdV;
151     else %comp == 2
152         dx(2*nx+1:3*nx-mom,:) = DR*x(2*nx+1:3*nx-mom,:) + FR*y + GR;
153     end

```

```

154     else %obs == 3
155         %GB = scriptG*scriptB was determined symbolically using MATLAB and
156         % Maple; this GB is example specific
157         %Evaluate GB at each time t
158         GB = [0 0 0;0 0 0;-1 0 0;0 3+cos(1/3*t) 0];
159         %Determine the size of GB
160         [mGB,nGB] = size(GB);
161         %if/else not in proper alignment for formatting in LaTeX
162         if comp == 1
163             dx(2*nx+1:3*nx-mmodC,:) = DR*x(2*nx+1:3*nx-mmodC,:) + FR*[GB*VdV;ye] + GR*VdV;
164         else %comp == 2
165             %[V;dV] will be used in constructing the output for the extended
166             % output equation
167             V = 4*cos(2*t)*sin(t/5);
168             d1V = -8*sin(2*t)*sin(t/5)+(4/5)*cos(2*t)*cos(t/5);
169             d2V = -(404/25)*cos(2*t)*sin(t/5)-(16/5)*sin(2*t)*cos(t/5);
170             VdV = [V;d1V;d2V];
171             dx(2*nx+1:3*nx-mmodC,:) = DR*x(2*nx+1:3*nx-mmodC,:) + FR*[GB*VdV;ye] + GR;
172         end
173     end
174     %Rewrite dSL in vector form to pass back to LTVcircuit.m
175     dx(3*nx-mom+1:3*nx-mom+nu^2,:) = reshape(dSL,nu^2,1);
176 end
177
178 %end      LTVDAE_Solveplus.m

```

```

1 function J = ExtendOutput(nx,C,obs,comp,SPML,t0,tf,treset,s,m,q,sv)
2
3 %Function ExtendOutput.m is called by RunLTV.m.
4
5 %The purpose of ExtendOutput.m is to determine a general (symbolic)
6 %definition for the extended output matrix and its corresponding output. If
7 %the extended output matrix is nonsingular, then the state is known without
8 %constructing an observer.
9
10 %This function calls LTVcircuit.m when constructing a maximally reduced
11 %observer and LTVComp_Solve.m if the extended output matrix is nonsingular.
12
13 %Description of inputs
14 %nx -the number of state variables.
15 %C -output matrix. Note, only constant output matrices have been tested.
16 % Trying a time-varying output matrix is an item for future research.
17 %obs -designates which observer is being constructed. Only '3' will be
18 % passed to ExtendOutput.m since this program is needed for only the
19 % maximally reduced observer. This input is not used in ExtendOutput.m but

```

```

20 % is being passed through to LTVcircuit.m.
21 %comp -designates which completion is being used (1 for (S)LSC, 2 for ASC).
22 %t0,tf -initial, final times in the time interval used when calling ode45.
23 %s,m,q -values used when constructing the gain matrix. These inputs are not
24 % used in ExtendOutput.m but are being passed through to LTVcircuit.m.
25 %sv -switch vector (on/off identifiers to let MATLAB know what section of
26 % code to run).
27
28 %This program is currently set up for the LTV circuit example, Chapter 6.
29 %Derivatives of time-varying matrices were determined outside of this
30 % function to eliminate finding symbolic derivatives within ExtendOutput.m.
31 %*****  

32
33 %Define symbolic variable t
34 syms t
35 %Assign switch: positive resistor if on1 = 0, negative resistor if on1 = 1
36 on1 = sv(1);
37 %Determine the size of C
38 [mC,nC] = size(C);
39
40 %Recall, the equation characterizing the solution manifold is
41 % (scriptG*scriptF)x = (scriptG*scriptB)v
42 %GF = scriptG*scriptF was determined symbolically using MATLAB and Maple;
43 % this GF is example specific
44 if on1 == 0 %Positive Resistors
45     GF = [-1 1 0 -4-2*sin(t) 0 0;1 0 0 0 -2-sin(t) 0;
46             1 0 0 0 0 0;-1/3*sin(1/3*t) 0 0 -1 1 -1];
47 else %on1 == 1, Negative Resistors
48     GF = [-1 1 0 4+2*sin(t) 0 0;1 0 0 0 2+sin(t) 0;
49             1 0 0 0 0 0;-1/3*sin(1/3*t) 0 0 -1 1 -1];
50 end
51 %Determine the size of GF
52 [mGF,nGF] = size(GF);
53 %Calculate the difference between the number of state variables and the
54 % number of rows of GF (nre = number of rows from C needed to extend the
55 % output matrix so it becomes nonsingular)
56 nre = nx - mGF;
57 %Initialize vector rowsC to keep track of the rows of C used to extend the
58 % output matrix
59 rowsC = zeros(nre,1);
60
61 %The first mGF rows of the extended output matrix EO equal GF
62 EO = GF;
63 %Set counter to 1; rowcount keeps track of how many rows of C have been
64 % used to extend the output matrix

```

```

65 rowCount = 1;
66 %Use a for loop to identify the rows of C linearly independent from GF
67 for i = 1:mC
68     %Define a test matrix for testing linear independence
69     test = [EO;C(i,:)];
70     %Determine the size of test and calculate its rank
71     [mtest,ntest] = size(test); rtest = rank(test);
72     %If test is full row rank, redefine EO as test, save the number of C's
73     % row used, and increase rowCount by 1. Also, define Cext as the
74     % submatrix of C used to extend the output matrix.
75     if rtest == mtest
76         EO = test; rowsC(rowCount) = i; Cext(rowCount,:) = C(i,:);
77         rowCount = rowCount + 1;
78     else end
79     %If rowCount is greater than nre, then the extended output matrix is
80     % nonsingular (no more rows of C are required for the extension) and
81     % break the for loop
82     if rowCount > nre
83         break
84     else end
85 end
86 %Display extended output matrix
87 EO
88 %GB = scriptG*scriptB was determined symbolically using MATLAB and Maple;
89 % this GB is example specific
90 GB = [0 0 0;0 0 0;-1 0 0;0 3+cos(1/3*t) 0];
91 %Calculate V, dV; this input is example specific
92 V = 4*cos(2*t)*sin(t/5);
93 d1V = -8*sin(2*t)*sin(t/5)+(4/5)*cos(2*t)*cos(t/5);
94 d2V = -(404/25)*cos(2*t)*sin(t/5)-(16/5)*sin(2*t)*cos(t/5);
95 dV = [d1V;d2V];
96 %GB*v, v = [V;dV], for constructing the extended output equation
97 GBv = GB*[V;dV];
98
99 %Determine the size of EO
100 [mEO,nEO] = size(EO);
101 %If EO is nonsingular, then determine the state without constructing an
102 % observer
103 if mEO == nx
104     %Time span; keep the number of elements in resulting vectors uniform
105     tspan = t0:0.01:tf;
106     %Define initial condition for completion
107     xtilde0 = [0;0;0;0;0;-16/5];
108     %Define a new tolerance level for ode45
109     options = odeset('RelTol',1e-9);

```

```

110      %Find completion solution
111      [tlong,xlong] = ...
112          ode45(@LTVComp_Solve,tspan,xtilde0,options,nx,comp,SPML,sv);
113      %Determine the size of tlong
114      [mtlong,ntlong] = size(tlong);
115      %xlong is returned as an mtlong by nx matrix
116      %Transpose xlong so x is nx by mtlong
117      x = xlong';
118      %Calculate the output of the LTV system of DAEs
119      y = C*x;
120      %Save the rows of y corresponding with the rows of C saved for
121      % extending the output matrix
122      for i = 1:(rowcount - 1)
123          ye(i,:) = y(rowsC(i),:);
124      end
125
126      for j = 1:mtlong
127          %Evaluate GBv at the time values returned in tlong
128          GBvt = subs(GBv,{t},{tlong(j)} );
129          %Evaluate EO at the time values returned in tlong
130          EOt = subs(EO,{t},{tlong(j)} );
131          %Construct the output EOE for the extended output equation
132          %The output from the completion is being used, but for a physical
133          % system, the output would come from measured values
134          EOE = [GBvt;ye(:,j)];
135          %The state variables evaluated at the time values returned in tlong
136          % (xf = inv([GF;Cext])*[GBv;ye])
137          xf(:,j) = inv(EOt)*EOE;
138      end
139      %Plot the solution found using the information from the solution manifold
140      % and original output equation
141      figure
142      plot(tlong,xf)
143      hgsave('xNO');
144      %Plot the completion and xf on the same graph
145      figure
146      hold on
147      plot(tlong,x), plot(tlong,xf,'--')
148      hold off
149      hgsave('xCompAndNO');
150      %Plot the difference between the completion and xf
151      figure
152      plot(tlong,(x-xf))
153      hgsave('xCompMinusNO');

```

```

154 %else find Cr for constructing the nonsingular transformation matrix Rbar
155 else
156     %Symbolically determine Cr
157     Crsyms = (null(EO))';
158     %If Cr is not constant, Cr will need to be hard coded into programs
159     % where necessary
160     %Assuming Cr is constant, MATLAB command double is used instead of subs
161     %Note, an error will occur if Cr is not constant
162     Cr = double(Crsyms);
163     display('The extended output matrix is not nonsingular.')
164     display('The transformation matrix \overline{R} is')
165     Rbar = [EO;Crsyms]
166     display('The row numbers of the saved rows of output matrix C are')
167     rowsC
168     display('The row count is')
169     rowcount
170     display('LTVcircuit.m has been called.')
171     LTVcircuit(nx,C,Cext,Cr,obs,comp,SPML,t0,tf,treset,s,m,q,sv,rowcount,rowsC)
172 end
173
174 %end      ExtendOutput.m

1 function dx = LTVComp_Solve(tspan,x0,nx,comp,SPML,sv)
2
3 %Function LTVComp_Solve.m is called by ExtendOutput.m.
4
5 %Function LTVComp_Solve.m finds the solution for xtilde (the completion,
6 % the solution of the example system).
7
8 %Input tspan –the values of time at which ode45 is evaluating
9 % LTVDAE_Solveplus.m.
10 %Input x0 – the initial condition vector.
11 %See ExtendOutput.m for description of other inputs.
12
13 %This program is currently set up for the LTV circuit example, Chapter 6.
14 %Derivatives of time-varying matrices were determined outside of this
15 %function to eliminate finding symbolic derivatives within LTVComp_Solve.m.
16 %*****
17
18 %Define time variable t as tspan
19 t = tspan;
20 %Determine the size of x0, the initial condition vector
21 [mx0,nx0] = size(x0);
22 %Initialize a zero vector to contain the results
23 dx = zeros(mx0,1);

```

```

24
25 %Determine the coefficient matrices of the completion and calculate the
26 % solution for xtilde, the completion (solution of example system)
27 %C, Cext, Cr are input as zero
28 C = 0; Cext = 0; Cr = 0;
29 %The completion for the original system (associated with the full-order
30 % observer) is required
31 obs = 1;
32 if comp == 1
33     [A,dA,B] = SLSCcoeff(t,nx,C,Cext,Cr,obs,SPML,sv);
34     %Calculate V, dV for the SLSC, v=[V;dV]
35     V = 4*cos(2*t)*sin(t/5);
36     d1V = -8*sin(2*t)*sin(t/5)+(4/5)*cos(2*t)*cos(t/5);
37     d2V = -(404/25)*cos(2*t)*sin(t/5)-(16/5)*sin(2*t)*cos(t/5);
38     VdV = [V;d1V;d2V];
39     dx(1:nx,:) = A*x0(1:nx,:) + B*VdV;
40 else %comp == 2
41     [A,dA,B] = ASCcoeff(t,nx,C,Cext,Cr,obs,SPML,sv);
42     %For the ASC, the B that is returned comes from \mathcal{B}^*v
43     dx(1:nx,:) = A*x0(1:nx,:) + B;
44 end
45
46 %end      LTVComp_Solve.m

```

```

1 %script CompUnobs.m
2
3 %Script CompUnobs.m compares full-order and reduced-order observers'
4 % unobservable subspaces for a linear time-varying system of DAEs.
5 %Based on theory in Section 4.3 of Bobinyec dissertation
6
7 %Use load('') to load information from full-order OBSDETINFO
8 load('OBSDETINFOfullorder.mat') %for example
9 %Assign basis vectors for nullspace of (observability) Gramian
10 yfoo = NullWo;
11 %Assign the rank vector of the Gramian
12 rankfoo = rDWo;
13 %Use load('') to load information from reduced-order OBSDETINFO
14 load('OBSDETINFOreducedorder.mat') %for example
15 yroo = NullWo;
16 rankroo = rDWo;
17
18 %Determine the sizes of the basis vectors
19 [xfsize,yfsize,zfsize] = size(yfoo);
20 [xrsize,yrsize,zrsize] = size(yroo);
21 %Determine the sizes of the rank vectors

```

```

22 [xrankfsize,yrankfsize] = size(rankfoo);
23 [xrankrsize,yrankrsize] = size(rankroo);
24
25 %Initialize maximum rank values for full-order pair's Gramian and the
26 % reduced-order pair's Gramian
27 maxrankf = 0; maxrankr = 0;
28 %Initialize a variable for keeping track of when the maximum ranks occur
29 tmrf = 0; tmrr = 0;
30 %Determine when the maximum Gramian ranks first occur for the two observers
31 for i = 1:yrankfsize
32     if maxrankf < rankfoo(i)
33         maxrankf = rankfoo(i); tmrf = i; %full-order
34     else end
35 end
36 for j = 1:yrankrsize
37     if maxrankr < rankroo(j)
38         maxrankr = rankroo(j); tmrr = j; %reduced-order
39     else end
40 end
41 %Let tindex be the later index of tmrf and tmrr
42 tindex = max(tmrf,tmrr);
43 %Initialize an xfszie by yrszie by zrszie array for the extension of the
44 % Gramian's basis vectors for the reduced-order pair
45 yzroo = zeros(xfszie,yrszie,zrszie);
46 %Beginning the index at tindex, determine if the full-order and the
47 % extended reduced-order basis vectors are linearly independent
48 %Since tindex is probably not 1, let icount = 1
49 icount = 1;
50 %Initialize a vector for storing ranks
51 rankFR = zeros(zrszie-tindex+1,1);
52 for i = tindex:zrszie
53     %Extend the reduced-order basis vector
54     yzroo(xfszie-xrszie+1:xfszie,:,:i) = yroo(:,:,:i);
55     %Construct an augmented matrix with the full-order and extended
56     % reduced-order basis vectors
57     aug = [yfoo(:,:,:i) yzroo(:,:,:i)];
58     %Determine the rank of the augmented matrix
59     [U,S0,V] = svd(aug);
60     S = Rounding(S0);
61     rankFR(icount) = rank(S);
62     icount = icount + 1;
63 %If rankFR equals maxrankf and maxrankr, then the basis vectors are
64 % linearly independent and the full-order and reduced-order observers'
65 % unobservable subspaces are related
66 hold on

```

```

67      plot(i,rankFR(icount-1))
68  end
69 hold off
70
71 %end      CompUnobs.m

```

B.3 Code for both Examples

```

1 function J = Rounding(matrix)
2
3 %Function Rounding.m can be called by any program.
4
5 %The purpose of Rounding.m is to force elements of a matrix that should be
6 % zero to be zero. This program is meant for matrices calculated by MATLAB
7 % subject to internal rounding errors.
8
9 %Define a tolerance level for rounding
10 tol = (1*(10^(-10)));
11 %Determine the size of the input matrix
12 [m,n] = size(matrix);
13 %If an entry of the matrix is within [-tol,tol], redefine as 0
14 for i=1:m
15     for j=1:n
16         if (-tol <= matrix(i,j)) & (matrix(i,j) <= tol)
17             matrix(i,j) = 0;
18         else end
19     end
20 end
21
22 J = matrix;
23
24 %end      Rounding.m

```

```

1 function J = DAEobservable(C,onEx,on1)
2
3 %Function DAEobservable.m checks whether or not a linear system of DAEs is
4 % observable based on the special observability matrix constructed in
5 % S. L. Campbell and W. J. Terrell, ``Observability of linear time-varying
6 % descriptor systems'', \textit{SIAM J. Matrix Anal. Appl.}, 12 (1991),
7 % pp. 484--496.
8
9 %The constant output matrix C is input by the user.
10 % If C is time-varying, this program needs to be modified.

```

```

11 %Input onEx is a switch. If onEx = 0, the code for the linear
12 % time-invariant example is called; if onEx = 1, the code for the linear
13 % time-varying example is called.
14 %Input onl is a switch for the linear time-varying example. If onl = 0,
15 % positive resistors; if onl = 1, negative resistors.
16 %*****
17
18 if onEx == 0
19 %Define the number of state variables
20 nx = 5;
21 %Define the index of the system of DAEs
22 ind = 3;
23 %Constant coefficient matrices from example system
24 K = [-2 1;1 -2]; S = (1/4)*[1 0;0 1]; H = [1 -1];
25 %Identity matrix and zero matrices and vectors for building the matrices of
26 % the matrix pencil {E,F}
27 I = eye(2); Z2b2 = zeros(2,2); Z2b1 = zeros(2,1); Z1b2 = zeros(1,2);
28 Z1b1 = zeros(1,1); Z5b2 = zeros(5,2); Z5b5 = zeros(5,5);
29 %Build the matrix pencil; E is 5x5 and F is 5x5
30 E = [I Z2b2 Z2b1;Z2b2 I Z2b1;Z1b2 Z1b2 Z1b1];
31 F = -[Z2b2 I Z2b1;K S H';H Z1b2 Z1b1];
32 %Build scriptE and scriptF from the derivative array equations (\lambda=0)
33 scriptE = [E Z5b5 Z5b5 Z5b5; F E Z5b5 Z5b5; Z5b5 F E Z5b5; Z5b5 Z5b5 F E];
34 scriptF = [F;Z5b5;Z5b5;Z5b5];
35 %Define variable kn for building scriptC
36 % kn = d - 1, where d is the dimension of the solution manifold
37 kn = 2 - 1;
38 %
39 else %onEx == 1
40 %
41 %Define symbolic variable t
42 syms t
43 %Define the number of state variables
44 nx = 6;
45 %Define the index of the system of DAEs
46 ind = 2;
47 %Construct the coefficient matrices from the derivative array equations
48 %The coefficient matrix of \dot{x} from the LTV example system of DAEs and
49 % its necessary derivatives
50 E = [3+cos(t/3) 0 0 0 0 0;0 2-cos(2*t) 0 0 0 0;0 0 2-exp(-t) 0 0 0;
51 0 0 0 0 0;0 0 0 0 0;0 0 0 0 0];
52 dE = [-(1/3)*sin(t/3) 0 0 0 0 0;0 2*sin(2*t) 0 0 0 0;0 0 exp(-t) 0 0 0;
53 0 0 0 0 0;0 0 0 0 0;0 0 0 0 0];
54 ddE = [-(1/9)*cos(t/3) 0 0 0 0 0;0 4*cos(2*t) 0 0 0 0;0 0 -exp(-t) 0 0 0;
55 0 0 0 0 0;0 0 0 0 0;0 0 0 0 0];

```

```

56 %Determine the size of E
57 [mE,nE] = size(E);
58 %The coefficient matrix of x from the LTV example system of DAEs and
59 % its necessary derivatives
60 if on1 == 0
61     F = [-(1/3)*sin(t/3) 0 0 -1 1 -1;0 2*sin(2*t) 1 1 0 0;0 -1 exp(-t) 0 0 0;
62         -1 1 0 -(4+2*sin(t)) 0 0;1 0 0 0 -(2+sin(t)) 0;1 0 0 0 0 0];
63     dF = [-(1/9)*cos(t/3) 0 0 0 0 0;0 4*cos(2*t) 0 0 0 0;0 0 -exp(-t) 0 0 0;
64         0 0 0 -2*cos(t) 0 0;0 0 0 -cos(t) 0;0 0 0 0 0 0];
65     ddF = [(1/27)*sin(t/3) 0 0 0 0 0;0 -8*sin(2*t) 0 0 0 0;0 0 exp(-t) 0 0 0;
66         0 0 0 2*sin(t) 0 0;0 0 0 sin(t) 0;0 0 0 0 0];
67 else %on1 == 1
68     F = [-(1/3)*sin(t/3) 0 0 -1 1 -1;0 2*sin(2*t) 1 1 0 0;0 -1 exp(-t) 0 0 0;
69         -1 1 0 (4+2*sin(t)) 0 0;1 0 0 0 (2+sin(t)) 0;1 0 0 0 0 0];
70     dF = [-(1/9)*cos(t/3) 0 0 0 0 0;0 4*cos(2*t) 0 0 0 0;0 0 -exp(-t) 0 0 0;
71         0 0 0 2*cos(t) 0 0;0 0 0 cos(t) 0;0 0 0 0 0 0];
72     ddF = [(1/27)*sin(t/3) 0 0 0 0 0;0 -8*sin(2*t) 0 0 0 0;0 0 exp(-t) 0 0 0;
73         0 0 0 -2*sin(t) 0 0;0 0 0 -sin(t) 0;0 0 0 0 0];
74 end
75 %The coefficient matrix of x from the LTV example system of DAEs and
76 % its necessary derivatives
77 %Initialize a 6 by 6 zero matrix and a 6 by 1 zero vector
78 z6b6 = zeros(6); z6b1 = zeros(6,1);
79 %Build scriptE and scriptF (\lambda is 0 so its terms have been removed)
80 scriptE = [E z6b6 z6b6;dE+F E z6b6;ddE+(2*dF) (2*dE)+F E];
81 scriptF = [F;dF;ddF];
82 %Define variable kn for building scriptC
83 % kn = d - 1, where d is the dimension of the solution manifold
84 kn = 2 - 1;
85 end %end derivative array equations construction
86
87 %*****
88 %Build scriptC
89 %scriptC = [Ctilde | Chat]
90 %      = [C      | 0      ...    ... 0]
91 %      [C'     | C      0      ...   :]
92 %      [C''    | 2C'    C      ...   :]
93 %      [*     | *      *      *   0]
94 %      [C^(kn)| *      *      *   C]
95 %Determine the size of C
96 [mC,nC] = size(C);
97 %Zero matrix used in building scriptC
98 ZmCbnC = zeros(mC,nC);
99 for i = 1:1+kn
100    for j = 1:1+kn

```

```

101      if i == j
102          scriptC(((i-1)*mC + 1):i*mC,((j-1)*nC + 1):j*nC) = C;
103      else
104          scriptC(((i-1)*mC + 1):i*mC,((j-1)*nC + 1):j*nC) = ZmCbnC;
105      end
106  end
107 end
108 %Display scriptC
109 scriptC
110
111 %Construct the observability matrix for a system of DAEs,
112 % where scriptO = [scriptE scriptF; [Chat 0] Ctildel]
113 %Determine the sizes of scriptC and scriptE
114 [mscriptC,nscriptC] = size(scriptC);
115 [mscriptE,nscriptE] = size(scriptE);
116 %Zero matrix used in building scriptO
117 ZLO = zeros((kn+1)*mC,((ind+1-kn)*nC)); %LO = leftover
118 %Build [Chat 0] for insertion into scriptO
119 Chatplus = [scriptC(:,(nC+1):nscriptC) ZLO];
120 %Construct and display scriptO
121 scriptO = [scriptE scriptF;Chatplus scriptC(:,1:nC)]
122
123 %*****%
124 %Determine if scriptO is 1-full
125 % (if scriptO is 1-full, the system of DAEs is observable)
126 if onEx == 0
127 %To check 1-fullness,
128 % scriptO*b equals zero implies the first n entries of b are zero;
129 % b is in the nullspace of scriptO.
130 %For any of the last 'n-r' columns of V, their first n entries should be 0.
131 %This method of checking 1-fullness is restricted to linear
132 % time-invariant systems of DAEs. The method for linear time-varying
133 % systems of DAEs should be modified to focus on the reduced structure.
134
135 %Initialize zerosum so the first norm has something to which to add
136 zerosum = 0;
137 %Calculate svd(scriptO) to find V
138 [U,S,V] = svd(scriptO); SR = Rounding(S); rSR = rank(SR);
139 [mscriptO,nscriptO] = size(scriptO);
140 for i = (rSR+1):nscriptO
141     zerocheck = norm(V(1:nC,i)); zerosum = zerosum + zerocheck;
142 end
143 %Make zerosum zero if it is meant to be zero
144 zerosumR = Rounding(zerosum);
145 if zerosumR == 0

```

```

146     disp('1-fullness holds')
147 else
148     disp('1-fullness fails')
149 end
150 %
151 else %onEx == 1
152 %
153 %The method of checking 1-fullness for a linear time-varying system of DAEs
154 % focuses on the reduced structure.
155 %Row reduce scriptO
156 rrscriptO = rref(scriptO);
157 %Determine the size of scriptO
158 [mscriptO,nscriptO] = size(scriptO);
159 %For 1-fullness to hold...
160 %This submatrix should be an (mE x mE) identity block
161 subscriptO1 = rrscriptO(1:mE,1:mE);
162 %This submatrix should be an (mE x (nscriptE-mE)) zero block
163 subscriptO2 = rrscriptO(1:mE,(mE+1):nscriptO);
164 %This submatrix should be an ((mscriptE-mE) x mE) zero block
165 subscriptO3 = rrscriptO((mE+1):mscriptO,1:mE);
166 %Check for identity block
167 if subscriptO1 == eye(mE)
168     %Check for zero block
169     if subscriptO2 == zeros(mE, (nscriptO-mE))
170         %Check for zero block
171         if subscriptO3 == zeros((mscriptO-mE),mE)
172             display('1-fullness holds')
173         else
174             display('1-fullness fails due to lack of zero block, rows ...
175                     beneath identity block')
176         end
177     else
178         display('1-fullness fails due to lack of zero block, columns next ...
179                     to identity block')
180     end
181 else
182     display('1-fullness fails due to lack of identity block')
183 end %end 1-fullness check
184
185 %end      DAEobservable.m

```